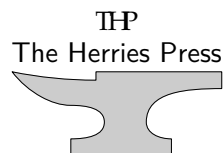


The

Memoir

Class

The Memoir Class
for
Configurable Typesetting
User Guide
Peter Wilson



© 2001, 2002, 2003, 2004, 2008 Peter R. Wilson
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 17 16 15 14 13 12 11

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

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 · xvi

List of Tables · xix

List of typeset examples · xxi

Preface · xxiii

Introduction to the seventh edition · xxv

Terminology · xxxiii

I Art and Theory **1**

1 The Parts of a Book · 3

2 The page · 13

3 Styling the elements · 43

4 Picky points · 51

II Practice **61**

5 Starting off · 63

6 Laying out the page · 69

7 Text and fonts · 93

8 Titles · 107

9	Abstracts	117
10	Document divisions	121
11	Pagination and headers	163
12	Paragraphs and lists	177
13	Contents lists	189
14	Floats and captions	213
15	Rows and columns	249
16	Page notes	273
17	Decorative text	287
18	Poetry	293
19	Boxes, verbatims and files	309
20	Cross referencing	333
21	Back matter	337
22	Miscellaneous	361
23	An example design	385
A	Packages and macros	395
B	LaTeX and TeX	399
C	The terror of errors	415
	Command summary	435
	Bibliography	473
	Index	481
	Index of first lines	523

Contents

Short contents	vii
Contents	ix
List of Figures	xvi
List of Tables	xix
List of typeset examples	xxi
Preface	xxiii
Introduction to the seventh edition	xxv
General considerations	xxv
Class options	xxvi
Sectioning styles	xxvi
Captions	xxvii
Tables	xxvii
Verse	xxviii
End matter	xxviii
Miscellaneous	xxix
Packages	xxix
Resources	xxx
Type conventions	xxxi
Acknowledgements	xxxi
Terminology	xxxiii
Units of measurement	xxxiv
I Art and Theory	1
1 The Parts of a Book	3
1.1 front matter	3
1.1.1 <i>Copyright page 4</i>	
1.2 Main matter	5
1.3 Back matter	6

1.4	Signatures and casting off	6
1.5	Paper	9
2	The page	13
2.1	The shape of a book	13
	2.1.1 <i>The golden section and Fibonacci series</i> 16	
2.2	The spread	18
	2.2.1 <i>A geometric construction</i> 31	
2.3	The typeblock	31
	2.3.1 <i>Page color</i> 34, 2.3.2 <i>Legibility</i> 35, 2.3.3 <i>Widows and orphans</i> 38, 2.3.4 <i>Paragraphs and versals</i> 38, 2.3.5 <i>Footnotes</i> 40	
2.4	Folios	40
2.5	Headers and footers	40
2.6	Electronic books	41
3	Styling the elements	43
3.1	Front matter	43
	3.1.1 <i>Title pages</i> 43, 3.1.2 <i>Copyright page</i> 43, 3.1.3 <i>Dedication</i> 44, 3.1.4 <i>Foreword and preface</i> 44, 3.1.5 <i>Acknowledgements</i> 44, 3.1.6 <i>Contents and illustration lists</i> 44, 3.1.7 <i>Introduction</i> 44, 3.1.8 <i>Part title page</i> 44	
3.2	Main matter	44
	3.2.1 <i>Chapter openings</i> 44, 3.2.2 <i>Mixed portrait and landscape pages</i> 45, 3.2.3 <i>Extracts</i> 45, 3.2.4 <i>Footnotes and endnotes</i> 46	
3.3	Back matter	47
	3.3.1 <i>Appendices</i> 47, 3.3.2 <i>Endnotes</i> 48, 3.3.3 <i>Bibliography</i> 48, 3.3.4 <i>Glossary</i> 48, 3.3.5 <i>Index</i> 48	
3.4	Type size	48
3.5	Poems and plays	49
	3.5.1 <i>Poetry</i> 49, 3.5.2 <i>Plays</i> 49	
4	Picky points	51
4.1	Word and line spacing	51
4.2	Abbreviations and acronyms	51
4.3	Dashes and ellipses	53
4.4	Punctuation	54
	4.4.1 <i>Quotation marks</i> 54, 4.4.2 <i>Footnote marks</i> 55, 4.4.3 <i>Font changes</i> 55	
4.5	Narrow measures	56
4.6	Emphasis	57
4.7	Captions and legends	57
4.8	Tables	58
4.9	Number formatting	58
II	Practice	61
5	Starting off	63
5.1	Stock paper size options	63

5.2	Type size options	64
	5.2.1 <i>Extended font sizes</i> 65	
5.3	Printing options	66
5.4	Other options	66
5.5	Remarks	67
6	Laying out the page	69
6.1	Introduction	69
6.2	Stock material	69
6.3	The page	70
6.4	The typeblock	75
6.5	Headers, footers and marginal notes	80
6.6	Putting it together	81
6.7	Side margins	85
6.8	Printing and viewing	85
6.9	Example	86
	6.9.1 <i>The page layout of this manual</i> 87	
6.10	Predefined layouts	89
7	Text and fonts	93
7.1	Fonts	93
7.2	Font sizes	97
7.3	Spaces	101
	7.3.1 <i>Paragraphs</i> 101, 7.3.2 <i>Double spacing</i> 102	
7.4	Overfull lines	104
7.5	Sloppybottom	104
8	Titles	107
8.1	Styling the titling	108
8.2	Styling the thanks	114
9	Abstracts	117
9.1	Styling	117
9.2	One column abstracts	119
10	Document divisions	121
10.1	Logical divisions	121
10.2	Sectional divisions	121
	10.2.1 <i>Appendices</i> 123	
10.3	Numbering	124
10.4	Book and part headings	125
	10.4.1 <i>Leadpage</i> 127	
10.5	Chapter headings	127
	10.5.1 <i>Defining a chapter style</i> 130, 10.5.2 <i>Further chapterstyles</i> 135, 10.5.3 <i>Chappell</i> 145, 10.5.4 <i>Demo, Demo2 and demo3</i> 147, 10.5.5 <i>Peder-</i> <i>sen</i> 148, 10.5.6 <i>Southall</i> 149, 10.5.7 <i>Veelo</i> 149, 10.5.8 <i>Chapter precis</i> 150	
10.6	Lower level headings	152

10.7	Fancy anonymous breaks	156
10.8	Footnotes in division headings	159
10.9	Predefined heading styles	159
11	Pagination and headers	163
11.1	Pagination and folios	163
11.2	Page styles	164
11.3	Making headers and footers	166
	11.3.1 <i>Example pagestyles</i> 169, 11.3.2 <i>Index headers</i> 173, 11.3.3 <i>Float pages</i> 174	
12	Paragraphs and lists	177
12.1	Paragraphs	177
	12.1.1 <i>Block paragraph</i> 177, 12.1.2 <i>Hanging paragraphs</i> 178	
12.2	Flush and ragged	179
12.3	Quotations	180
12.4	Changing the textwidth	180
12.5	Lists	182
13	Contents lists	189
13.1	General ToC methods	189
13.2	The class ToC methods	194
	13.2.1 <i>Changing the titles</i> 194, 13.2.2 <i>Typesetting the entries</i> 196, 13.2.3 <i>Example: No section number</i> 203, 13.2.4 <i>Example: Multicolumn entries</i> 204, 13.2.5 <i>Example: Multiple contents</i> 204	
13.3	New ‘List of...’ and entries	207
	13.3.1 <i>Example: plates</i> 210	
13.4	Chapter precis	211
14	Floats and captions	213
14.1	New float environments	213
14.2	Setting off a float	214
14.3	Multiple floats	216
14.4	Where LaTeX puts floats	220
14.5	Captions	225
14.6	Caption styling	225
14.7	Continuation captions and legends	229
14.8	Bilingual captions	234
14.9	Subcaptions	236
14.10	Side captions	240
	14.10.1 <i>Tweaks</i> 241	
14.11	How LaTeX makes captions	243
14.12	Footnotes in captions	246
15	Rows and columns	249
15.1	General	249
15.2	The preamble	249

15.2.1	<i>D</i> column specifiers	251,	15.2.2	Defining new column specifiers	253,
15.2.3	Surprises	254			
15.3	The array environment	255			
15.4	Tables	257			
15.5	Fear's rules	259			
15.5.1	Fills	261			
15.6	Tabular environments	263			
15.6.1	Examples	263			
15.7	Spaces and rules	266			
15.7.1	Spacing	266,	15.7.2	Special variations on horizontal lines	267,
15.7.3	Handling of rules	268			
15.8	Free tabulars	268			
15.8.1	Continuous tabulars	269,	15.8.2	Automatic tabulars	270
16	Page notes	273			
16.1	Footnotes	273			
16.1.1	A variety of footnotes	274,	16.1.2	Styling	277
16.2	Marginal notes	280			
16.3	Side notes	281			
16.4	Sidebars	283			
17	Decorative text	287			
17.1	Epigraphs	287			
17.2	General	288			
17.3	Epigraphs before chapter headings	289			
17.3.1	Epigraphs on book or part pages	292			
18	Poetry	293			
18.1	Classy verse	295			
18.1.1	Indented lines	299,	18.1.2	Numbering	300
18.2	Titles	300			
18.2.1	Main Poem Title layout parameters	301,	18.2.2	Detailed Poem Title layout parameters	302
18.3	Examples	302			
	<i>A Limerick</i> 303, <i>Love's lost</i> 303, <i>Fleas</i> 304, <i>In the beginning</i> 304, <i>Mathematics</i> 305, <i>The Young Lady of Ryde</i> 306, <i>Clementine</i> 306, <i>Mouse's Tale</i> 308				
19	Boxes, verbatims and files	309			
19.1	Boxes	310			
19.2	Long comments	313			
19.3	Verbatims	315			
19.3.1	Boxed verbatims	317,	19.3.2	New verbatims	319,
19.3.3	Example: the <code>lcode</code> environment	320			
19.4	Files	321			
19.4.1	Writing to a file	322,	19.4.2	Reading from a file	323,
19.4.3	Example: endnotes	323,	19.4.4	Example: end floats	325,
19.4.5	Example: questions and answers	328			

19.5	Answers	331
20	Cross referencing	333
20.1	Labels and references	333
20.2	Reference by name	334
21	Back matter	337
21.1	Bibliography	337
	21.1.1 <i>BibTeX</i> 339	
21.2	Index	340
	21.2.1 <i>Printing an index</i> 340, 21.2.2 <i>Preparing an index</i> 342, 21.2.3 <i>MakeIndex</i> 344, 21.2.4 <i>Controlling MakeIndex output</i> 347, 21.2.5 <i>Indexing and the natbib package</i> 349	
21.3	Glossaries	350
	21.3.1 <i>Controlling the glossary</i> 351	
21.4	Endnotes	355
	21.4.1 <i>Changing the appearance</i> 358	
22	Miscellaneous	361
	<i>In which we talk of many things, but not shoes or ships or sealing wax, nor cabbages and kings.</i>	
22.1	Draft documents	361
22.2	Change marks	361
22.3	Trim marks	362
22.4	Sheet numbering	363
22.5	Gatherings or signatures	364
22.6	Time	365
22.7	Page breaks before lists	365
22.8	Changing counters	365
22.9	New new and provide commands	366
22.10	Changing macros	367
22.11	String arguments	368
22.12	Odd/even page checking	368
22.13	Moving to another page	369
22.14	Number formatting	370
	22.14.1 <i>Numeric numbers</i> 370, 22.14.2 <i>Named numbers</i> 371, 22.14.3 <i>Fractions</i> 373	
22.15	An array data structure	374
22.16	Checking the processor	375
	22.16.1 <i>Checking for pdfLaTeX</i> 375, 22.16.2 <i>Checking for etex</i> 376, 22.16.3 <i>Checking for XeTeX</i> 376	
22.17	Leading	376
22.18	Minor space adjustment	377
22.19	Adding a period	377
22.20	Words and phrases	377
22.21	Symbols	379
	22.21.1 <i>Two simple macros</i> 379, 22.21.2 <i>Vertical centering</i> 379	

22.22	For package writers	379
22.22.1	<i>Emulating packages</i> 379, 22.22.2 <i>Inserting code before and after a file, package or class</i> 380	
22.23	Heading hooks	382
22.23.1	<i>Documenting LaTeX commands</i> 383	
23	An example design	385
23.1	Introduction	385
23.2	Design requirements	385
23.3	Specifying the page and typeblock	386
23.4	Specifying the sectional titling styles	388
23.4.1	<i>The chapter style</i> 388, 23.4.2 <i>Lower level divisions</i> 388	
23.5	Specifying the pagestyle	389
23.6	Captions and the ToC	390
23.7	Preamble or package?	391
A	Packages and macros	395
A.1	Packages	395
A.2	Macros	396
B	LaTeX and TeX	399
B.1	The TeX process	400
B.2	LaTeX files	401
B.3	Syntax	402
B.4	(La)TeX commands	403
B.5	Calculation	405
B.5.1	<i>Numbers</i> 405, B.5.2 <i>Lengths</i> 407	
B.6	Programming	410
C	The terror of errors	415
C.1	TeX messages	415
C.1.1	<i>TeX capacity exceeded</i> 422	
C.2	LaTeX errors	424
C.3	LaTeX warnings	427
C.4	Class errors	429
C.5	Class warnings	432
	Command summary	435
	Bibliography	473
	Index	481
	Index of first lines	523

List of Figures

2.1	Some page proportions	15
2.2	Two spreads: Canada, 1992 and England, 1970	20
2.3	Two spreads: USA, 1909 and England, 1964.	21
2.4	Two spreads: France, 1559 and Canada, 1995	21
2.5	Two spreads: USA, 1949 and 1990	22
2.6	Two spreads: England, 1908 and USA, 1993	23
2.7	Two spreads: USA, 1931 and England, 1968	23
2.8	Two spreads: USA, 1994 and England, 1988	24
2.9	Title page design based on <i>The Thames and Hudson Manual of Typography</i> (1988)	25
2.10	Two spreads: Italy, 1523 and 1499	26
2.11	Two spreads: France/Portugal, 1530 and Gutenberg, C15th	26
2.12	Two spreads: Persia, 1525 and USA, 1975	27
2.13	Two spreads: USA, 1952 and England, 1087	27
2.14	Two spreads for ISO page sizes	28
2.15	Two spreads: England, 1973 and LaTeX 10pt book style	28
2.16	Two spreads: USA, 1967 and England, 1982	28
2.17	Title page design based on Adrian Wilson's <i>The Design of Books</i>	30
2.18	Two spreads: England, 1972 and Switzerland, 1980	31
2.19	Two spreads: England, 1969 and USA 1989	31
2.20	The construction of the Gutenberg page design	32
3.1	Portrait and landscape spreads	45
3.2	Landscape and portrait spreads	45
3.3	Double landscape spreads	46
4.1	Interword spacings	52
4.2	Interline spacings	53
4.3	Quotation marks: top English, bottom American	54
4.4	Raggedright text in narrow columns	56
6.1	LaTeX page layout parameters for a recto page	70
6.2	The memoir class page layout parameters for a verso page	72
6.3	The memoir class page layout parameters for a recto page	73
6.4	The recto page layout for this manual	88
6.5	Default layout for letterpaper	90
6.6	Letterpaper layout: Left \medievalpage, Right \medievalpage[12]	90

6.7	Letterpaper layout: Left \isopage, Right \isopage[12]	90
6.8	Letterpaper layout: Left \semiisopage, Right \semiisopage[12]	90
6.9	Default layout for a4paper	91
6.10	A4paper layout: Left \medievalpage, Right \medievalpage[12]	91
6.11	A4paper layout: Left \isopage, Right \isopage[12]	91
6.12	A4paper layout: Left \semiisopage, Right \semiisopage[12]	91
8.1	Example of a mandated title page style for a docotoral thesis	109
8.2	Example of a Victorian title page	110
10.1	Class layout parameters for chapter titles	129
10.2	The default chapterstyle	131
10.3	The section chapterstyle	131
10.4	The hangnum chapterstyle	132
10.5	The companion chapterstyle	133
10.6	The article chapterstyle	133
10.7	The bianchi chapterstyle	136
10.8	The bringhurst chapterstyle	137
10.9	The brotherton chapterstyle	137
10.10	The chappell chapterstyle	138
10.11	The crosshead chapterstyle	138
10.12	The culver chapterstyle	139
10.13	The dash chapterstyle	139
10.14	The demo2 chapterstyle	140
10.15	The dowding chapterstyle	140
10.16	The ell chapterstyle	141
10.17	The ger chapterstyle	141
10.18	The komalike chapterstyle	142
10.19	The lyhne chapterstyle	142
10.20	The madsen chapterstyle	143
10.21	The ntglke chapterstyle	143
10.22	The southall chapterstyle	144
10.23	The tandh chapterstyle	145
10.24	The thatcher chapterstyle	145
10.25	The veelo chapterstyle	146
10.26	The verville chapterstyle	146
10.27	The wilsondob chapterstyle	147
10.28	Displayed sectional headings	152
10.29	Run-in sectional headings	152
11.1	Header and footer slots	167
12.1	Paragraphing parameters	177
12.2	The layout parameters for general lists	184
13.1	Example extracts from toc, lof and lot files	190
13.2	Layout of a ToC (LoF, LoT) entry	191

14.1	Example framed figure	215
14.2	Example framed figure and caption	215
14.3	Example ruled figure	216
14.4	Example ruled figure and caption	216
14.5	Example float with two illustrations	216
14.6	Graphic 1 in a float	217
14.7	Graphic 2 in same float	217
14.8	Left center aligned	218
14.9	Right figure. This has more text than the adjacent caption (14.8) so the heights are unequal	218
14.10	Left top aligned	219
14.11	Right figure. This has more text than the adjacent caption (14.10) so the heights are unequal	219
14.12	Left bottom aligned	219
14.13	Right figure. This has more text than the adjacent caption (14.12) so the heights are unequal	219
14.14	Float and text page parameters	221
14.15	Float parameters	222
14.16	Long <code>\bitwonumcaption</code>	235
14.16	Lang <code>\bitwonumcaption</code>	235
14.17	Long English <code>\bionenumcaption</code>	235
	Lang Deutsch <code>\bionenumcaption</code>	235
14.18	Short English <code>\bicaption</code>	236
14.19	Figure with two subfigures	239
	(a) Subfigure 1	239
	(b) Subfigure 2	239
14.20	A picture is worth a thousand words	245
14.21	A different kind of figure caption	246
15.1	Example of a regular <code>tabular</code>	264
15.2	Example <code>tabularx</code> and <code>tabular*</code> with widths of 250pt	264
15.3	Example <code>tabularx</code> and <code>tabular*</code> with widths of 300pt	264
15.4	Changing the width of a row ordered table	271
15.5	Changing the width of a column ordered table	272
16.1	Footnote layout parameters	277
21.1	Raw indexing: (left) index commands in the source text; (right) <code>idx</code> file entries	345
21.2	Processed index: (left) alphabeticized <code>ind</code> file; (right) typeset index	346
21.3	Example endnote listing	357

List of Tables

1	Traditional font size designations	xxxiv
2	Printers units	xxxiv
1.1	Front matter	4
1.2	Common signatures	7
1.3	Some American paper sizes	7
1.4	Some traditional British book paper sizes	8
1.5	Metric book paper sizes	8
1.6	Common American commercial paper sizes	8
2.1	Some page designs	19
2.2	Average characters per line	33
3.1	Some relative type sizes for elements of books	49
5.1	Class stock metric paper size options, and commands	63
5.2	Class stock US paper size options, and commands	64
5.3	Class stock British paper size options, and commands	64
6.1	Arguments and results for <code>\settrimmedsize</code> and <code>\settypeblocksize</code> .	74
6.2	Lowercase alphabet lengths in points	76
6.3	Arguments and results for <code>\setlrmargins</code>	77
6.4	Arguments and results for <code>\setlrmarginsandblock</code>	78
6.5	Arguments and results for <code>\setulmargins</code>	79
6.6	Arguments and results for <code>\setulmarginsandblock</code>	80
6.7	Arguments and results for <code>\setheaderspaces</code>	81
6.8	The class and LaTeX page layout parameters	83
6.9	Results from sample <code>\textheight</code> adjustments	84
7.1	Font categorisation and commands	94
7.2	Font declarations	94
7.3	Standard font size declarations	97
7.4	Standard font sizes	97
7.5	The memoir class font size declarations	98
7.6	The memoir class font sizes	98
10.1	Division levels	124

10.1	Default display sectioning layout parameter values	153
10.2	Default runin sectioning layout parameter values	153
10.3	Values for S in section styling macro names.	153
10.4	Default fonts for sectional headings	160
10.5	Fonts used by different headstyles	161
11.1	The use of \thispagestyle	165
11.2	Mark macros for page headers	166
13.1	Indents and Numwidths	192
13.2	Values for X in macros for styling the titles of ‘List of...’	195
13.3	Value of K in macros for styling entries in a ‘List of...’	197
14.1	Float placement parameters	223
14.2	Float spacing parameters	223
14.3	Redesigned table caption style	228
14.4	A multi-part table	230
14.5	Another table	230
	Legendary table (toc 1)	231
	Legendary table (toc 2)	231
14.6	Permitted arguments for some sidecaption related commands	242
15.1	The array and tabular preamble options.	250
15.2	Demonstrating the parts of a table	257
15.3	Two views of one table	258
15.4	Micawber’s law	259
15.5	A narrow table split half and half	259
15.6	Example table with fills	262
15.7	Example automatic row ordered table	270
16.1	Some footnote text styles	279
21.1	MakeIndex configuration file input parameters	345
21.2	MakeIndex configuration file output parameters	348
22.1	Defined words and phrases	378
B.1	Some internal macros for numbers	406

List of typeset examples

7.1	Badly mixed fonts	95
7.2	Sometimes mixed fonts work	96
7.3	Emphasis upon emphasis	96
8.1	Example <code>\maketitle</code> title	111
10.1	A variety of subhead styles	155
12.1	Changing space before and after lists	186
15.1	Tabular with narrow and wide headings	253
18.1	Phantom text in verse	298
18.2	Verse with regular quote marks	299
18.3	Verse with hanging left quote marks	299
20.1	Named references should be to titled elements	335
22.1	TeX's minimum number in words (English style)	372
22.2	TeX's maximum number in words (American style)	373
22.3	Varieties of fractions in text	374
22.4	Super- and subscripts in text	374

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 facilities 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 seventh 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 indistinguishable 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.

memoir 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 [Wil07b]. 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 [Wil07c] which runs to about 500 pages, and from time to time an Addendum [Wil07d] is released noting extensions to the class. 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. There is also the `memexsupp` package by Lars Madsen [Mad07] which provides some extra facilities for the class.

The first part of this Manual discusses some aspects of book design and typography in general, something that I haven't come across in the usual LaTeX books and manuals. This is intended to provide a little background for when you design your own printed documents.

The second, and by far the longer part, describes the capabilities of memoir and how to use them. 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.

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

Code from the array, dcolumn, delarray and tabularx packages is integrated within the class. To improve the appearance of rules in tabular material the booktabs package is also included.

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}{1}{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}{1}{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 `\sidenote` 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 wish to use `hyperref` with memoir then you must use the `memhfixc` package¹ after using `hyperref`. For example like:

```
\documentclass[...]{memoir}
...
\usepackage[...]{hyperref}
\usepackage{memhfixc}
...
\begin{document}
```

However, if you have a version of `hyperref` dated 2006/11/15 or after, `hyperref` will automatically call in `memhfixc` so that you don't have to do anything.

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`, `array`, `booktabs`, `ccaption`, `chngcntr`, `chngpage`, `dcolumn`, `delarray`, `enumerate`, `epigraph`, `framed`, `ifmtarg`, `ifpdf`, `index`, `makeidx`, `moreverb`, `needspace`, `newfile`, `nextpage`, `parskip`, `patchcmd`, `setspace`, `shortvrb`, `showidx`, `tabularx`, `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}
```

¹`memhfixc` is supplied as part of the memoir distribution.

```
\usepackage{titling}
```

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 occurred 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.

RESOURCES

Scattered throughout, but mainly in Part I, are comments about aspects of book design and typography, in some cases accompanied by examples of better and poorer practice. If you want more authoritative remarks there are several books on the subject listed in the Bibliography; I prefer Bringhurst's *The Elements of Typographic Style* [Bri99].

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 the `comp.text.tex` newsgroup. If you are a newcomer to CTT please read the FAQ [FAQ] before asking a question, and also read a few day's worth of messages to check that your question hasn't just been answered.

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 $\langle arg \rangle$

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 Caldenty, Ezequiel Martín Cámara, David Carlisle, Gustafo Cevolani, Jean-Côme Charpentier, Michael A. Cleverly, Steven Douglas Cochran, Frederic Connes, Žarko F. Čučej, Christopher Culver, Michael W. Daniels, Michael Downes, Christopher Dutchyn, Thomas Dye, Victor Eijkhout, Danie Els, Robin Fairbairns, Simon Fear, Kai von Fintel, Ivars Finvers, Ulrike Fischer, Matthew Ford, Musa Furber, Daniel Richard G, Ignacio Fernández Galván, Gerardo Garcia, Romano Giannetti, Donald Goodman, Gabriel Guernik, Matthias Haldiman, Kathryn Hargreaves, Sven Hartrumpf, hazydirk, Carsten Heinz, Florence Henry, Peter Heslin, Morton Høgholm, Henrik Holm, Vladimir Ivanovich, Martin Jørgensen, Stefan Kahrs, 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, 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, Aaron Rendahl, René, Alan Ristow, Robert, Chris Rowley, 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 omission.² Along those lines, if you have any questions please direct

²I am currently occasionally reachable via email at `herries dot press (at) earthlink dot net`.

them to the `comp.text.tex` newsgroup instead of directly to me as you are much more 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.

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.

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 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 × 20, and

Table 1: Traditional font size designations

Points	Name	Points	Name
3	Excelsior	11	Small Pica
3½	Brilliant	12	Pica
4	Diamond	14	English
5	Pearl	18	Great Primer
5½	Agate	24	Double (or Two Line) Pica
6	Nonpareil	28	Double (or Two Line) English
6½	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

14/16 × 22 describes a 14pt font with 16pt leading set on 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 PostScript³, in order to make calculations

³PostScript is a registered trademark of Adobe Systems Incorporated.

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.

Part I

Art and Theory

One

The Parts of a Book

This chapter describes the various parts of a book, the ordering of the parts, and the typical page numbering scheme used in books.

1.1 FRONT MATTER

There are three major divisions in a book: the front matter or preliminaries, the main matter or text, and the back matter or references. The main difference as far as appearance goes is that in the front matter the folios are expressed as roman numerals and sectional divisions are not numbered. The folios are expressed as arabic numerals in the main matter and back matter. Sectional divisions are numbered in the main matter but not in the back matter.

The front matter consists of such elements as the title of the book, a table of contents, and similar items. All pages are paginated — that is they are counted — but the first few pages in the front matter, the title pages and such, do not usually have folios. The remainder of the pages in the front matter do have folios which are usually expressed as roman numerals. Not all books have all the elements described below.

The first page is a recto *half-title*, or *bastard title*, page with no folio. The page is very simple and displays just the main title of the book — no subtitle, author, or other information. One purported purpose of this page is to protect the main title page.

The first verso page, the back of the half-title page, may contain the series title, if the book is one in a series, a list of contributors, a frontispiece, or may be blank. The series title may instead be put on the half-title page or on the copyright page.

The *title page* is recto and contains the full title of the work, the names of the author(s) or editor(s), and often at the bottom of the page the name of the publisher, together with the publisher's logo if it has one.

The title page(s) may be laid out in a simple manner or can have various fol-de-rols, depending on the impression the designer wants to give. In any event the style of this page should give an indication of the style used in the main body of the work.

The verso of the title page is the copyright page. This contains the copyright notice, the publishing/printing history, the country where printed, ISBN and/or CIP information. The page is usually typeset in a smaller font than the normal text.

Following the copyright page may come a dedication or an epigraph, on a recto page, with the following verso page blank.

This essentially completes the unfolioed pages.

The headings and textual forms for the paginated pages should be the same as those for the main matter, except that headings are usually unnumbered.

The first folioed page, usually with roman numerals (e.g., this is folio iii), is recto with the Table of Contents (ToC). If the book contains figures (illustrations) and/or tables, the

Table 1.1: Front matter

Element	Page	Folio	Leaf
Half-title page	recto	no	1
Frontispiece, etc., or blank	verso	no	1
Title page	recto	no	2
Copyright page	verso	no	2
Dedication	recto	no	3
Blank	verso	no	3
Table of Contents	recto	yes	3 or 4
List of Figures	recto or verso	yes	3 or 4
List of Tables	recto or verso	yes	etc.
Foreword	recto or verso	yes	etc.
Preface	recto or verso	yes	etc.
Acknowledgements	recto or verso	yes	etc.
Introduction	recto or verso	yes	etc.
Abbreviations, etc	recto or verso	yes	etc.

List of Figures (LoF) and/or List of Tables (LoT) come after the ToC, with no blank pages separating them. The ToC should contain an entry for each following major element. If there is a LoT, say, this should be listed in the ToC. The main chapters must be listed, of course, and so should elements like a preface, bibliography or an index.

There may be a foreword after the listings, with no blank separator. A foreword is usually written by someone other than the author, preferably an eminent person whose name will help increase the sales potential, and is signed by the writer. The writer's signature is often typeset in small caps after the end of the piece.

A preface is normally written by the author, in which he includes reasons why he wrote the work in the first place, and perhaps to provide some more personal comments than would be justified in the body. A preface starts on the page immediately following a foreword, or the lists.

If any acknowledgements are required that have not already appeared in the preface, these may come next in sequence.

Following may be an introduction if this is not part of the main text. The last elements in the front material may be a list of abbreviations, list of symbols, a chronology of events, a family tree, or other information of a like sort depending on the particular work.

Table 1.1 summarises the potential elements in the front matter.

Note that the titles Foreword, Preface and Introduction are somewhat interchangeable. In some books the title Introduction may be used for what is described here as the preface, and similar changes may be made among the other terms and titles in other books.

1.1.1 Copyright page

Most people are familiar with titles, ToC, prefaces, etc., but like me are probably less familiar with the contents of the copyright page. In any event this is usually laid out by the publishing house, but some authors may like to be, or are forced into being, their own publisher.

The main point of the copyright page is to display the copyright notice. The Berne Convention does not require that published works carry a copyright notice in order to

secure copyright protection but most play it on the safe side and include a copyright notice. This usually comes in three parts: the word *Copyright* or more usually the symbol ©, the year of publication, and the name of the copyright owner. The copyright symbol matches the requirements of the Universal Copyright Convention to which the USA, the majority of European and many Asian countries belong. The phrase ‘All rights reserved’ is often added to ensure protection under the Buenos Aires Convention, to which most of the Americas belong. A typical copyright notice may look like:

© 2035 by Frederick Jones. All rights reserved.

Somewhere on the page, but often near the copyright notice, is the name and location(s) of the publisher.

Also on the copyright page is the publishing history, denoting the edition or editions¹ and their dates, and often where the book has been printed. One thing that has puzzled me in the past is the mysterious row of numbers you often see, looking like:

02 01 00 99 98 97 10 9 8 7 6 5

The set on the left, reading from right to left, are the last two digits of years starting with the original year of publication. The set on the right, and again reading from right to left, represents the potential number of new impressions (print runs). The lowest number in each group indicates the edition date and the current impression. So, the example indicates the fifth impression of a book first published in 1997.

In the USA, the page often includes the Library of Congress Cataloging-in-Publication (CIP) data, which has to be obtained from the Library of Congress. This provides some keywords about the book.

The copyright page is also the place for the ISBN (International Standard Book Number) number. This uniquely identifies the book. For example: ISBN 0-NNN-NNNNN-2. The initial 0 means that the book was published in an English-speaking country, the next group of digits identify the publisher, the third group identifies the particular book by the publisher, and the final digit, 2 in the example, is a check digit.

It is left as an exercise for the reader to garner more information about obtaining CIP and ISBN data.

1.2 MAIN MATTER

The main matter forms the heart of the book.

Just as in all the other parts of a book the pages within the main matter are included in the pagination, even though some folios may not be expressed. The folios are normally presented as arabic numerals, with the numbering starting at 1 on the first recto page of the main matter.

The main matter is at least divided into *chapters*, unless it is something like a young child’s book which consists of a single short story. When the material may be logically divided into sections larger than chapters, the chapters may be grouped into *parts* which would then be the highest level of division within the book. Frederic Connes has told me that in French typography there is often a division above the part level. This is also sometimes the case with English typography where it is typically called a *book* — the *Chicago Manual of Style* [Chi93, p. 21] shows an example. A single physical book may thus be divided into levels from *book* through *part* and *chapter* to further refinements. Typically all

¹A second edition should be more valuable than a first edition as there are many fewer of them.

of books, parts and chapters are numbered. Obviously, part numbering should be continuous throughout the book, but even with parts the chapter numbering is also continuous throughout the book.

The title of a part is usually on a recto page which just contains the part title, and number if there is one. Book titles are usually treated the same way. Chapters also start on recto pages but in this case the text of the chapter starts on the same page as the chapter title.

Where chapters are long, or when the material is complicated, they may be divided into sections, each introduced by a subhead, either numbered or unnumbered, with the numbering scheme starting afresh within each chapter. Similarly sections may be partitioned into subsections by inserting sub-subheads, but except for more technical works this is usually as fine as the subdivisions need go to. Normally there are no required page breaks before the start of any subhead within a chapter.

The title page of a part or chapter need not have the folio expressed, nor a possibly textless verso page before the start of a chapter, but all other pages should display their folios.

There may be a final chapter in the main matter called Conclusions, or similar, which may be a lengthy summary of the work presented, untouched areas, ideas for future work, and so on.

If there are any numbered appendices they logically come at the end of the main matter. Appendices are often 'numbered' alphabetically rather than numerically, so the first might be Appendix A, the second Appendix B, and so on.

An epilogue or an afterword is a relatively short piece that the author may include. These are not normally treated as prominently as the preceding chapters, and may well be put into the back matter if they are unnumbered.

1.3 BACK MATTER

The back matter is optional but if present conveys information ancillary to that in the main matter. The elements are not normally numbered, so an unnumbered appendix would normally come in the back matter.

Other elements include Notes, a Glossary and/or lists of symbols or abbreviations, which could be in the front matter instead. These elements are normally unnumbered, as is any list of contributors, Bibliography or Index.

In some instances appendices and notes may be given at the end of each chapter instead of being lumped at the back.

The first element in the back matter starts on a recto page but the remainder may start on either recto or verso pages.

In older books it was often the custom to have a colophon as the final element in a book. This is an inscription which includes information about the production and design of the book and nearly always indicates which fonts were used.

1.4 SIGNATURES AND CASTING OFF

Professionally printed books have many pages printed per sheet of (large) paper, which is then folded and cut where necessary to produce a *gathering* or *signature* of several smaller sheets. An unfolded sheet is called a *broadside*. Folding a sheet in half produces a one

Table 1.2: Common signatures

Name	Folds	Size	Sheets	Leaves	Pages
Broadside	0	$a \times b$	1	1	2
Folio	1	$b/2 \times a$	1	2	4
Quarto, <i>4to</i>	2	$a/2 \times b/2$	2	4	8
Octavo, <i>8vo</i>	3	$b/4 \times a/2$	4	8	16
<i>16mo</i>	4	$a/4 \times b/4$	8	16	32
<i>32mo</i>	5	$b/8 \times a/4$	16	32	64
<i>64mo</i>	6	$a/8 \times b/8$	32	64	128

Table 1.3: Some American paper sizes (in inches)

'Dollar bill'	7×3	Used for origami, not bills
Statement	8.5×5.5	
Executive	10.5×7.25	
Letter	11×8.5	Also in double, half or quarter size
Old (untrimmed)	12×9	Also called Architectural-A
Legal	14×8.5	
Ledger	17×11	Also called Tabloid
Broadsheet	22×17	As used in newsprint

sheet *folio* signature with two leaves and four pages. Folding it in half again and cutting along the original fold gives a two sheet *quarto* signature with four leaves and eight pages. Folding in half again, results in a four sheet *octavo* signature with eight leaves and 16 pages, and so on as listed in Table 1.2.

In Table 1.2 the Size column is the untrimmed size of a leaf in the signature with respect to the size of the broadside. When made up into a book the leaves will be trimmed to a slightly smaller size, at the discretion of the designer and publisher; typically a minimum of $1/8$ inch or 3 millimetres would be cut from the top, bottom and fore-edge of a leaf.

Other folds can produce other signatures. For example a *sexto*, obtained by folding in thirds and then folding in half, is a three sheet signature with six leaves and 12 pages.

Paper has always been made in a wide range of sizes for a myriad of uses. Table 1.3 lists some common American paper sizes.

Traditionally the sizes are denoted by name but manufacturers did not necessarily make paper of the size that matched the name they gave it. Some common names and trimmed sizes for British book work are given in Table 1.4.

The metric sizes, given in Table 1.5, are those now recommended for book production where the metric system holds sway, which includes the UK [McL80, p. 104].

In making up the book, the pages in each signature are first fastened together, usually by sewing through the folds. The signatures are then bound together and the covers, end papers and spine are attached to form the completed whole.

Commercial printers use paper larger than shown in the previous tables; they print several (final) pages on a single sheet, then fold it and trim it down to the finished page size. Table 1.6 is from [Wil93, p. 59]. He also says that other common trimmed sizes are 9.25×6.125 in out of 50×38 in sheets, 10.25×8.25 in out of 45×35 in sheets, and so on.

Table 1.4: Some traditional British book paper sizes (in inches)

Name	Quarto	Octavo
pott	8×6.5	$6.25 \times 4\text{in}$
foolscap	8.5×6.75	6.75×4.25
crown	10×7.5	7.5×5
post	10×8	8×5
large crown	10.5×8	8×5.25
large post	10.25×8.25	8.25×5.25
small demy	11.25×8.5	8.5×5.675
demy	11.25×8.75	8.75×5.675
medium	11.5×9	9×5.75
small royal	12.25×9.25	9.25×6.175
royal	12.5×10	10×6.25
super royal	13.5×10.25	10.25×6.75
imperial	15×11	11×7.5

Table 1.5: Metric book paper sizes (in mm)

	untrimmed	trimmed
metric crown octavo	192×126	186×123
metric large crown octavo	205×132	198×129
metric demy octavo	222×141	216×138
metric small royal octavo	240×158	234×156
A5		210×148

Table 1.6: Common American commercial paper sizes (in inches)

Sheet size	Book trim size	Common use	Pages per sheet (max)
45×35	8.5×5.5	scholarly works	32 pages
50×38	9.25×6.125	major nonfiction	32 pages
66×44	8×5.375	fiction & minor fiction	64 pages
68×45	8.25×5.5	major fiction & nonfiction	64 pages
45×35	11×8.5	children's books, manuals	16 pages
50×38	12.125×9.25	art monographs, children's books	16 pages

Publishers like the final typeset book to be of a length that just fits within an integral number of signatures, with few if any blank pages required to make up the final signature. Casting off is the process of determining how many lines a given text will make in a given size of type, and hence how many pages will be required.

To cast off you need to know how many characters there will be in a line, and how many characters there are, or will be, in the text. For the purposes of casting off, 'characters' includes punctuation as well as letters and digits. The first number can be easily obtained, either from copy fitting tables or by measurement; this is described in more detail in §2.3. The second is more problematic, especially when the manuscript has yet to be written. A useful rule of thumb is that words in an English text average five letters plus one space (i.e., six characters); word length in technical texts might be greater than this.

To determine the number of words it is probably easiest to type a representative portion of the manuscript, hand count the words and then divide that result by the proportion of the complete text that you have typed. For example, if you have typed 1/20th of the whole, then divide by 1/20, which is equivalent to multiplying by 20. To fully estimate the number of pages required it is also necessary to make allowance for chapter titles, illustrations, and so forth.

If it turns out, say, that your work will require 3 signatures plus 2 pages then it will be more convenient to make it fit into 3 signatures, or 4 signatures minus a page or two. This can be done by expanding or cutting the text and/or by changing the font and/or by changing the number or width of lines on a page.

When I was editing a technical journal the authors were given a word limit. The primary reason was not that we were interested in the actual word count but rather so that we could estimate, and possibly limit, the number of pages allotted to each article; we used *octavo* signatures and no blank pages. I suspect that it is the same with most publishers — it is the page count not the word count that is important to them.

In some special cases, extra pages may be 'tipped in' to the body of the book. This is most likely to occur for illustrations which require special paper for printing and it would be too costly to use that paper for the whole work. Another example is for a fold-out of some sort, a large map, say, or a triple spread illustration. The tipped in pages are glued into place in the book and may or may not be paginated. For tipped in illustrations, a List of Illustrations may well start with a phrase like: 'Between pages 52 and 53'.

1.5 PAPER

Paper, on which I assume your work will be printed, can be thought of in seven categories, six of which are used in the making of books. The categories are:

Special is not used for books. It includes 'wet strength tissues' and other sanitary, cosmetic and industrial papers.

Wrapping papers are for protective purposes. Of these kraft paper is made from unbleached chemical wood fibre sized with resin. The fibres are long and strong, hence the name 'kraft' from the German word for 'strength'. The usual colour is brown. Kraft paper is used in bookbinding for reinforcing endpapers and, strengthening and shaping spines.

Printing paper covers a wide range, from economical to expensive, in surface finish from rough to highly polished (for fine art four colour printing), and in colour.

Writing paper is suitable for all stationery requirements. Ledger paper is made from rag fibre, or a mixture of rag and wood pulp, and is strong, opaque and durable, with a smooth surface. It is used for visitor's and account books and registers, and for fine printing. Bank and bond papers are of good quality, strong, durable and nearly Ph neutral; they are made from fibres of chemical wood sized with resin. In books they are mainly used for strengthening damaged signatures. Artists' and designers' drawing papers usually have a rough surface — cartridge paper, made from well sized chemical wood fibres, is often used for tipped on endpapers.

Decorative papers used for the endpapers and sides of books are of an extensive variety of colours, textures, patterns, and quality. Any decorative paper used in a book should be strong with a good firm surface.

Ingres and similar papers are mould-made from linen and/or cotton with a little wood pulp. They come from Europe in a variety of quiet colours and are used in fine bindings for sides and endpapers.

Japanese papers and tissues are mould-made from good quality rag fibre. They are fine but strong and are extensively used for repairing documents, mending leaves, and replacing damaged or missing areas. I find Kozo paper very useful for repairing documents and, for example, as hinges when bookbinding. Although not paper, there are some wonderful Japanese bookcloths for binding covers.

Machine-made paper, which is the commonest, comes in a number of sometimes overlapping categories, of which the main ones are:

Antique papers are soft textured papers originally made for letterpress printing, but there are now surface sized ones for offset lithography.

Machine finish papers have varying degrees of surface smoothness. They are also known as super-calendered or English.

Coated paper has been flooded with fine clay and adhesive to make them particularly good for halftones. Finishes range from dull through matte to glossy.

Impregnated papers are also known as as pigmented. They are surface sized, lightly coated and calendered and can take halftones, especially by lithography.

Text papers are textured and coloured and are useful for limited editions, book jackets and end papers. They often have a deckle edge on the two long sides.

Cover papers are heavier varieties of text and other papers and are typically used for pamphlet binding and paperback covers.

Moldmade papers are made by machine to resemble handmade papers, with deckle edges. They come in a wide range of textures, colours, and weights. The available range includes papers suitable for binding sides, endpapers, book jackets, or the text block.

Handmade paper comes as single sheets but machine made paper can be obtained in either rolls or sheets. For some letterpress printing I recently bought some Strathmore 400 Drawing Paper as 3 × 30 feet rolls at about 1/3 the price of the same quantity of paper in sheet form; the downside was that I had to slice it up into the sheet size I wanted to use, but in this case the upside was that I tore rather than cut and obtained sheets with deckle edges on all four sides so that it at the end it looked rather like handmade paper.

It is not often that books include information about the paper on which they are printed. If they do they are likely to be fine press books or limited editions, but even then most I have seen are silent on the matter. A few trade books do include details. Among the more

popular papers² I have come across are: Arches, of various kinds; Curtis Rag; Fabriano; Glatfelter; Linweave Early American, which has been used by the University of California press; Mohawk Superfine; Strathmore,³ of several kinds; and Warren's Old Style, which has been used for several books published by the University of California.

²Meaning that I know that they have been used in more than one book.

³At the time of writing I have nearly finished hand letterpress printing a small book of 35 printed pages on Strathmore 400 Drawing Paper.

Two

The page

Authors usually want their works to be read by others than themselves, and this implies that their manuscript will be reproduced in some manner. It is to be hoped that the published version of their work will attract readers and there are two aspects to this. The major is the actual content of the work — the thoughts of the author couched in an interesting manner — if something is boring, then there are too many other interesting things for the reader to do than to plow on until the bitter end, assuming that he even started to read seriously after an initial scan. The other aspect is the manner in which the content is displayed. Or, in other words, the *typography* of the book, which is the subject of this chapter.

The essence of good typography is that it is not noticeable at first, or even second or later, glances to any without a trained eye. If your initial reaction when glancing through a book is to exclaim about its layout then it is most probably badly designed, if it was designed at all. Good typography is subtle, not strident.

With the advent of desktop publishing many authors are tempted to design their own books. It is seemingly all too easy to do. Just pick a few of the thousands of fonts that are available, use this one for headings, that one for the main text, another one for captions, decide how big the typeblock is to be, and there you are.

However, just as writing is a skill that has to be learned, typography is also an art that has to be learned and practised. There are hundreds of years of experience embodied in the good design of a book. These are not to be cast aside lightly and many authors who design their own books do not know what some of the hard-earned lessons are, let alone that what they are doing may be the very antithesis of these. An expert can break the rules, but then he is aware that he has good reasons for breaking them.

The author supplies the message and the typographer supplies the medium. Contrary to Marshall McLuhan, the medium is *not* the message, and the typographer's job is not to intrude between the message and the audience, but to subtly increase the reader's enjoyment and involvement. If a book shouts 'look at me!' then it is an advertisement, and a bad one at that, for the designer.

2.1 THE SHAPE OF A BOOK

Books come in many shapes and sizes, but over the centuries certain shapes have been found to be more pleasurable and convenient than others. Thus books, except for a very very few, are rectangular in shape. The exceptions on the whole are books for young children, although I do have a book edited by Fritz Spiegl and published by Pan Books entitled *A Small Book of Grave Humour*, which is in the shape of a tombstone — this is an anthology

of epitaphs. Normally the height of a book, when closed, is greater than the width. Apart from any aesthetic reasons, a book of this shape is physically more comfortable to hold than one which is wider than it is high.

It might appear that the designer has great freedom in choosing the size of the work, but for economic reasons this is not normally the case. Much typographical design is based upon the availability of certain standard industrial sizes of sheets of paper. A page size of 12×8 inches will be much more expensive than one which fits on a standard US letter sheet of $11 \times 8\frac{1}{2}$ inches. Similarly, one of the standard sizes for a business envelope is $4\frac{1}{8} \times 9\frac{1}{2}$ inches. Brochures for mailing should be designed so that they can be inserted into the envelope with minimal folding. Thus a brochure size of 5×10 inches will be highly inconvenient, no matter how good it looks visually.

Over the years books have been produced in an almost infinite variety of proportions, where by *proportion* I mean the ratio of the height to the width of a rectangle. However, certain proportions occur time after time throughout the centuries and across many different countries and civilizations. This is because some proportions are inherently more pleasing to the eye than others are. These pleasing proportions are also commonly found in nature — in physical, biological, and chemical systems and constructs.

Some examples of pleasing proportions can be seen in Japanese wood block prints, such as the *Hoso-ye* size ($2 : 1$) which is a double square, the *Oban* ($3 : 2$), the *Chuban* ($11 : 8$) and the *Koban* size ($\sqrt{2} : 1$). Sometimes these prints were made up into books, but were often published as stand-alone art work. Similarly Indian paintings, at least in the 16th to the 18th century, often come in the range $1.701 : 1$ to $13 : 9$, thus being around $3 : 2$ in proportion.

In medieval Europe page proportions were generally in the range $1.25 : 1$ to $1.5 : 1$. Sheets of paper were typically produced in the proportion $4 : 3$ ($1.33 : 1$) or $3 : 2$ ($1.5 : 1$). All sheet proportions have the property that they are reproduced with each alternate folding of the sheet. For example, if a sheet starts at a size of 60×40 (i.e., $3 : 2$), then the first fold will make a double sheet of size 30×40 (i.e., $3 : 4$). The next fold will produce a quadrupled sheet of size 30×20 , which is again $3 : 2$, and so on. The Renaissance typographers tended to like taller books, and their proportions would go up to $1.87 : 1$ or so. The style nowadays has tended to go back towards the medieval proportions.

The standard ISO page proportions are $\sqrt{2} : 1$ ($1.414 : 1$). These have a similar folding property to the other proportions, except in this case each fold reproduces the original page proportion. Thus halving an A0 sheet (size 1189×841 mm) produces an A1 size sheet (594×841), which in turn being halved produces the A2 sheet (420×594), down through the A3, A4 (210×297 mm), A5, ... sheets.

For many years it was thought that it was impossible to fold a sheet of paper, no matter how large and thin, more than six times altogether. This is not so as in 2002 a high school student, Britney Gallivan, managed to fold a sheet of paper in half twelve times (see, for example, <http://mathworld.wolfram.com/Folding.html>).

There is no one perfect proportion for a page, although some are clearly better than others. For ordinary books both publishers and readers tend to prefer books whose proportions range from the light $9 : 5$ ($1.8 : 1$) to the heavy $5 : 4$ ($1.25 : 1$). Some examples are shown in Figure 2.1. Wider pages, those with proportions less than $\sqrt{2} : 1$ ($1.414 : 1$), are principally useful for documents that need extra width for tables, marginal notes, or where multi-column printing is preferred.

In books where the illustrations are the primary concern, the shape of the illustrations is

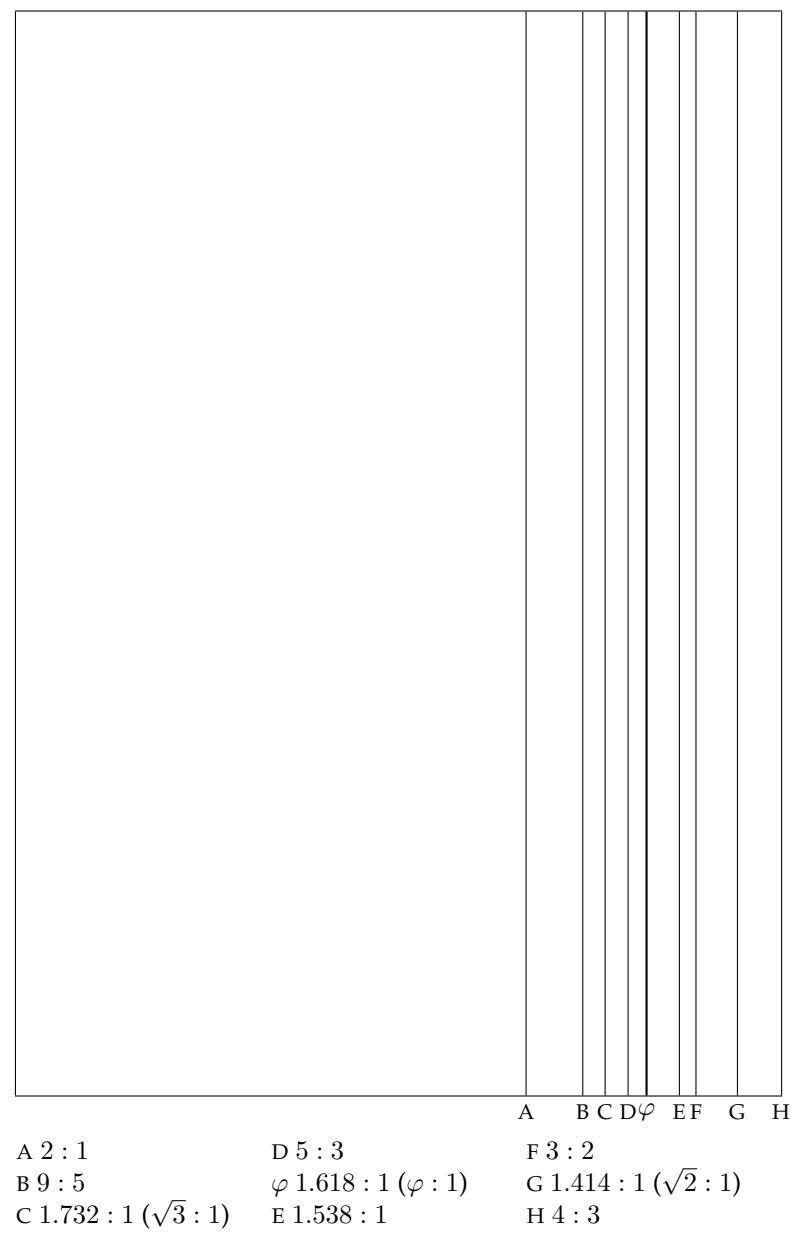


Figure 2.1: Some page proportions

generally the major influence on the page proportion. The page size should be somewhat higher than that of the average illustration. The extra height is required for the insertion of captions describing the illustration. A proportion of $\pi : e$ (1.156 : 1), which is slightly higher than a perfect square, is good for square illustrations.¹ The $e : \pi$ (0.864 : 1) proportion is useful for landscape photographs taken with a 4×5 format camera, while those from a 35mm camera (which produces a negative with a $2 : 3$ proportion) are better accommodated on an $0.83 : 1$ page.

2.1.1 The golden section and Fibonacci series

Typographers need a modicum of mathematical ability, but no more than an average teenager can do — basically simple arithmetic. You can skip this section if you wish as it just provides some background mathematical material which might be of interest.

Since ancient Greek times or even before, the golden section, which is denoted by the Greek letter φ (phi), has been considered to be a particularly harmonious proportion. It should come as no surprise, then, that this also has applications in typography.

The Greeks were interested in geometry (think of Euclid). They discovered that if you divide a straight line into two unequal parts then a certain division appeared to have an especially appealing aesthetic quality about it. Call the length of the line l and the length of the two parts a and b , where a is the smaller and b is the larger. The division in question is when the ratio of the larger to the smaller division (b/a) is the same as the ratio of the whole line to the larger division (l/b). More formally, two elements embody the golden section, symbolised by φ , when the ratio of the larger to the smaller is the same as the ratio of the sum of the two to the larger. If the two elements are a and b , with $a < b$, then

$$\varphi = \frac{b}{a} = \frac{a+b}{b} = (1 + \sqrt{5})/2 \quad (2.1)$$

The golden section has been called by a number of different names during its history. Euclid called it the ‘extreme and mean ratio’ while Renaissance writers called it the ‘divine proportion’; now it is called either the ‘golden section’ or the ‘golden ratio’. The symbol φ is said to come from the name of the Greek artist Phidias (C5th BC) who often used the golden section in his sculpture. A rectangle whose sides are in the same proportion as the golden section is often called a ‘golden rectangle’. The front of the Parthenon on the Acropolis in Athens is a golden rectangle, and such rectangles appear often in Greek architecture. The symbol of the Pythagorean school was the star pentagram, where each line is divided in the golden section.

The approximate decimal value for φ is 1.61803. The number has some unusual properties. If you add one to φ you get its square, while subtracting one from φ gives its reciprocal.

$$\varphi + 1 = \varphi^2 \quad (2.2)$$

$$\varphi - 1 = 1/\varphi \quad (2.3)$$

¹Both e and π are well known mathematical numbers. e ($= 2.718 \dots$) is the base of natural logarithms and π ($= 3.141 \dots$) is the ratio of the circumference of a circle to its diameter.

It also has a very simple definition as the continued fraction

$$\varphi = 1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \dots}}}} \quad (2.4)$$

In 1202 Leonardo Pisano, also known as Leonardo Fibonacci, wrote a book called *Liber Abbaci*.² One of the topics he was interested in was population growth. The book included this exercise:

How many pairs of rabbits can be produced from a single pair in a year? Assume that each pair produces a new pair of offspring every month, a rabbit becomes fertile at age one month, and no rabbits die during the year.

After a month there will be two pairs. At the end of the next month the first pair will have produced another pair, so now there are three pairs. At the end of the following month the original pair will have produced a third pair of offspring and their firstborn will also have produced a pair, to make five pairs in all. And so on. If, like the rabbits, you are not too exhausted to continue, you can get the following series of numbers³:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 \dots$$

After the first two terms, each term in the series is the sum of the two preceding terms. Also, as one progresses along the series, the ratio of any adjacent pair of terms oscillates around φ ($= 1.618\dots$), approaching it ever more closely.

$$\begin{aligned} 8/5 &= 1.6 \\ 13/8 &= 1.625 \\ 21/13 &= 1.615 \\ 34/21 &= 1.619 \\ 55/34 &= 1.6176 \\ 89/55 &= 1.6182 \end{aligned}$$

For the mathematically inclined there is another, to me, typographically striking relationship between φ and the Fibonacci series. Define the Fibonacci numbers as F_n , where

$$F_0 = 0; \quad F_1 = 1; \quad F_{n+2} = F_{n+1} + F_n, \quad n \geq 0. \quad (2.5)$$

Then

$$F_n = \frac{1}{\sqrt{5}}(\varphi^n - (-\varphi)^{-n}) \quad (2.6)$$

Both the Fibonacci series and the golden section appear in nature. The arrangement of seeds in a sunflower, the pattern on the surface of a pinecone, and the spacing of leaves around a stalk all exhibit Fibonacci patterns (for example see [CG96]). Martin Gardner [Gar66] reports on a study of 65 women that claimed that the average ratio of a

²Book of the Abacus.

³The numbers at the start of the series depend on whether you consider the initial pair of rabbits to be adults or babies.

person's height to the height of the navel is 1.618+ — suspiciously close to φ . According to Dan Brown, the author of *The Da Vinci Code*, Mario Livio's *The Golden Ratio* [Liv02] '... unveils the history and mystery of the remarkable number phi in such a way that ... you will never again look at a pyramid, pinecone, or Picasso in the same light'.

2.2 THE SPREAD

The typeblock is that part of the page which is normally covered with type. The same proportions that are useful for the shape of a page are also useful for the shape of the typeblock. This does not mean, though, that the proportions of the page and the typeblock should be the same. For instance, a square typeblock on a square page is inherently dull.

When we first start to learn to read we scan horizontally along each line of text. As our skills improve we tend to scan vertically rather than horizontally. A tall column of text helps in this process, provided that the column is not too wide.

A page in a book will typically contain several elements. Principal among these is the typeblock, but there are also items like the folio (that is, the page number), a running header and/or footer which carries the chapter and/or book title, and possibly marginalia and footnotes. These latter elements, although essential to the content of the book, are minor visual elements compared to the typeblock. But even minor decoration can obscure or kill an otherwise good design.

The major concern is the positioning of the typeblock on the page. The mere fact of positioning the typeblock also has the result of producing margins onto the page. Page design is a question of balancing the page proportions with the proportions of the typeblock and the proportions of the margins to create an interesting yet harmonious composition. A single page, except for a title page, is never the subject of a design but rather the design is in terms of the two pages that are on view when a book is opened — the left and right hand pages are considered as a whole. More technically, the design is in terms of a *double spread*.

Table 2.1 gives some examples of page designs. These are arranged in increasing order of fatness. In this table, and afterwards, I have just used a single number to represent the ratio of the page height to the width; that is, for example, 1.5 instead of 1.5 : 1 or 12/7 instead of 12 : 7. The following symbols are used in the table:

Proportions :

P = page proportion = h/w

T = typeblock proportion = d/m

Page size :

w = width of page

h = height of page

Typeblock :

m = measure (i.e., width) of primary typeblock

d = depth (excluding folios, running heads, etc.)

Margins :

s = spine margin (back margin)

t = top margin (head margin)

e = fore-edge (front margin)

f = foot margin (bottom margin)

g = internal gutter (on a multi-column page)

Table 2.1: Some page designs

P	T	Margins & Columns					Figure
		s	t	e	f	g	
$\sqrt{3}$	2	$w/13$	$8s/5$	$16s/5$	$16s/5$		2.2 left
$\sqrt{3}$	e/φ	$w/10$	$2s$	$2s$	$3s$		2.2 right
12/7	1.701	$w/7$	$8s/5$	$8s/5$	$14s/5$		2.3 left
e/φ	7/4	$w/10$	$5s/4$	$5s/3$	$11s/8$		2.3 right
φ	1.866	$w/9$	s	$2s$	$7s/3$		2.4 left
φ	φ	$w/12$	$2s$	$5s/2$	$4s$		2.4 right
8/5	1.634	$2w/15$	$7s/5$	$9s/5$	$13s/5$		2.5 left
19/12	7/4	$2w/15$	s	$9s/8$	$11s/8$		2.5 right
19/12	$\sqrt{3}$	$w/7$	s	$5s/4$	$1.84s$		2.6 left
19/12	8/5	$w/12$	$7s/5$	$8s/5$	$2s$		2.6 right
$\pi/2$	9/5	$w/9$	$3s/2$	$5s/2$	$3s$		2.7 left
$e/\sqrt{3}$	1.71	$w/10$	$11s/8$	$24s/11$	$8s/3$		2.7 right
1.553	1.658	$w/11$	φs	φs	φs		2.8 left
1.538	$\sqrt{7}$	$w/10$	s	$23s/6$	$3s/2$		2.8 right
3/2	2	$w/5$	$s/2$	s	s		2.10 left
3/2	1.701	$w/9$	s	$2s$	$7s/3$		2.10 right
3/2	$\pi/2$	$w/13$	$2s$	$10s/3$	$30s/7$		2.11 left
3/2	3/2	$w/9$	$3s/2$	$2s$	$3s$		2.11 right
3/2	1.68	$w/23$	$2s$	$5s$	$2s$		2.12 left
3/2	3/2	$w/10$	$2s$	$5s/2$	$2.85s$		2.12 right
1.48	1.376	$w/12$	$7s/4$	$2s$	$7s/2$		2.13 left
13/9	$\sqrt{2}$	$w/30$	$2s$	$9s/2$	$4s$	$s/2$	2.13 right
$\sqrt{2}$	φ	$w/9$	s	$2s$	$2s$		2.14 left
$\sqrt{2}$	φ	$w/8$	s	$5s/3$	$5s/3$		2.14 right
7/5	1.641	$w/7$	s	$8s/5$	$8s/5$		2.15 left
17/22	1.594	$0.176w$	$1.21s$	$1.47s$	$1.05s$		2.15 right
1.294	13/9	$w/12$	s	$2s$	$10s/7$	$s/2$	2.16 left
9/7	19/9	$2w/5$	$5s/8$	$5s/8$	$5s/6$		2.16 right
5/4	13/11	$w/10$	$3s/2$	$2s$	$8s/3$		2.18 left
7/6	55/48	$w/10$	$9s/10$	$8s/10$	$13s/10$	$1.05s$	2.18 right
e/π	0.951	$w/9$	s	$2s$	$3s/2$		2.19 left
5/7	2/3	$w/9$	$s/2$	$2s/3$	s	$s/3$	2.19 right

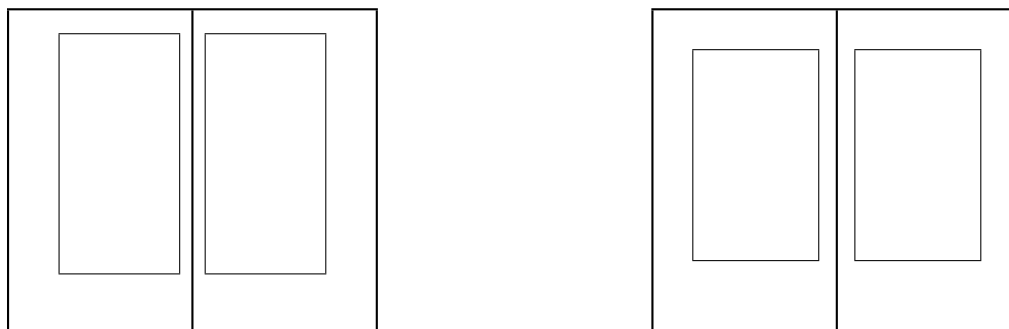


Figure 2.2: Two spreads: (Left) Canada, 1992. (Right) England, 1970.

Theoretically the following relationship holds among the several variables:

$$f + t - T(s + e) = w(P - T)$$

However, due to measurement and other difficulties, the numbers given in the table do not always obey this rule but they are close enough to give a good idea of the relative values. In any event, page design is not a simple arithmetic exercise but requires aesthetic judgement.

The designs are also shown in Figures 2.2 to 2.19. Each of these shows a double page spread; the page width has been kept constant throughout the series to enable easier visual comparison — it is the relative proportions, not the absolute size, that are important. I have only shown the pages and the typeblocks to avoid confusing the diagrams with headers, footers or folios.

Shown in Figure 2.2 are two modern books. On the left is the layout for Robert Bringhurst's *The Elements of Typographical Style* published by Hartley & Marks in 1992, and designed by Bringhurst. The text face is Minion set with 12pt leading on a 21pc measure. The captions are set in Scala Sans. The original size is 227 × 132 mm and is printed on Glatfelter laid paper. I highly recommend this book if you are interested in typography.

The layout on the right is The Folio Society's 1970 edition of *The Prince* by Niccolò Machiavelli. The original size is 216 × 125 mm and is set in 12/13 × 22 Centaur. Chapter titles are set as raggedright block paragraphs using Roman numbers and small caps for the text; not all chapters start a new page. There are no running headers and the folios are set at the center of the footer. The ToC is typeset like the standard LaTeX ToC but with the chapter titles in small caps.

Figure 2.3 (left) illustrates a small book by Wilfred T. Grenfell entitled *Adrift on an Ice-Pan* published in 1909 by the Riverside Press of Boston. The text is set with a leading of 16pt on a 16pc measure. The large leading and small measure combine to give a very open appearance. The original size is 184 × 107 mm. The half-title is set in bold uppercase about 1/3 of the way down the page. Uppercase is used for chapter headings which are centered. Captions for the photographs are also uppercase and are listed on an illustrations page. The folios are centered in the footer and enclosed in square brackets (e.g., [17]), and the headers contain the book title, centered, and in uppercase.

On the right is another book from the Folio Society — *Three Men in a Boat* by Jerome K. Jerome printed in 1964. The original size is 215 × 128 mm and is typeset with Ehrhardt

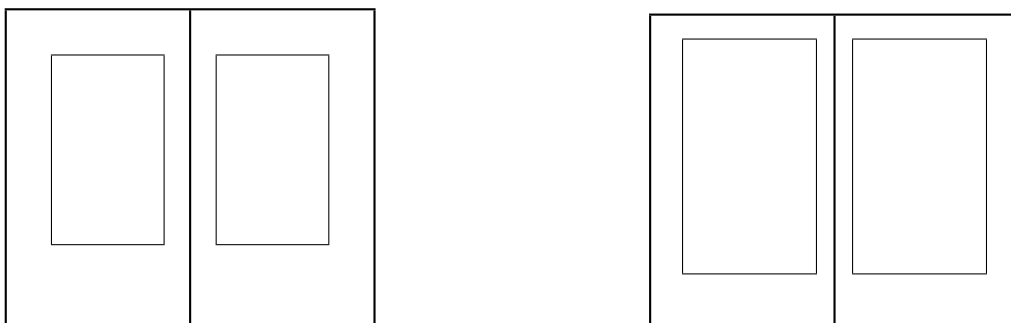


Figure 2.3: Two spreads: (Left) USA, 1909. (Right) England, 1964.

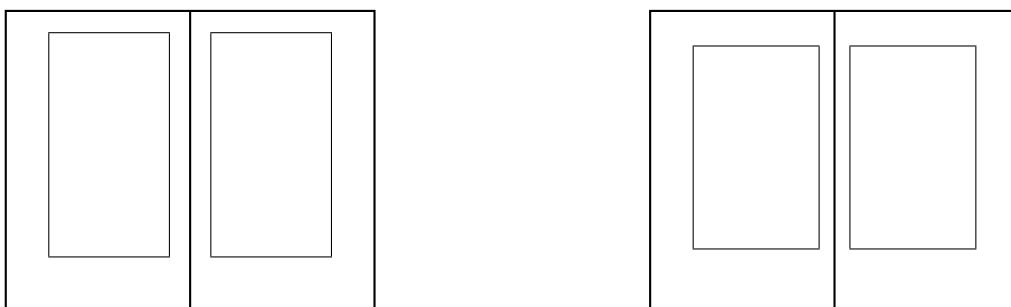


Figure 2.4: Two spreads: (Left) France, 1559. (Right) Canada, 1995.

at $11/12 \times 22$. Chapter titles are centered and simply consist of the word ‘CHAPTER’ followed by the number. There are no headers and the folio is set between square brackets (like [27]) in the center of the footer. The ToC title is centered, and the chapter entries are like standard LaTeX except that the numbers are set in a roman font while the texts, which give a summary of the chapter contents, are typeset in italic.

Jean de Tourmes, a Parisian publisher, printed *Histoire et Chronique* by Jean Froissart in 1559. This is a history book with the main text in roman and sidenotes in italic at roughly 80% of the size of the main text. The layout is shown in Figure 2.4 (left). The gutter (not shown) between the main text and the sidenote column is very small, but the change in fonts and sizes enables the book to be read with no confusion.

Another Hartley & Marks typography book — *Finer Points in the Spacing & Arrangement of Type* by Geoffrey Dowding — is shown at the right of Figure 2.4. This is typeset in Ehrhardt at $10.5/14 \times 23$ on a page size of 231×143 mm on Glatfelter Laid Offset paper. The half-title is uppcased, centered, and in the upper quarter of the page. On the title page the title is typeset with a large bold italic font while the author’s is set using normal uppcase and the publisher is set in small caps. Dowding uses ‘part’ instead of ‘chapter’. Chapter heads are centered with the number written out, like ‘PART ONE’, and below this is the title set in large italics. Section heads are in uppcase and subsection heads in small caps, both centered. Folios are in the center of the footer; verso running heads consist of the book title in small caps and centered, and recto heads contain the chapter title in italics

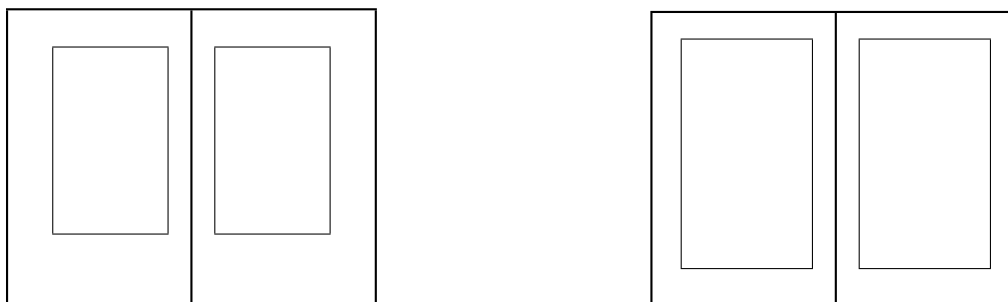


Figure 2.5: Two spreads: (Left) USA, 1949. (Right) USA, 1990.

and centered. On the contents page the part (chapter) numbers and titles are centered, using small caps and large italics respectively (and no page numbers). Section titles are in small caps, left justified with the page numbers right justified. Titles from the front matter and back matter, for example the Foreword and Bibliography, are typeset in italics.

Bruce Rogers (1870–1957) described how he came to design his Centaur typeface in *Centaur Types*, a privately published book by his studio October House in 1949. The layout of this book, which of course was typeset in Centaur, is shown at the left of Figure 2.5. Centaur is an upright seriffed type based on Nicolas Jenson’s type as used in *Eusebius* published in 1470. *Centaur Types* demonstrates typefaces other than Centaur, and also includes exact size reproductions of the engraver’s patterns. It is set at 14/16 \times 22 on a page size of 240 \times 150 mm.

Figure 2.5 (right) is the layout of another book on typefaces. It is *The Anatomy of a Typeface* by Alexander Lawson published by David R. Godine in 1990. This is set in Galliard with 13pt leading and a measure of 24pc on a page size of 227.5 \times 150 mm on Glatfelter Offset Smooth Eggshell paper. The half-title is set in uppercase in the upper quarter of the page. On the title page the title is in uppercase in a large outline font, with a double rule above and a short single rule below. The author is set in small caps (both upper- and lowercase like LAWSON) and the publisher is in regular lowercase small caps. Chapter heads are centered with the number set between a pair of fleurons, followed by the title in large uppercase, and with a short rule between the title and the start of the text. The folios are in the center of the footer with a short rule above them; there are no running headers. The contents page is set with the body type; chapter numbers are flushleft with a following period and the page numbers are flushright.

Microcosmographica Academia by F. M. Cornford is shown in Figure 2.6. Despite its title, it is written in English and was published by Bowes & Bowes, London, in 1908. It is a dryly humorous look at academic politics as practised in Cambridge University at the turn of the nineteenth century (probably in the twentieth and twenty-first as well). It is set with 14pt leading on 22pc. The original page size is 216 \times 136 mm. The half-title is in normal uppercase in the upper sixth of the page; the title page is all uppercase in various sizes. Chapter heads are centered with first the number in Roman numerals and below the title in uppercase. Folios are centered in the footer and there are no running heads. There is no table of contents.

The right of this figure illustrates a book with another unusual title — *The Alphabet*

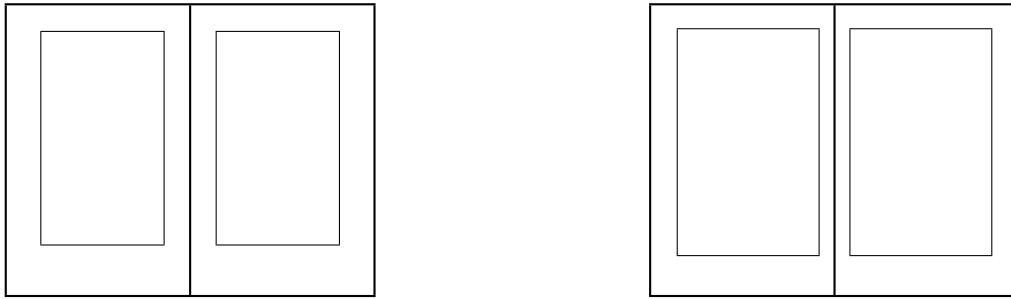


Figure 2.6: Two spreads: (Left) England, 1908. (Right) USA, 1993.

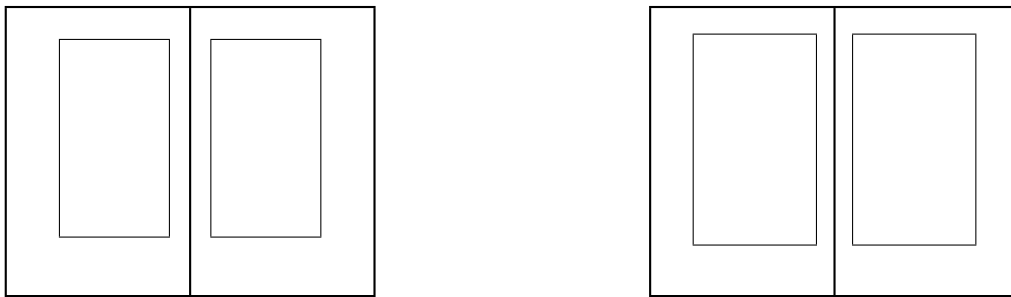


Figure 2.7: Two spreads: (Left) USA, 1931. (Right) England, 1968.

Abecedarium by Richard A. Firmage and published by David R. Godine in 1993. It is set in Adobe Garamond on a 27pc measure with 14pt leading. The original page size is 227.5×150 mm. The book gives a history of each letter of the Latin alphabet. Chapter heads are centered and consist of an ornamental version of the letter in question. One unusual feature is that there is a deep footer on each page showing many examples of typefaces of the letter being described. Verso running headers consist of the book title in mixed small caps and centered with the folio flushleft. Recto headers have the folio flushright, and centered is the alphabet, typeset in small caps except for the current letter which is enlarged.

W. A. Dwiggins was, among many other things, an American book designer. Figure 2.7 (left) shows his layout of H. G. Wells' *The Time Machine* for Random House in 1931. The page size is 231×147 mm.

The right of the figure illustrates the layout of a book called *Two Men — Walter Lewis and Stanley Morrison at Cambridge* by Brooke Crutchley and published by Cambridge University Press in 1968. This is typeset in Monotype Barbon with 17.5 leading on a 26pc measure on a 253×162 mm page. Crutchley was the Cambridge University Printer and each year would produce a limited edition of a book about Cambridge or typography, and preferably both together, for presentation to friends of the Press. The tradition of the Printer's Christmas Book was started by Stanley Morison in 1930 and continued until 1974. The books usually consisted of a short essay on a particular topic, so they did not have chapter heads, tables of contents, or other appurtenances, apart from a Preface.



Figure 2.8: Two spreads: (Left) USA, 1994. (Right) England 1988.

A modern technical book layout is given in Figure 2.8. The book is *Information Modeling the EXPRESS Way* by Douglas Schenck and Peter Wilson, published by Oxford University Press (New York) in 1994. This is set in Computer Modern Roman at $10/12 \times 27$ on a page 233×150 mm. It has the typical LaTeX appearance with perhaps the exception of the epigraphs after each chapter heading.

Ruari McLean's *The Thames and Hudson Manual of Typography* (1988) is at the right in Figure 2.8. This is typeset in $10/11 \times 20$ Monophoto Garamond on a 240×156 mm page. The wide fore-edge is used for small illustrations. Notes are also set in this margin rather than at the foot of the page. The half-title is in a bold font, flushright, in the upper quarter of the page; there is a wood engraving of a galleon at the bottom, also flushright. The title uses a mixture of fonts and is set flushright; an example title page based on this design is shown in Figure 2.9. Chapter are on recto pages and consist of the number and title in a bold font, flushleft and near the top of the page, and an engraving of some kind is at the bottom right of the page; there is no other text on this page, the body of the chapter starting at the top of the following verso page. Folios are in the footers at the outer edge of the page. Running headers contain the chapter title in small caps flushright in the outer margin.

Many page layouts in earlier days were constructed by drawing with compass and ruler, usually based on regular geometric figures; the use of squares, pentagons and hexagons being particularly prevelant. Unusually, the typeblock in Figure 2.10 (left) is centered on the page. The typeblock is based on a square, the depth being twice the measure. The book, *Canzone* by Giangiorgio Trissino, is a volume of poems and was published in Rome about 1523 by Ludovico degli Arrighi. Prose works from the same typographer followed the normal style of having the fore-edge wider than the spine margin.

The page proportion in Figure 2.10 (right) is also a simple $3 : 2$ ratio. The proportions of the typeblock, being $1.7 : 1$, are based upon a pentagon. The book is *Hypnerotomachia Poliphili* by Francesco Colonna and was published by Aldus Manutius in Venice in 1499. The story of this, including some reproductions from the original, is told by Helen Barolini [Bar92].

In 1519 the Portugese explorer Ferdinand Magellan set sail from Sanlúcar de Barrameda, near Cádiz in Spain, with five ships and about 270 men. Three years later one ship and 18 men returned, having made the first circumnavigation. Among the few survivors was Antonio Pigafetta who recorded the adventure. A very few manuscripts of his report are in existence. The layout of one of these manuscripts which is in the Beinecke Rare

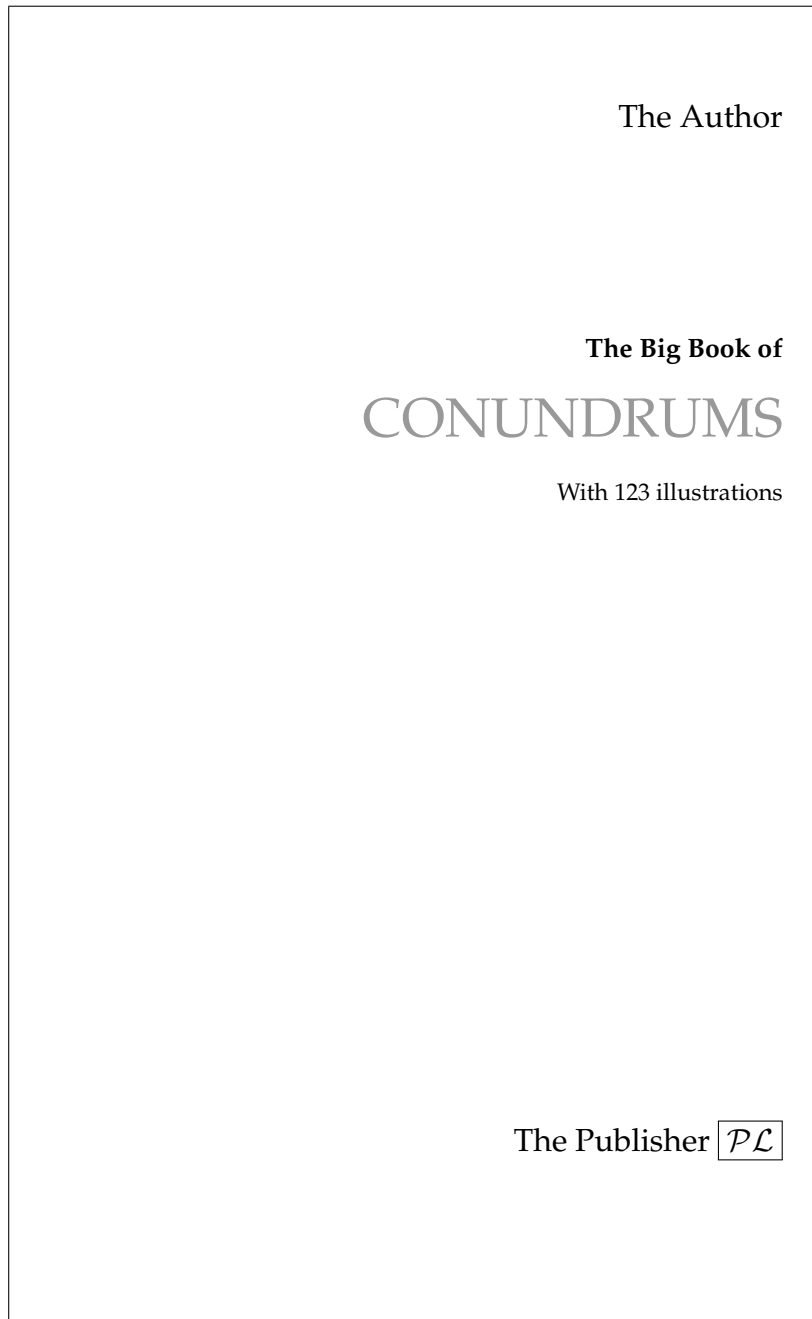


Figure 2.9: Title page design based on *The Thames and Hudson Manual of Typography* (1988)



Figure 2.10: Two spreads: (Left) Italy, 1523. (Right) Italy 1499.

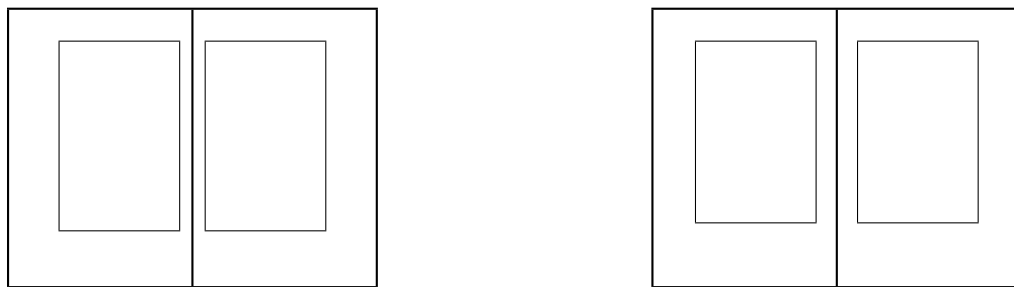


Figure 2.11: Two spreads: (Left) France/Portugal, 1530. (Right) Gutenberg, C15th.

Book and Manuscript Library at Yale is shown at the left of Figure 2.11. The manuscript, which is written in French, is called *Navigation et descouurement de la Inde superieure et isles de Malueque ou naissent les cloux de Girofle* (Navigation and discovery of Upper India and the Isles of Molucca where the cloves grow) is written in a beautiful humanistic minuscule. There are 27 lines to a page, which is 286×190 mm and made of vellum. The text measure is 29.5 and the 'leading' is 21pt. The wide outer (fore-edge) margin is used for sidenotes indicating highlights of the story. The manuscript was probably prepared soon before 1530; the scribe and where he worked is unknown.

Many of the books produced by Johannes Gutenberg (1398–1468) and his early successors followed the form shown in Figure 2.11 (right). This set of proportions was also often used in medieval incunabula⁴ and manuscripts. The page and typeblock proportions are the same (3 : 2). The margins are in the proportions 2 : 3 : 4 : 6. A graphical method for constructing this, and similar designs, is shown later in Figure 2.20.

Two versions of the same publication are shown in Figure 2.12. On the left is a Persian manuscript *Khamsch of Nizami* written about 1525. The page size is about 324×216 mm. The illustrations and the typeblock are inextricably mixed. On the right is a translation of some of the manuscript published as *Tales from the Khamsch of Nizami* by the Metropolitan Museum of Art, New York, in 1975. The modern version has a page size of 300×200 mm, slightly smaller than the original but in the same proportions. The typeblock is 32pc wide and the type is set with a 15pt leading.

⁴Early books, especially those printed before 1500.



Figure 2.12: Two spreads: (Left) Persia, 1525. (Right) USA, 1975.



Figure 2.13: Two spreads: (Left) USA, 1952. (Right) England, 1087.

Frederic Goudy was a prolific American type designer. Shown at the left of Figure 2.13 is the layout of his book *The Alphabet and Elements of Lettering* published by the University of California Press in 1952. This is typeset in his University of California Old Style, which has interesting ct and st ligatures. The measure is 36pc and the leading is 18pt. The first half of the book gives a short history of the development of writing and fonts. The second half consists of 27 plates, one for each letter of the alphabet, and the last one for the ampersand character. These show the evolution of each letter from Roman times to the mid-twentieth century.

Figure 2.13 (right) shows the layout of the English *Domesday Book* which is a manuscript book written in 1087. It records all the domains won by William the Conqueror in 1066. The book is written in a Carolingian minuscule in two columns, with 44 lines per column ragged right. The two columns have slightly different widths. The first part of the book is more meticulously written than the later parts, where the scribe appears to be in haste to finish.

Figure 2.14 shows two different layouts for a page corresponding to the ISO international standard proportion of $\sqrt{2}$. In each case the typeblock is the same and proportioned in the golden section, but the margins are different. The layout on the left provides adequate room for marginal notes in the fore-edge.

Another of the Cambridge University Printer's Christmas books is at the left of Figure 2.15. In this case it is *Emery Walker — Some Light on his Theories of Printing and on his Relations with William Morris and Cobden-Sanderson* by Colin Franklin and published in 1973. The page size is 295 × 210 mm with a measure of 31pc set with 15pt leading. Unusu-



Figure 2.14: Two spreads: (Left) ISO (1). (Right) ISO (2).



Figure 2.15: Two spreads: (Left) England, 1973. (Right) LaTeX 10pt book style.



Figure 2.16: Two spreads: (Left) USA, 1967. (Right) England, 1982.

ally for this series it has chapter heads which are simply the number centered above the title in a large font. It also has illustrations which are listed on an Illustrations page where the caption titles are set flushleft and page numbers flushright. The page is divided into two lists. The first has a heading (centered) in italics of '*In text*' with '*page*' flushright above the page numbers. The second has the centered heading '*In pocket at end*' and there are no page numbers in this list as the corresponding illustrations are not bound into the book, instead they are loosely inserted in a pocket at the end of the book.

On the right is the default layout provided by the LaTeX 10pt book class on US letterpaper.

Adrian Wilson, who died in 1988, was an acclaimed American book designer. His work on book design, *The Design of Books*, out of print since 1988 but reissued in 1993 by Chronicle Books, is outlined at the left of Figure 2.16. This is in two columns, with

many illustrations, on letterpaper size pages. It is typeset in Palatino and Linotype Aldus with 12pt leading. Each column is 18pc wide. The title page is a simple design and an example based on it is shown in Figure 2.17. Chapter heads are flushright in a large italic, preceded by the number. Section heads are flushleft in uppercase and subsection heads are also flushleft but in normal sized italics. The Contents list is in the left hand column, typeset using the normal font with titles flushleft and page numbers flushright; there is an engraving in the bottom half of the right hand column. There are no running headers. Verso footers have the chapter title flushleft in small caps with the folio to the left of this (i.e., in the margin); recto footers similarly have the chapter title flushright and the folio to the right in the margin.

The other layout in this figure is B. W. Robinson's *Kuniyoshi: The Warrior Prints* published by Phaidon, Oxford in 1982. The page size is 310×242 mm with a measure of 28.5pc. The type is set with 13pt leading. The wide spine margin is used for some small reproductions of Japanese woodblock prints, some of which extend across the binding itself. The majority of the book has no text apart from captioning the many reproduced prints which take up full pages.

The Waterways of the Fens by Peter Eden with drawings by Warwick Hutton is another of the Cambridge Printer's Christmas books. This is set with 17pt leading on a measure of 27pc. The original page size is 195×150 mm and is illustrated on the left of Figure 2.18. The amount of text on a page varies and there are many line drawings, some of which take a double spread. Folios are in the outer margin level with the top line of text.

On the right of this figure is another art book, namely *Dürer* by Fedja Anzelewsky published by Chartwell Books in 1980. This is set in two columns with 14pt leading on a 23.5pc measure, although there are more illustrations than text. The page size is 280×240 mm, considerably larger than its companion in the figure, yet with much smaller margins. Roman numerals are used for chapter heads which are set flushleft in a large font. Immediately below the chapter head is a line of 'section' titles, flushleft, in a font size intermediate between the chapter head and the body. A centered dot is used to separate the section titles. Folios are set in the foot flush with the outside of the typeblock; there are no running heads. The Table of Contents title matches the chapter heads. Chapter title entries are set flushleft with their page numbers flushright. The section titles are set in a line below the chapter entry, again separated by centered dots. In the text, references to illustrations are placed in the outside margins in a small font.

Two more layouts for illustrated books are given in Figure 2.19. In this case the illustrations are drawings in landscape mode (i.e., they are wider than they are high); the shape of the drawings has had a major effect on the page proportions. In the case on the left the page proportion is in the ratio $\pi : e$. The measure is longer than usual at 37pc and to compensate for this the leading of 17pt is also larger than customary. It is typeset in Centaur. The book is *Hammer and Hand* by Raymond Lister with drawings by Richard Bawden. It was published in 1969 by Cambridge University Press and is another of the University Printer's Christmas books. Folios are in a large font at the outside edge of the page and level with the top text line.

Shown on the right of Figure 2.19 is *Hokusai — One Hundred Poets* by Peter Morse and published by George Braziller in 1989. The introductory text is set in two columns as shown. The body consists of illustrations of Japanese wood block prints, originally in the large *oban* size of about 250×380 mm. The half-title is set in a large font in the top right hand corner of the page, but the text on the title page is centered. The main body is

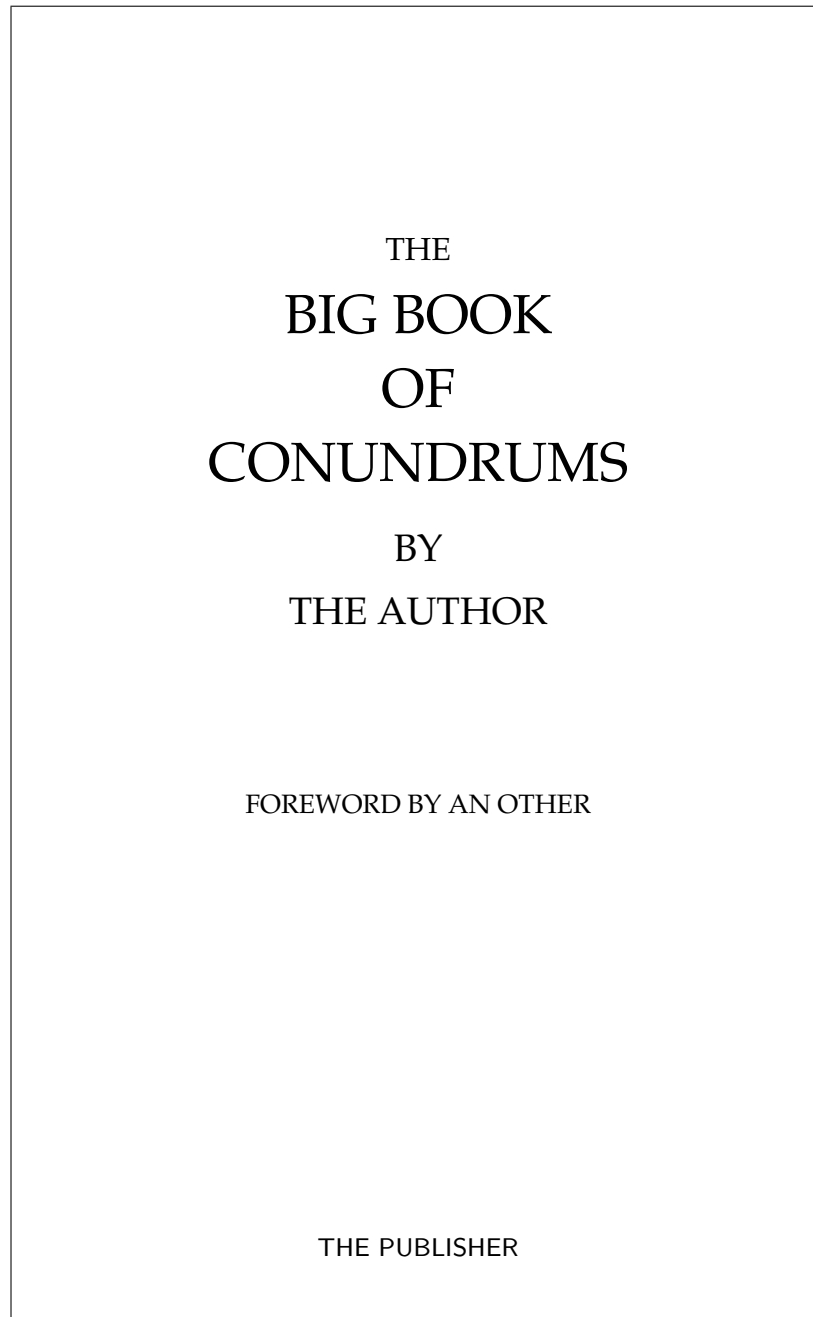


Figure 2.17: Title page design based on Adrian Wilson's *The Design of Books*



Figure 2.18: Two spreads: (Left) England, 1972. (Right) Switzerland, 1980.

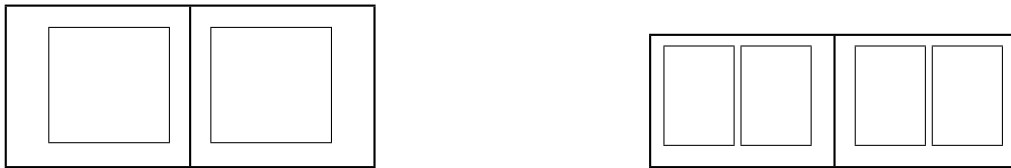


Figure 2.19: Two spreads: (Left) England, 1969. (Right) USA, 1989.

organised with a wood block print on each recto page and the commentary on the facing verso page. At the top of each commentary page, centered, is the number of the print, a short rule, the title of the print in a large italic font, a longer rule, and then side by side the poem in Japanese and an English translation. The commentary itself is underneath, set in three raggedright columns; minor heading in the commentary are flushleft in small caps. The folio is at the center of the foot. At the end of the book is a commentary on those poems where there are no known illustrations; this is typeset raggedright in four columns.

2.2.1 A geometric construction

Nowadays it is easy to pick and calculate any kind of page proportion that takes your fancy, but how did the early printers do it? They certainly did not have the use of calculators and I suspect that they had only enough arithmetic to keep their accounts. Printing was a craft and craftsmen did not release their trade secrets lightly. I believe that most of the designs were based on simple geometric figures, which required nothing more than a ruler and a pair of compasses.

Jan Tschichold gives a simple construction for the layout of many of the books based on Gutenberg's work [Tsc91, pages 44–57], which is shown in Figure 2.20. The construction actually divides the page up into ninths (the point P in the diagram, which is at the intersection of the main and half diagonal construction lines, is one third of the way down and across both the page and the typeblock). This construction can be used no matter what the page proportions and will give the same relative result.

2.3 THE TYPEBLOCK

The typeblock is not just a rectangular block of text. If the typeblock does consist of text, then this will normally be broken up into paragraphs; it is not good authorial style to have paragraphs that are longer than a page. Also, the typeblock may include tables and illustrations which provide relief from straight text. Some pages may have chapter or

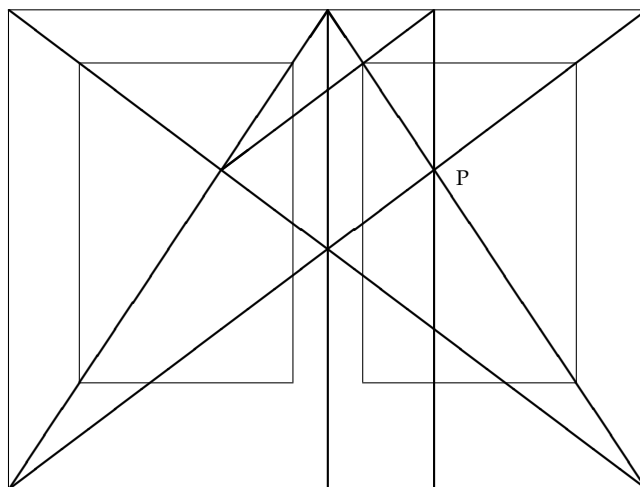


Figure 2.20: The construction of the Gutenberg page design

section headings on them which will also break the run of the text. In general the typeblock will be a mixture of text, white space, and possibly non-text items.

Consider a typeblock that includes no illustrations or tables. 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.

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 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 Bringhurst'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

$$L_{65} = 2.042\alpha + 33.41 \quad (2.7)$$

and

$$L_{45} = 1.415\alpha + 23.03 \quad (2.8)$$

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 2.2: 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

The vertical height of the typeblock should be constant from page to page. The lines of text on facing pages should be aligned horizontally across the spine, which also means that they will be at the same place on both sides of a leaf. Alignment across the spine means that the eye is not distracted by an irregularity at the centre of a spread, and leaf alignment stops ghosting of text through a thin page, giving a crisper look to the work. So, the spacing between lines should be constant. This implies that the depth of the typeblock should be an integral multiple of the space required for each line; that is, be specified as a multiple of the leading. A ten point type, for example, will normally have two points between lines, to give a leading of 12 points. This can be written as 10/12. Usefully, one pica is 12 points so with a 12pt leading vertical distances can be conveniently expressed in picas (one pica per line). Another implication of this is that any space left for illustrations or tables, or the amount of space taken by chapter and section headings should also be an integer multiple of the leading.

A ten point type set solid is described as 10/10. The theoretical face of the type is ten points high, from the top of a *d* to the bottom of a *p*, and the distance of the baseline of one row of text to the next row of text is also ten points. Note that if a *p* is vertically above a *b* then the ascender of the *b* will meet the descender of the *p*. To avoid this, the vertical separation between baselines is increased above the type size. Adding two extra points of vertical space allows the text to breathe, and gives a leading of 12 points. Few fonts read well when set solid. Typical settings are 9/11, 10/12, 11/13 and 12/15. Longer measures require more leading than shorter ones, as do darker and larger fonts compared with lighter and smaller fonts. More leading is also useful if the text contains many super- or sub-scripts, or many uppercase letters.

2.3.1 Page color

One of the aims of the typographer is to produce pages that are uniform in 'color'. By this they mean that the typeblock has a reasonably constant grayness, not being broken up by too much white space which is a distraction to the reader. There will be white space around headings, which is acceptable as a heading is meant to attract attention. There may be white space between paragraphs, and this is usually under the control of the designer. But there may be vertical rivulets, or even rivers, of white space when the interword spaces on adjacent lines coincide; fixing this usually requires some handwork, either by the author changing his wording so as to alter the location of the spaces, or by the typesetter tweaking a little bit.

Another form of distraction is if too many lines end with hyphens, or several adjacent lines start or end with the same text; not only does this cause a stack of identical characters but will make it harder for the reader to reliably jump to the next line.

The main font used controls the depth of the color of a page. To see what color is produced by a particular font it is necessary to look at a fairly long, preferably a page, piece of normal text. Fonts from different families produce different colors, and so may mixed fonts from the same family. You can try this yourself by typesetting the same page in, say, Computer Modern Roman, Italic, and Sans fonts. The books by Rogers [Rog43], Lawson [Law90], Dowding [Dow98], and Morison [Mor99] all show pages set in many different fonts.

2.3.2 Legibility

One of the principle requirements on the typography of a document is that the document is *legible*. Legibility means that the document is designed to be easily read under a certain set of circumstances. The criteria for legibility on a poster that is placed on the side of a bus, for example, are different from those that apply to a book to be read while sitting in an easy chair. Essentially, the viewer should be able to read the document with no physical strain caused by the appearance, but the contents, of course, may lead to anything ranging from acute mental strain to extreme boredom.

Type faces and the layout of the typeblock must be chosen to optimise between legibility and 'artistic' presentation. The design of the document should be almost invisible, giving full compliments to the author's communication. However, if you are a master, like Hermann Zapf [Zap00], you can break the rules.

Type faces

The first European letter forms that have survived are Greek inscriptions carved into stone. These were freehand carvings with thin strokes. In time, the lettering became thicker and serifs started to appear. The Romans picked up on this later style of letter form. In carving inscriptions, they first wrote the inscription on the stone using a broad, flat brush. This naturally led to serifs and differing thicknesses of the letter strokes, depending on the angle of the stroke with respect to the movement and orientation of the brush. Similarly the written letterforms included serifs.

Between the Roman times and Gutenberg there were many changes and experiments in European letterforms. The scribes used different scripts for titles, subheads, continuous text, illuminated initial letters, and so on. In time, two families of letterforms evolved, called *majuscules* and *minuscules*. The former were larger and more formal, while the latter were smaller and less formal. We now call these two divisions upper case and lower case. The upper case derives from Roman times, while the lower case acquired its fundamental form during the reign of the Holy Roman Emperor Charlemagne a thousand years later. In order to promote communication throughout his wide flung empire the Anglo Saxon Benedictine monk Alcuin, at the behest of Charlemagne, established a common script to be used; this is now called Carolingian minuscule. A further division also appeared, between black letter (what is commonly referred to as Gothic or Old English) type and the roman type.

These types were all upright. Italic letterforms were cut in Italy in the early sixteenth century, as a more cursive style. Initially these were lower case only, used in conjunction with upper case roman. By the end of the century, sloped roman capitals were also in use with italic.

The late nineteenth century saw the appearance again of sans-serif typefaces.

Looking carefully at seriffed and sans-serif fonts it is apparent that the serifs have three main functions:

1. They help to keep letters apart.
2. At the same time, they help to keep letters in a word together. This helps with legibility as research has shown that we tend to recognize words by the shape of the word rather than by individual characters.
3. They help to differentiate between individual but similar letters.

Long experience has shown that a seriffed font is easier to read⁵ than a sans-serif font, particularly if part of the text is obscured. You can try an experiment yourself to verify this. Try writing a phrase, once using a sans-serif font and then with a seriffed font. Cover up the top halves of the two phrases and try to make out what they say. Then repeat this, except this time cover up the bottom halves of the phrase. Which is easier to read? Here are some example characters, firstly in san-serif:

a c l m n p q o

and then in roman:

a c l m n p q o

Sans-serif fonts often require context to decipher the word. For example [McL80], seeing this in isolation

III

does it stand for ‘III’, ‘one hundred and eleven’, ‘three’, or something completely different like a dingbat or a set of cricket stumps?

There are three generally agreed legibility principles for setting text for continuous reading.

1. *Sans-serif type is intrinsically less legible than seriffed type* [Whe95].

We have already seen that this is the case — there is more variety among seriffed letters than among sans-serif letters. Further, serifs perform other functions as well, such as binding letters together within a word.

This is not to say that a sans-serif letterform is always more illegible than a roman one. A poor seriffed form can be much more illegible than a well used good sans-serif. In general, there is an illegibility factor associated with sans-serif that must be borne in mind; for general *continuous* reading, a good seriffed form is more likely to be easy on the eye than a good sans form.

2. *Well designed upper and lower case roman type is easier to read than any of its variants.*

This is a guiding principle with many exceptions. Among the variants can be considered to be italic and bold types. These have usually been designed for a special purpose, like emphasizing certain pieces of text, rather than for general legibility. Some italic types, though, are as legible as their roman counterparts. In the seventeenth century many books were set entirely in italic, but we have become accustomed to the roman type.

3. *Words should be set closer together than the space between lines.*

All text is a mixture of ink and white space. The eye, when reading, tends to jump over the white spaces. Given a choice between two spaces, it will tend to jump over the smaller of the two. If the word spacing is greater than the line spacing, then you can find yourself skipping from one line to the next before finishing the first one.

Further, if the lines are too long, then when the eye jumps back from the end of one line to the start of the next, it may have difficulty in picking up the correct one.

⁵This is actually somewhat contentious as some take the view that with enough practice, sans-serif is just as easy to read.

Text lines are justified by altering the inter-word spacing, and possibly by hyphenating the last word on the line if the spacing would be too bad otherwise. Sans-serif fonts often look best if set ragged right, as this will keep the inter-word spacing constant. Text set in narrow columns also often looks best when set ragged right.

Seriffed versus sans-serifed fonts

As noted earlier there seems to be a permanent debate over the use of serifed and sans fonts. You will have to make up your own mind as to what is best for any particular work, but here are a few general comments from some of the literature on the subject.

Bohle [Boh90] notes: Readers prefer a roman typeface for body type because they are most used to seeing that face [Reh72]. Roman type may well also be more readable than sans serif faces because the serifs help connect the letters to form the word shape when we read [Reh72].

Craig [Cra92] says: You will find that the serifs on a typeface facilitate the horizontal flow necessary to comfortable reading.

Degani [Deg92] in a study of pilots reading checklists in emergency cockpit situations decided that sans serif faces were better than serif faces.

Schrivver [Sch97] notes: Serif and sans serif typefaces are likely to be equally preferred by readers [HR83, Tin63] and read equally quickly [Gou87, HR83, Zac69]. Serif faces may be easier to read in continuous text than sans serif faces [Bur59, HK75, RAE71, Whe95].

Wheildon [Whe95] did a series of studies with around 250 readers in Sydney, Australia, asking them to rate serif and sans fonts in a variety of uses. Among the many results he reported:

- More than five times as many readers are likely to show good comprehension when a serif body type is used instead of a sans serif body type.
- The top half of [upper case] letters is more recognizable than the bottom half.
- There is little difference in legibility between headlines [section titles] set in serif and sans serif typefaces, or between roman and italic.
- Headlines set in capital letters are significantly less legible than those set in lower case.

The consensus, such as it is, seems to lean towards serified typefaces for continuous reading, but for titling the choice is wide open.

To finish off in a lighter vein, Daniel Luecking had this to say on the subject in a posting to CTT in January 2002.

It is often conjectured that serified fonts are easier to read because the serifs contribute more points of difference between words. This is often countered with the conjecture that they are easier to read because that is what we are used to reading. But no one can doubt that words like *Ill*, *Iliad* and *Illinois* in a sans-serif font [e.g., *Ill*, *Iliad* and *Illinois*] are going to cause the eye/brain system at last momentary confusion while it sorts out which plain vertical lines are uppercase i's and which lowercase L's.

I don't know if this contributes anything, but I can say unequivocally that serif fonts are somewhat easier to read upside down than sans-serif, but sans-serif is far easier to read mirrored than serif. (I spent much of my time as a child reading comics on the floor with my brother. As he hated reading any way but

straight on, we faced in different directions and I saw the page upside down. I tried mirror reading just to see if I could do that as easily. Serif fonts were almost impossible, sans-serif actually quite easy.)

He later expanded on the mirror reading to me as follows.

Here's an interesting (to me) anecdote about mirror reading: I was waiting in line at an airport lunch counter, reading the menu posted on the wall, when it suddenly struck me as odd that the menu was on the wall opposite (so that one had to turn away from the counter to read it). Then I realized in a sort of flash that I was reading it from a mirror. I turned to look at the real menu and was momentarily disoriented (while my brain turned itself around I guess) before I could read the actual menu. That was when I first ran some tests to see why that was so easy to read and other mirror writing was not. It seemed to be serif vs sans-serif, but it might also be the typical letter forms: the typical serif lower case 'a', the one with the 'flag' above the bowl [e.g., a], is particularly difficult to recognize compared to the simple 'circle plus stick' [e.g., a] form.

Some sort of dyslexia (or eulexia), no doubt, when backwards words are nearly as easy to read as normal ones.

2.3.3 Widows and orphans

Inconvenient page breaks can also cause a hiatus in the reader's perusal of a work. These happen when a page break occurs near the start or end of a paragraph.

A *widow* is where the last line of a paragraph is the first line on the page. The term is sometimes also used to refer to when the last word in a paragraph is on a line by itself. A widow looks forlorn. In German they are called *Hurenkinder* — whores' children — which seems rather cruel to me. As Robert Bringhurst said, 'A widow has a past but no future'. Typographically, widows should be avoided as they are a weak start to a page and may optically destroy the page and type rectangle. However, a single widow is not too troubling if the header includes a rule across the width of the typeblock. Especially to be avoided are widows that are the only line on a page, for example at the end of a chapter. Five lines on the last page of a chapter is a reasonable minimum. Jan Tschichold [Tsc91] claims that *Hurenkinder* can always be avoided by even if a recto (verso) page must be made a line shorter or longer than the corresponding verso (recto) page, which he considers to be less of an affront than widows.

An orphan is not nearly so troubling to typographers as a widow. An *orphan* is where the first one or two lines of a paragraph are at the bottom of a page. In German they are called *Schusterjungen* — cobbler's apprentices. Bringhurst's memory trick for orphans is, 'An orphan has a future but no past'.

2.3.4 Paragraphs and versals

Early books did not have paragraphs as we know them nowadays; the text was written continuously, except for a break at a major division like the start of a new book in a bible. Instead the scribes used a symbol like ¶ (the pilcrow) to mark the beginning of paragraphs. This symbol is derived from the Greek Π, for *parágraphos*. Mind you, they often did not use any punctuation at all and were sparing in their use of uppercase letters, so you might have seen something like this⁶

⁶But probably not. The two 'paragraphs' are Latin abecedarian sentences.

usque ¶ te canit adcelebratque polus rex gazifier hymnis ¶ transzephyrique
globum scandunt tua facta per axem

Often the ¶ was colored red by the rubricators and the scribe, or printer, would leave a blank space for the rubricator to add the ¶. This did not always happen and the start of a paragraph eventually became marked by a space rather than a symbol.

Nowadays paragraphs are ended by stopping the line of text at the end of the paragraph, and then starting the next paragraph on a new line. The question then becomes: how do you indicate a new paragraph when the last line of the previous paragraph fills up the measure? There are two solutions, which unfortunately you sometimes see combined. Either indent the beginning of the first line of each paragraph, or put additional vertical space between the last and first lines of paragraphs.

The traditional technique, which has served well for centuries, is to indent the first line of a paragraph. The indentation need not be large, about an em will be enough, but more will be required if the typeblock is wide.

The other method is used mainly in business letters and is a recent invention. The first lines of paragraphs are not indented and typically one blank line is left between paragraphs. This may perhaps be acceptable when using a typewriter, but seems to have no real justification aesthetically. There is also the problem when a paragraph both ends with a full line and ends a page. As the next paragraph then starts at the top of the next page, the blank line separating the two paragraphs has effectively dissappeared, thus leaving the reader in a possible state of uncertainty as to whether the paragraph continues across the page break or not.

If the paragraph is the first one after a heading, then there is no need to indicate that it is a new paragraph — it is obvious from its position. So, the first paragraph after a heading need not be indented, and for some centuries now the tradition is not to indent after a heading. In some novels only chapters are headed yet each chapter is broken into sections by putting additional vertical blank space between the sections. Like nonindented paragraphs, this can cause problems where a section division coincides with a page break. In this case, typographers sometimes use a decoration to separate sections (for example, a short centered row of a few asterisks).

SOME TYPOGRAPHERS like to start the first paragraph in a chapter with a versal. A *versal* is a large initial letter, either raised or dropped. This comes from the scribal tradition of illuminating the first letter of a manuscript. The versal may be raised or dropped, as already noted, or it may be placed in the margin, or otherwise treated in a special manner.

SOME VERSALS, especially dropped versals, are very difficult to typeset correctly. Many attempts of this kind are abject failures, so be warned. For example, compare the dropped versals at the start of these first two paragraphs. They are both of the same letter and font, yet the first one is horrible compared to the one starting this paragraph.

A RAISED VERSAL is often easier to use to start a paragraph than a dropped versal. However, a raised versal should only be used where there is naturally some vertical space above it. As you can see, extra spacing has had to be inserted before this paragraph to accommodate the versal. There are still problems with typesetting a raised versal but as these tend to be subtler than with a dropped versal, readers are less likely to notice problems.

Typically, small caps are used for a little while following a versal to provide a transition between the large versal font and the normal body font. These should not continue throughout the first line as this tends to divorce it from the remainder of the paragraph. ANOTHER WAY OF STARTING a paragraph is to use small caps for the first few words. The font difference highlights the start of the paragraph but in a much quieter manner than a versal does. Using normal sized upper-case instead of the small caps is too much of a contrast with the lower-case.

2.3.5 Footnotes

Footnotes are considered to be part of the typeblock. They are typeset in the space allocated for the typeblock, in contrast to footers which are typeset below the typeblock.

Footnotes are normally set in the same type style as the typeblock. That is, if an upright seriffed font is used for the typeblock, it is also used for the footnote. The type size is smaller to distinguish the note from the body text and often the leading in the footnote is also reduced from that in the main text body. The bottom footnote line should be at the same height as the bottom line of the typeblock. This usually requires some adjustment of the vertical space before the footnotes.

A vertical blank space is often used to set off the footnotes from the main text, and sometimes a short horizontal line is also used as demarcation.

2.4 FOLIOS

The word *folio* is a homonym. It can mean a leaf (two back-to-back pages) in a book, the size of a book or a book of folded sheets (as in Shakespeare's first folio), or the printed page number in a book. Here I use folio in this last sense.

Documents should have folios, at a minimum to help the reader know where he is. Occasionally books have their folios placed near the spine but this positioning is unhelpful for navigation. The more usual positions are either centered with respect to the typeblock or aligned with the outside of the typeblock, and sometimes even in the outside margin. The folios can be either at the top or bottom of the page but at least on pages with chapter openings are normally placed at the bottom of the page so that they do not distract from the title text.

Every page in a book is numbered, even if the page does not have a folio. In books, the folios for the front material are often in roman numerals. The main matter and back matter folios are arabic numerals, with the sequence starting from 1 after the front matter. In certain technical documents, folios may be in the form of chapter number and page number, with the page number starting from 1 in each new chapter. Other folio schemes are possible but unusual.

Folios should be placed harmoniously with respect to the typeblock and page margins. The font used for the folios need not be the same as that for the typeblock but must at least be complementary and non-intrusive.

2.5 HEADERS AND FOOTERS

Headers and footers are repetitive material that is placed at either the head or the foot of the page. Typically, folios are headers or footers, but not always as sometimes they are placed in the margin at or below the first line in the typeblock.

For the time being I will not distinguish between headers and footers and just use the word header. Sometimes the header is purely decorative (apart from a folio) like a horizontal line or some other non-textual marking. Normally, though, they have a functional use in helping the reader locate himself in the document.

The most ubiquitous header is one which gives the title of the document. If this is the only header, then I consider this to be decorative rather than functional. As a reader I know what document I am reading and do not need to be reminded every time I turn a page. More useful are headers that identify the current part of the document, like a chapter title or number. When you put the document down and pick it up later to continue reading, these help you find your place, or if you need to refer back to a previous chapter for some reason, then it is a boon to have a chapter heading on each spread. The minimally functional headers are where the document title is on one page and the chapter heading is on the facing page. In more technical documents it may be more useful to have headers of chapter and section titles on alternate pages.

Occasionally both headers and footers are used, in which case one normally has constant text, like a copyright notice. I have the feeling that using the latter is only functional for the publishers of the document when they fear photocopying or some such.

The header text is usually aligned with the spine side of the typeblock, but may be centered on top of the typeblock. In any event, it should not interfere with the folio. The type style need not be the same as the style for the typeblock. For example, headers could be set in italic or small caps, which, however, must blend with the style used for the typeblock.

2.6 ELECTRONIC BOOKS

For want of a better term I am calling electronic books, or Ebooks, those documents intended to be read on a computer screen. The vast bulk of Ebooks are in the form of email but I'm more interested here in publications that are akin to hardcopy reports and books that require more time than a few minutes to read.

Unlike real books which have been available for hundreds of years there is virtually no experience to act as a guide in suggesting how Ebooks should appear. However, I offer some suggestions for the layout of Ebooks, based on my experience of such works. Not considered are internal navigation aids (e.g., hyperlinks) within and between Ebooks, nor HTML documents where the visual appearance is meant to set by the viewing software and not by the publisher.

The publication medium is obviously very different — a TV-style screen with limited resolution and pretty much fixed in position versus foldable and markable paper held where the reader finds it best. These differences lead to the following suggestions.

A book can be held at whatever distance is comfortable for reading, even when standing up. The computer user is normally either sitting in a chair with the monitor on a desk or table, or is trying to read from a laptop, which may be lighter but nobody would want to hold one for any length of time. To try and alleviate the physical constraints on the Ebook reader the font size should be larger than normal for a similar printed book. This will provide a wider viewing range. A larger font will also tend to increase the sharpness of the print as more pixels will be available for displaying each character. The font size should not be less than 12pt. The font may have to be more robust than you would normally use for printing, as fine hairlines or small serifs will not display well unless on a high resolution screen.

I find it extremely annoying if I have to keep scrolling up and down to read a page. Each page should fit within the screen, which means that Ebook pages will be shorter than traditional pages. A suggested size for an Ebook page, in round numbers, is about 9 by 6 inches [Ado01] or 23 by 15 centimetres overall.

The page design for printed books is based on a double spread. For Ebooks the design should be based on a single page. The typeblock must be centered on the page otherwise it gets tiring, not to mention aggravating, if your eyes have to flip from side to side when moving from one page to the next. Likewise any header and the top of the typeblock must be at a constant height on the screen. A constant position for the bottom of the text is not nearly so critical.

It is more difficult with an Ebook than with a paper book to flip through it to find a particular place. Navigation aids — headers and footers — are therefore more critical. Each page should have both a chapter (perhaps also a section) header title and a page number. Note that I'm not considering HTML publications.

Many viewers for Ebooks let you jump to a particular page. The page numbers that they use, though, are often based on the sequence number from the first page, not the displayed folio. In such cases it can be helpful to arrange for a continuous sequence of page numbers, even if the folios are printed using different styles. For example, if the front matter uses roman numerals and the main matter arabic numerals and the last page of the front matter is page xi, then make the first page of the main matter page 12.

I see no point in Ebooks having any blank pages — effectively the concept of recto and verso pages is irrelevant.

Some printed books have illustrations that are tipped in, and the tipped in pages are sometimes excluded from the pagination. In an Ebook the illustrations have to be 'electronically tipped in' in some fashion, either by including the electronic source of the illustrations or by providing some navigation link to them. Especially in the former case, the tipped in elements should be included in the pagination.

Don't forget that a significant percentage of the population is color-blind. The most common form is a reduced ability to distinguish between red and green; for example some shades of pink may be perceived as being a shade of blue, or lemons, oranges and limes may all appear to be the same color. Along with color-blindness there may be a reduced capacity to remember colors.

I have seen Ebooks where color has been liberally used to indicate, say, different revisions of the text or different sources for the data in a graph. Unless the colors used are really distinctive 10% or more of the potential readership will be lost or confused. Further, Ebooks may be printed for reading off-line and if a non-color printer is used then any colors will appear as shades of grey; these must be such that they are both readily distinguishable and legible. Yellow on white is almost as difficult to read as off-white on white or navy blue on black, all of which I have seen on web sites but rarely have I seen the text after I have tried to print the page.

Three

Styling the elements

A book should present a consistent typographic style throughout, although some elements, principally those in the front matter and back matter may be treated slightly differently than the main body.

Much of this chapter is based my interpretation of my namesake's work [Wil93] and the *Chicago Manual of Style* [Chi93].

3.1 FRONT MATTER

3.1.1 Title pages

The main and half-title pages are the gatekeepers to the book. As such, they need to be welcoming and give an indication of the 'look and feel' of the contents. You don't want to scare off potential readers before they have even cursorily scanned the contents.

The half-title, or bastard title, page contains just the title of the work, which is traditionally set high on the page — perhaps about as high as the chapter openings. The title page itself presents the title in full, the author and maybe the illustrator or other names likely to attract the reader, and perhaps the publisher and date. Both the title pages are recto but the full title may be a double spread. The full title layout in particular must be both attractive and informative.

Advertising

If the book is one of a series, or the author has been prolific, then details of associated works may be provided. Titles by the same author are usually set in italic, and a series title perhaps in small caps. The font size should be no larger than for the main text. This advertising material may be put on the copyright page but it is more often on the recto page immediately after the half-title, before the title page, or on the verso of the half-title page.

Frontispiece

The traditional place for a frontispiece, which may be the only illustration in the book, is facing the title page. Every attempt must be made to make the resulting double spread hamonious.

3.1.2 Copyright page

The copyright and related publishing information is set in small type on the verso of the title page. In some instances the book designer's name may be listed among the small print.

3.1.3 Dedication

A dedication is nearly always on a recto page and simply typeset. If pages are limited it could be placed at the top of the copyright page instead.

3.1.4 Foreword and preface

The same type size should be used for the headings 'foreword', 'preface', 'acknowledgements', etc., and the similar ones in the back matter. This may be the same size as the chapter heads, or smaller. The body type should be the same as for the main matter text. The foreword starts on a recto page. It may face the copyright page, or if there is a dedication it will face the dedication's blank verso.

The preface, which is the author's opening statement, is treated like a chapter opening, and commences on a recto page.

3.1.5 Acknowledgements

If there are any acknowledgements and they require only a few sentences then they are often put at the end of the preface, if there is one. Otherwise the acknowledgements should be treated as a distinct unit, like a foreword or preface, and commence on a recto page.

3.1.6 Contents and illustration lists

The table of contents is often laid out so that the page numbers are not too distant from the titles, thus reducing the need for dotted lines, even if this makes the contents block narrower than the text page width. An alternate strategy is to use more interlinear space between the entries so that there is little or no difficulty in recognising which page numbers belong to which titles. The parts of major sections should be clearly separated to aid visual navigation.

If the captions in an illustration list are short, and it follows the contents list then it should be set in the same style as the contents. If, however, the list is at the back of the book a smaller font size could be used.

3.1.7 Introduction

The heading for an Introduction is an unnumbered chapter opening on a recto page. The body text is set in the same font as the main body text.

3.1.8 Part title page

If the book is divided into Parts then the Part One page could either precede or follow the Introduction. Ideally the layout of the title pages for the Parts should follow the layout of the book's half-title page to provide a cohesiveness throughout the work.

3.2 MAIN MATTER

3.2.1 Chapter openings

Normally, chapters are the major divisions of a (volume of a) book, but if it is long it may have higher divisions, such as parts, or even books. In any event, chapters are numbered consecutively throughout the volume. The numbering is usually arabic but if there are few chapters, or there are numbered subdivisions within chapters, or the work has classical allusions, then roman numerals might be appropriate.

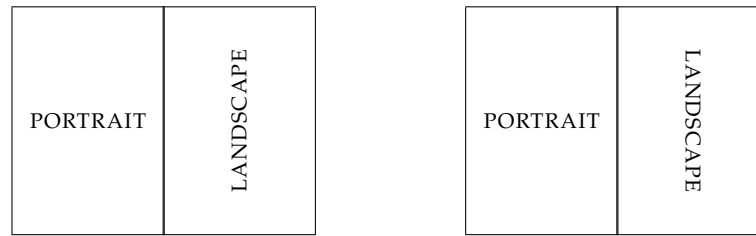


Figure 3.1: Portrait and landscape spreads: the layout on the left is preferable to the one on the right



Figure 3.2: Landscape and portrait spreads: the layout on the left is preferable to the one on the right

Chapters often begin on recto pages but if there are many short chapters or reproduction cost is all important then they may begin on the page immediately following the end of the previous chapter, or, in rare cases, even on the same page if there is enough space. In any event, the first chapter should start on a recto page.

3.2.2 Mixed portrait and landscape pages

It is usual for all the pages in a document to be in portrait but this is not always possible if there are illustrations or tabular material that is better displayed in landscape orientation rather than portrait.

When a double spread consists of a portrait page and a landscape page, as in Figure 3.1 and 3.2, there are two choices for which way the landscape page is turned with respect to the portrait page. In each case it seems more natural to me when the document has to be turned to the right to view the landscape page.

Figure 3.3 shows double spreads where both pages are in landscape orientation. Whichever way they face, they must both face in the same direction so that the document has only to be turned once to read both of them.

The general rule for mixed landscape and portrait pages is that the document is held in one position to read the portrait pages and is turned in a consistent direction, usually to the right, to read the landscape pages. The key is consistency.

3.2.3 Extracts

Typographers use ‘extract’ as a generic term for what I would think of as a quotation. Essentially a quotation is an extract from some source. Quotations from other works, unless they are so short as to be significantly less than a line, should be set off from the main text.

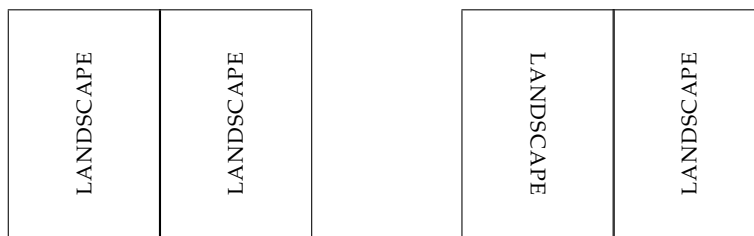


Figure 3.3: Double landscape spreads: never use the layout on the right

This could be as an indented block, or by using a different type style or size, or a combination of these. A size difference of one or two points from the body size is usually enough to be distinctive, for instance 11/13 or 10/12 point for a 12/14 point body size. In any event, extra space, at least two or three points, should be inserted and below the extract.

3.2.4 Footnotes and endnotes

This section is a synthesis of the views of Ruari McLean [McL80], Jan Tschichold [Tsc91] and Emerson Wulling [Wul53].

Footnotes are ancilliary the the main material and expand in some way upon the current theme. For instance, remarks that are too large or off the main thread, or some side comment by the author, may be sunk to a footnote at the bottom of the page. By definition, a footnote is placed at the bottom of a page but, if it is long or space is short, may run over to a second, or even a third page. A footnote should have some immediate relevance to the reader.

Endnotes, which are collected together at the end of the document, include material similar to that in footnotes, but which is not of immediate interest. If you use endnotes it is safe to assume that only a small percentage of your readers will ever correlate them with the text.

Within the text the presence of a footnote is indicated by a raised ‘reference mark’ following the word, phrase, or sentence to which it refers. The same mark is used to introduce the footnote at the bottom of the page. The reference mark may be either a symbol or a number. For illustrative purposes I’m using symbols as markers in this section.

If there are many footnotes then it is convenient for the reader if numbers are used for the marks. The numbering may be continuous throughout the document, or start afresh with each chapter; starting anew on each page may lead to some confusion. When there is only an occasional footnote then symbols are usually preferred as reference marks.

Endnotes may or may not be marked in the main text. If they are marked then numbers should be used, not symbols. If there are both footnotes and marked endnotes then different series of marks must be used for the two classes of notes.

There is some debate as to how reference marks should relate to the marked element in the main text. For example, should the mark immediately follow the element* or should there be a thin space[†] separating the two. A convenient procedure is to use a thin space

*This mark immediately follows the main element.

[†]There is a thin space between this mark and the element it is attached to.

between the element[‡] and the mark when the end of the element is tall, and none[§] when the end of the element is low. There is no need for any extra space between punctuation and a reference mark.[¶]

McLean, Tschichold, and Wulling are all agreed that there should not be a rule separating the main text from any footnotes — a space and change in font size is sufficient to distinguish the two. The font size for footnotes is usually two sizes smaller than the main font but with the same leading. For example if the main text is set 10/12 then footnotes would be set 8/10. Notes to tables, though, are often set even smaller; for instance at 6pt or 7pt in a 10pt document. Each footnote should be introduced by the appropriate reference mark in the same font size as the note itself. If it is not raised then follow it by a period if it is a number, or by a space if a symbol, in order to distinguish it from the note's text. If there are any footnotes on a short page, such as perhaps the last page of a chapter, they are placed at the bottom of the page, not immediately below the last line of the main text.

Endnotes may be set in the same font as the main text, but usually in the same size as for footnotes.^{||}

Endnotes may be grouped at the end of each chapter or collected together towards the end of the document. If the latter, then they should be presented in groups corresponding to the noted chapters. It is a courtesy to the reader to indicate the page which gives rise to each note so that backward reference, as well as forward reference, is facilitated; this is especially important if there is no endnote mark in the main text. Endnotes that have no reference mark in the text are usually tagged with some words from the main text that identify the idea or statement that they are referring to. In an endnote listing note numbers are usually either indented or the note is set flush-and-hang style; that is, with the first line set flushleft and any remaining lines indented.

Whether a note is presented as a footnote or an endnote, it should always finish with a period.

3.3 BACK MATTER

Divisions in the back matter are not numbered.

In commercial printing, saving a page here and a page there can save the publisher money and hopefully at least some of the reduced cost will be passed on to the readers. One way of reducing the number of pages is by reducing the font size. The material in the back matter is in some sense auxiliary to the main matter, hence of less importance, and some of it may then be reasonably set in smaller type.

3.3.1 Appendices

Appendices come immediately after the main text. Depending on their importance and interest they can be set in the same manner as the main text. If the appendices consist of long supporting documents they could be set in a type one or two points smaller than the main text.

In some instances there are appendices at the end of individual chapters, where they form the last divisions of the chapters, and are treated as any of the other divisions.

[‡]A thin space is used here.

[§]There is no extra space here.

[¶]There is no extra space here.

^{||}That is, two sizes smaller than the main text, but with the same leading.

3.3.2 Endnotes

In an endnote listing, note numbers are usually either indented or the note is set flush-and-hang style; that is, with the first line set flushleft and any remaining lines indented. There are, of course, corresponding numbers at the appropriate points in the main text.

In another style, the endnotes are identified by a phrase taken from the main text together with the relevant page number. In this case there are no numbers in the text to disturb the flow, but that often means that the notes never get read.

The notes are often set in the same sized type as used for footnotes or, if they are exceptionally interesting, in a size intermediate with the main text.

3.3.3 Bibliography

The list of books, etc., that the author has used as source material is usually placed at the back of the book under the title 'Bibliography'. In some works there may be a bibliography at the end of each chapter and the title 'References' is often used for these.

A font size intermediate between those for quotations and footnotes is very reasonable, and a slight extra space between entries, say two points, can improve the readability.

3.3.4 Glossary

The list of definitions of terms or symbols used in the text normally comes towards the end of a book, although it could as well come towards the end of the front matter, or a symbol list in the front matter and a glossary in the back matter. The terms are usually set in italics, or in textbooks in bold. Most often a flush-and-hang style is used, with perhaps one or two points extra leading between the entries.

The type size could be the same as for quotations if the glossary is short, or the size used for the bibliography for longer lists.

3.3.5 Index

The entries in an index are usually short and most indexes are set in two, or more columns. As examples, author's names are usually relatively short so an index of names would typically be in two columns; on the other hand, verse lines are relatively long and an index of first lines is often set as a single column. In either case, the entries are usually set raggedright with the page numbers close to the corresponding item's text. In multicolumn setting the gutter between the columns must be wide enough, at least a pica, so that the eye does not jump across it when reading an entry. The entries are normally set flush-and-hang.

When there are subentries, or sub-subentries, they are typically each indented by 1em with respect to the major entry.

A change of collation, such as between entries starting with 'P' and those with 'Q' should be signalled by at least one or two blank lines. If the index is long, then a suitable character (e.g., 'Q') or word should be used as part of the break, indicating what is coming next. The index could be set in the same size type as the bibliography.

3.4 TYPE SIZE

As indicated above, the type size is normally related to the 'importance' of what is being set. Chapter headings are set in large type and footnotes set in small type. Of course, it is a matter of judgement as to what 'important' means in any given work. Some possible

Table 3.1: Some relative type sizes for elements of books

Body size	12/14	10/12
Extracts	10/12	9/10
Bibliography	9/10	8/9
Glossary	9/10	8/9
Footnotes	8/9	8/8

combinations of type sizes are given in Table 3.1 though these should be considered as starting points for a design rather than fixed rules.

3.5 POEMS AND PLAYS

In literature such as poems or plays the length of the line is determined by the author whereas in prose works the book designer establishes the measure. For this kind of work the designer should respect the author's wishes as far as possible within a maximum text width.

3.5.1 Poetry

If possible the type and measure should be chosen so that the longest poetical line will fit on one printed line, so that the shape of the poem is retained.

Poems in a book of poetry will differ from one another in their width, and the best way of setting these is to optically center each poem on the page. However, blank verse and poems where the majority of the lines are long are usually indented by a constant amount from the left margin.

In some context verse lines are numbered, often every fifth or tenth line. The numbers are usually small and right justified.

3.5.2 Plays

When presenting a play a list of characters (*Dramatis Personae*) is frequently given at the beginning of the play. It is presented between the title and the start of the play itself, either on the same page as the title, or on a page by itself, or at the top of the first page of the play. The list may be ordered alphabetically, in order of appearance, or by the character's importance. A remark about a character, if less than a sentence, follows the name, separated by a comma. If the remark is one or several sentences they are set as usual.

Act and scene names and numbers are often treated in the same manner as subheads in a prose work. A new act does not necessarily start a new page but there should be at least twelve points above and six below the number. A new scene has about eight points above and six points below the number. Either arabic or roman numerals may be used for the numbers. If roman is used for both, then uppercase for acts and lowercase for scenes.

The name of each speaker in a play must be readily identifiable and stand apart from the speech. Names are commonly set in a different font, such as small caps or italic, to the text which is usually set in roman. They may be placed on a separate line, where they are most easily identifiable, or, to save space, in the margin. The names are often abbreviated, and if so the abbreviations must be consistent throughout the work.

Stage directions have to be differentiated from the text. They are usually set in italics and enclosed in brackets, or less often, in parentheses; speakers' names in stage directions,

though, are set in roman to distinguish themselves. Directions at the start of a scene, such as saying who is entering, are typically centered while in the body of the scene are set flush right, often on a line by themselves.

Four

Picky points

The main elements of good typography are legibility and page color. This chapter discusses some of the smaller points related to these topics.

4.1 WORD AND LINE SPACING

Research has shown that the competent reader recognises words by their overall shape rather than by stringing together the individual letters forming the words. A surprisingly narrow gap between words is sufficient for most to distinguish the word boundaries.

Most typographers state that the space between words in continuous text should be about the width of the letter ‘i’. Any closer and the words run together and too far apart the page looks speckled with white spots and the eye finds it difficult to move along the line rather than jumping to the next word in the next line. Figure 4.1 illustrates different values of interword spacing.

In keeping with avoiding white spots, many typographers do not recommend extra spacing after punctuation, although this does depend partly on a country’s typographic history and partly on the individual. I always found typewritten texts with double spaces after the end of sentences a particular eyesore. However, with typeset texts any extra spacing is usually not as large as that.

The spacing between lines of text should be greater than the interword spacing, otherwise there is a tendency for the eye to skip to the next line instead of the next word. The space in question is the apparent amount of whitespace between the bottom of the text on one line and the top of the text on the next line. In a rough sense it is the leading minus the actual height of the font. Figure 4.2 illustrates some text typeset with different line spacings. The normal interword spacing is used in the samples. When text is set solid there is a tendency for the descenders on one line to touch, or even overlap, the ascenders on the following line.¹

4.2 ABBREVIATIONS AND ACRONYMS

The English style with abbreviations is to put a full stop (period) after the abbreviation, unless the abbreviation ends with the same letter as the full word. Thus, it is Mr for Mister, Dr for Doctor, but Prof. for Professor. No extra spacing should be used after the full stop, even if extra spacing is normally used after punctuation.

The general American, and English, trend nowadays is away from the use of periods (full stops) after abbreviations following the precept that reducing typographic fussiness

¹TeX has a built-in mechanism that tries hard to prevent this from happening.

The following paragraph is typeset with double the normal interword spacing for this font.

Most typographers state that the space between words in continuous text should be about the width of the letter ‘i’. Any closer and the words run together and too far apart the page looks speckled with white spots and the eye finds it difficult to move along the line rather than jumping to the next word in the next line. Extra spacing after punctuation is not necessary.

The following paragraph is typeset with the normal interword spacing for this font.

Most typographers state that the space between words in continuous text should be about the width of the letter ‘i’. Any closer and the words run together and too far apart the page looks speckled with white spots and the eye finds it difficult to move along the line rather than jumping to the next word in the next line. Extra spacing after punctuation is not necessary.

The interword spacing in the following paragraph is three-quarters of the width of the letter ‘i’.

Most typographers state that the space between words in continuous text should be about the width of the letter ‘i’. Any closer and the words run together and too far apart the page looks speckled with white spots and the eye finds it difficult to move along the line rather than jumping to the next word in the next line. Extra spacing after punctuation is not necessary.

Figure 4.1: Interword spacings

increases the ease of reading. Having said that, where an abbreviation is a combination of abbreviations, such as Lt.Col for Lieutenant Colonel, often an internal period is used with a word space between the elements.

Acronyms are typeset in uppercase but the question is, which uppercase? The simple way is to use the uppercase of the normal font, like UNICEF, but if there are too many acronyms scattered around the speckled effect starts to intrude. If the font family has one, then small caps can be used, giving UNICEF. If small caps are not available, or appear undesirable, then a smaller size of the normal uppercase can be used, such as UNICEF or UNICEF; some experimentation may be required to select the appropriate size. These several versions were input as:

regular UNICEF text	regular UNICEF text
regular UNICEF text	regular \textsc{unicef} text
regular UNICEF text	regular {\small UNICEF} text
regular UNICEF text	regular {\footnotesize UNICEF} text

This and the next paragraph are set solid — the interline spacing is the same as the font size.

The normal interword spacing is used in these paragraphs. The spacing between lines of text should be greater than the interword spacing, otherwise there is a tendency for the eye to skip to the next line instead of the next word.

This and the next paragraph are set with the normal interline spacing for the font.

The normal interword spacing is used in these paragraphs. The spacing between lines of text should be greater than the interword spacing, otherwise there is a tendency for the eye to skip to the next line instead of the next word.

This and the next paragraph are set with the interline spacing 20% greater than is normal for the font.

The normal interword spacing is used in these paragraphs. The spacing between lines of text should be greater than the interword spacing, otherwise there is a tendency for the eye to skip to the next line instead of the next word.

Figure 4.2: Interline spacings

4.3 DASHES AND ELLIPSES

Most fonts provide at least three lengths of dashes. The shortest is the hyphen (-), then there is the en-dash (–) which is approximately the width of the letter ‘n’, and the largest is the em-dash (—) which is approximately twice the length of an en-dash. An expert font may provide more.

Unsurprisingly, the hyphen is used for hyphenation, such as in *em-dash*, or at the end of a line where a word had to be broken.

The en-dash is normally used between numerals to indicate a range. For example a reference may refer to pages 21–27 in some journal or book. There is no space surrounding the en-dash when used in this manner.

The em-dash, or the en-dash, is used as punctuation — often when making a side remark — as a phrase separator. When en-dashes are used as punctuation it is normal to put spaces around them but the question of spaces around an em-dash appears to be the subject of some contention. Roughly half the participants in any discussion advocate spaces while the other half view them as anathema. If you do use em-dashes be sure to be consistent in your use, or otherwise, of spaces.

Ellipses are those three, or is it four, dots indicating something is missing or continues somewhat indefinitely. In the middle of a sentence, or clause or ... they have a space on either side. At the end of a sentence the English style is to have no spaces and include the full stop, making four dots in all, like so. . . .

Dashes are also used to indicate missing characters or a word. Missing characters in the middle of a word are indicated by a 2em-dash (a dash that is twice as long as an em-dash), as in:

snafu, (*U.S. slang*) *n.* chaos. — *adj.* chaotic. [*situation normal — all f—d up.*]

<p>'There's glory for you!'</p> <p>'I don't know what you mean by "glory"', Alice said.</p> <p>Humpty Dumpty smiled contemptuously. 'Of course you don't — till I tell you. I meant "there's a nice knock-down argument for you!"'</p> <p>'But "glory" doesn't mean "a nice knock-down argument"', Alice objected.</p> <p>'When <i>I</i> use a word', Humpty Dumpty said, in a rather scornful tone, 'it means just what I choose it to mean — neither more nor less'.</p>	
<p>"There's glory for you!"</p> <p>"I don't know what you mean by 'glory,'" Alice said.</p> <p>Humpty Dumpty smiled contemptuously. "Of course you don't — till I tell you. I meant 'there's a nice knock-down argument for you!'"</p> <p>"But 'glory' doesn't mean 'a nice knock-down argument,'" Alice objected.</p> <p>"When <i>I</i> use a word," Humpty Dumpty said, in a rather scornful tone, "it means just what I choose it to mean — neither more nor less."</p>	

Figure 4.3: Quotation marks: top English, bottom American

A 3em-dash is used to indicate a missing word. When I lived in Maryland my local small town newspaper was the *Frederick Post*. The following is from an obituary I happened to read; I have hidden the name to protect the innocent.

Although he had spent the last 92 years of his life here, Mr ——— was not a Fredericktonian.

4.4 PUNCTUATION

4.4.1 Quotation marks

Quotation marks surrounding speech and associated punctuation are a fruitful source of confusion.

The American style is to use double quotes at the start(") and end(") of spoken words. If the speaker quotes in the speech then single quote marks (' and ') are used to delineate the internal quotation.

The English practice is exactly the opposite. Main speech is delineated by single quotes and internal quotations by double quotes. In any event, if single and double quotes are adjacent they should be separated by a thin space in order to distinguish one from the other — a full interword space is too wide.

As there are likely to be few internal quotations it seems to me that the English practice produces a less spotty appearance than the American. Figure 4.3 shows the same text typeset in both the English and American styles. The example is from Lewis Carroll's *Through the Looking Glass and what Alice Found There* and has an unusually large number of internal quotations.

Where to put punctuation marks with quotes is vexatious. Again the English and American practice tends to differ. The American tendency is to put commas and periods inside the closing quote mark and colons and semicolons after the mark. English editors prefer to put punctuation after the mark. In either case, it is difficult to know exactly what

to do. I get the impression that for every example of the ‘correct’ form there is a counter-example. Some try and avoid the problem altogether by putting the lower marks, like commas or periods, directly below the quotation mark but that may cause problems if the resulting constructs look like question or exclamation marks. In Figure 4.3 I have tried to use the English and American punctuation styles in the respective examples but it is likely that there are misplacements in both. I think it’s basically a question of doing what you think best conveys the sense, provided there is consistency.

4.4.2 Footnote marks

Where to put a footnote marker may be another vexed question in spite of the general principal being easy to state: The mark goes immediately after the text element that the note refers to.

There is no doubt what this means² when the text element is a word in the middle of other words. Doubt raises its head when the reference is to a phrase, like this one³, which is set off within commas, or when the note refers to a complete sentence.⁴

Like punctuation and quotation marks, should a footnote mark come before or after the punctuation mark at the end of a phrase or a sentence? I have shown both positions⁵ in the previous paragraph. The general rule that I have deduced is that the mark comes after the punctuation, but there are always those who like to prove a rule.

There are other marks that may be associated with a word, like (registered) trademarks. These may produce ugly gaps. Sometimes these cannot be avoided but it may be possible to change the text to minimise the hiccup. There is an example of this on page xxxiv. I tried various schemes in identifying ‘PostScript’ as being a registered trademark of Adobe Systems Incorporated. Among the discarded trials were:

...languages like PostScriptTM, presumably ...
 ...languages like PostScript[®], presumably ...
 ...like the PostScript[®] language, presumably ...

My final solution was to note the registered trademark information in a footnote:

...languages, like PostScript⁶, presumably ...

In this case I decided that the footnote was really tied to the word ‘PostScript’, taking the place of the registered symbol, so I put the footnote mark before the comma rather than after it.

4.4.3 Font changes

Sometimes a word or two may be set in a different font from the surrounding text, such as when emphasizing a word by setting it in an italic font. If the word is followed by a punctuation mark the normal practice is to set the punctuation mark using the new font instead of the normal font. In some cases the font used for the punctuation may not be particularly noticeable but sometimes it may be.

The front matter contains definitions of the word *memoir*, which is typeset using a bold font. The definitions thus commence like

memoir, *n.* ...

²Except to some I know.

³I hope that this is a phrase.

⁴Is this mark in the correct place?

⁵Marks 3 and 4.

⁶PostScript is a registered trademark of Adobe Systems Incorporated.

No hyphenation	Hyphenation (1)	Hyphenation (2)
<p>The LaTeX document preparation system is a special version of Donald Knuth's TeX program. TeX is a sophisticated program designed to produce high-quality typesetting, especially for mathematical works. It is extremely flexible albeit somewhat idiosyncratic. One can typeset justified, flushleft-raggedright, centered, or raggedleft-flushright.</p>	<p>The LaTeX document preparation system is a special version of Donald Knuth's TeX program. TeX is a sophisticated program designed to produce high-quality typesetting, especially for mathematical works. It is extremely flexible albeit somewhat idiosyncratic. One can typeset justified, flushleft-raggedright, centered, or raggedleft-flushright.</p>	<p>The LaTeX document preparation system is a special version of Donald Knuth's TeX program. TeX is a sophisticated program designed to produce high-quality typesetting, especially for mathematical works. It is extremely flexible albeit somewhat idiosyncratic. One can typeset justified, flushleft-raggedright, centered, or raggedleft-flushright.</p>

Figure 4.4: Raggedright text in narrow columns

instead of

memoir, *n.* ...

4.5 NARROW MEASURES

Typesetting in a narrow column is difficult, especially if you are trying to make the text flush left and right. As the lines get shorter it becomes more and more difficult to fit the words in without an excessive amount of interword spacing or word breaking at the ends of lines. In the limit, of course, there will not be even enough room to put a syllable on a line.

The best recourse in situations like this is to forget justification and typeset raggedright. Raggedright looks far better than justified text with lots of holes in it. The question then is, to hyphenate or not to hyphenate?

With no hyphenation there is likely to be increased raggedness at the line ends when compared with permitting some hyphenation. Hyphenation can be used to reduce the raggedness but somehow short lines ending with a hyphen may look a bit odd. This is where you have to exercise your judgement and design skills.

Figure 4.4 shows a text set in a column 107.0pt wide with different raggednesses. Preventing hyphenation, as in the left column, resulted in very noticeably ragged text. Hyphenation has been allowed in the other two columns, to differing degrees. I prefer the one on the right but with different text and column widths the results might have been different.

Indexes are often typeset in double, or even triple or quadruple columns, as each entry is typically short. Also, indexes are typically consulted for a particular entry rather than

being read as continuous text. To help the eye, page numbers are normally typeset immediately after the the name of the indexed topic, so indexes tend to be naturally raggedright as a matter of reader convenience.

Talking of hyphenation, each language has its own rules for allowable hyphenation points. As you might now have come to suspect, English and American rules are different even though the language is nominally the same. Broadly speaking, American English hyphenation points are typically based on the sound of the word, so the acceptable locations are between syllables. In British English the hyphenation points tend to be related to the etymology of the word, so there may be different locations depending on whether the word came from the Greek or the Latin. If you are not sure how a particular word should be hyphenated, look it up in a dictionary that indicates the potential break points.

4.6 EMPHASIS

Underlining should emphatically not be used to emphasise something in a typeset document. This is a hangover from the days when manuscripts were typewritten and there was little that could be done. The other way of emphasising something was to put extra space between the characters of the word being emphasised, as has been done twice in this sentence (for the words ‘word’ and ‘emphasised’ in case you didn’t spot them). This is called *letterspacing*; it is more often used to make fine adjustments to the physical spacing between letters in a book title in order to make them appear to be optically uniformly spaced. As an aside, for me at least, that extra spacing just now produces the illusion that the characters are slightly larger than normal, which is not the case.

With the range of fonts and sizes available when typesetting there are other methods for emphasis, although German typographers have used letterspacing for emphasis with the fraktur and other similar font types.

There are basically three approaches: change the **size** of the font; change the **weight** of the font; or most usually, change the *shape* of the font. There is a creative tension when trying to emphasise something — there is the need to show the reader the emphasised element, but there is also the desire not to interrupt the general flow of the text. Out of the three basic options, changing the shape seems to be a reasonable compromise between the need and the desire.

4.7 CAPTIONS AND LEGENDS

I am not entirely sure what is the difference between a caption and a legend as both terms refer to the title of an illustration or table. However, legend may also be used to refer to some explanatory material within an illustration, such as the explanation of the symbols used on a map.

In any event, captions and legends are usually typeset in a font that is smaller than the main text font, and which may also be different from the main font. For example, if the main font is roman and a sans font is used for chapter titles, then it could be appropriate to use a small size of the sans font for captions as well.

The caption for a table is normally placed above the table while captions for illustrations are placed below.

4.8 TABLES

A table is text or numbers arranged in columns, and nearly always with a ‘legend’ above each column describing the meaning of the entries in the column. The legends and the column entries are separated from each other, perhaps by some vertical space but more often by a horizontal line.

In general typographers dislike vertical lines in a table, which may be likely to be used to separate the columns. I’m not sure why this is. There is an obvious explanation when hand setting the individual characters as, although it would be easy to set horizontal rules, it would be very difficult to get all the pieces of type with the bits of the vertical rules aligned properly — the eye is very sensitive to jags in what is meant to be a straight line. In the days of digital typography the alignment problem has gone away, so perhaps the antipathy to vertical lines is a tradition from earlier days. On the other hand Edward Tufte [Tuf83, p. 96] exhorts us to ‘Maximize the data-ink ratio’ and to ‘Erase non-data-ink’ and Bringhurst [Bri99, p. 70] says ‘There should be a minimum amount of furniture (rules, boxes, dots and other guiderails for travelling through typographic space) and a maximum amount of information’.

If you want to use vertical lines, just be aware that not everybody may appreciate your effort.

4.9 NUMBER FORMATTING

Number formatting is country- and language-dependent. Continental Europe differs from England, and in its turn the United States differs from England.

Ignoring decimal numbers we have *cardinal* and *ordinal* numbers. An ordinal number, like 3rd, indicates a position in a sequence, while a cardinal number, like 3, expresses ‘how many’. LaTeX typesets numbers as cardinals, and these can be displayed as a sequence of arabic digits or as upper- or lower-case Roman numerals.

In general text the tradition seems to be that cardinal numbers between one and ten are spelled out rather than being presented as numerals. In the United States numbers between one and ninety-nine are spelled (with twenty-one, twenty-two, etc., being hyphenated). Similar customs apply to ordinal numbers such as eighth, twenty-first. When not spelled out ordinals should be set like 378th rather than 378th.

Regarding cardinal numbers represented as arabic digits, some cultures prefer these to be presented as an unbroken string of digits (e.g. 12345). Other societies prefer the digits in longer numbers to be separated, in some cases by commas (e.g., 12,345) or other punctuation marks (e.g., 12.345), and in others by small spaces (e.g., 12 345); the digits are grouped into threes, counting from the right.

When the arabic digits became generally used they, like the letters, were given both uppercase and lowercase forms. The uppercase form, like these 1 2 3 4 5 6 7 8 9 0, which is the one normally supplied as part of a font are called *titling figures*, *ranging figures*, or *lining figures* because they range or align with the uppercase. Digits in this class all have the same width so they are used in tables where numbers are meant to be aligned vertically. They are also used when typesetting mathematics.

The lowercase form, like 1 2 3 4 5 6 7 8 9 0, are called *text figures*, *hanging figures*, *lowercase figures*, or *old-style figures*. These may be used whenever the surrounding text is

set in mixed case, or small caps; I have seen them used typesetting the folios, but I must admit that they look very odd to me in that situation.

If you are typesetting mathematics, where you use lining figures, and are also using old-style figures in the text then be very careful; ‘mathematical numbers’ should always be set with lining figures even if they are in the body of the text. For example:

... from the equation the result is 42 ...

... the men of the 42nd Foot performed magnificently

Part II

Practice

Five

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.

5.1 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 5.1 through Table 5.3. 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.

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

Table 5.1: Class stock metric paper size options, and commands

Option	Size	stock size command	page size command
<code>a6paper</code>	148×105 mm	<code>\stockavi</code>	<code>\pageavi</code>
<code>a5paper</code>	210×148 mm	<code>\stockav</code>	<code>\pageav</code>
<code>a4paper</code>	297×210 mm	<code>\stockaiv</code>	<code>\pageaiv</code>
<code>a3paper</code>	420×297 mm	<code>\stockaiii</code>	<code>\pageaiii</code>
<code>b6paper</code>	176×125 mm	<code>\stockbvi</code>	<code>\pagebvi</code>
<code>b5paper</code>	250×176 mm	<code>\stockbv</code>	<code>\pagebv</code>
<code>b4paper</code>	353×250 mm	<code>\stockbiv</code>	<code>\pagebiv</code>
<code>b3paper</code>	500×353 mm	<code>\stockbi</code>	<code>\pagebi</code>
<code>mcrownvopaper</code>	186×123 mm	<code>\stockmetriccrownvo</code>	<code>\pagemetriccrownvo</code>
<code>mlargecrownvopaper</code>	198×129 mm	<code>\stockmlargecrownvo</code>	<code>\pagemlargecrownvo</code>
<code>mdemyvopaper</code>	216×138 mm	<code>\stockmdemyvo</code>	<code>\pagemdemyvo</code>
<code>msmallroyalvopaper</code>	234×156 mm	<code>\stockmsmallroyalvo</code>	<code>\pagemsmallroyalvo</code>

Table 5.2: 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 5.3: 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	\pagedmediumvo
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

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

5.2 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
- 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

extrafont sizes Using scalable fonts that can exceed 25pt.

These options, except for extrafont sizes, 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 extrafont sizes 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).

5.2.1 Extended font sizes

By default, if you use the extrafont sizes 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 document's preamble (but with the extrafont sizes 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{\lrm}  

\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

```
\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.¹ Internally, the class specifies

¹If 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.

```

\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.

5.3 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.
- `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`.

5.4 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 succeeding 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. 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. None of these options are defaulted.

5.5 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 you 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 affect 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.

Six

Laying out the page

Up until this chapter the *headings* pagestyle has been used; pagestyles are described in §11.2. This, and later chapters, are typeset with the *ruled* pagestyle.

6.1 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].

6.2 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.

6. LAYING OUT THE PAGE

The circle is at 1 inch from the top and left of the page. Dashed lines represent ($\backslash\text{offset} + 1$ inch) and ($\backslash\text{voffset} + 1$ inch) from the top and left of the page.

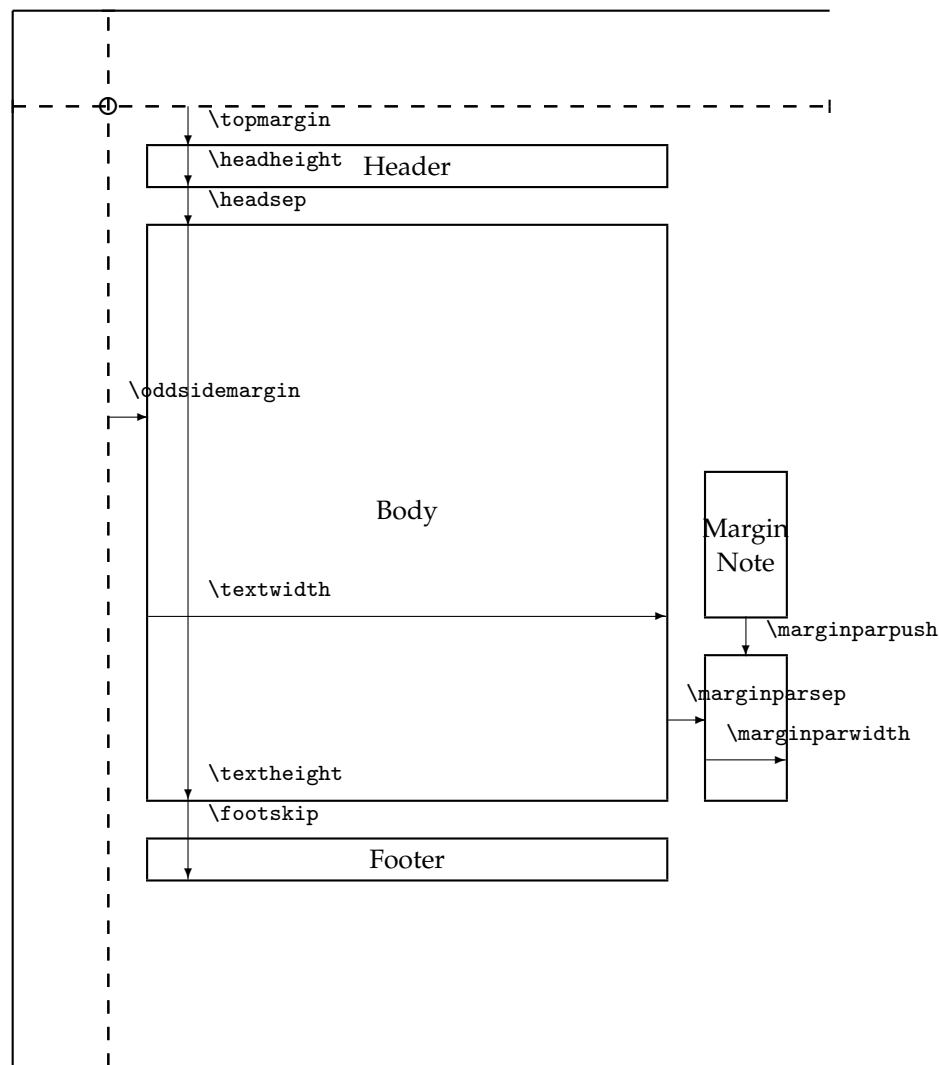


Figure 6.1: LaTeX page layout parameters for a recto page

6.3 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 6.1. 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 6.3 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 6.2 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. 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.

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 6.1.

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

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

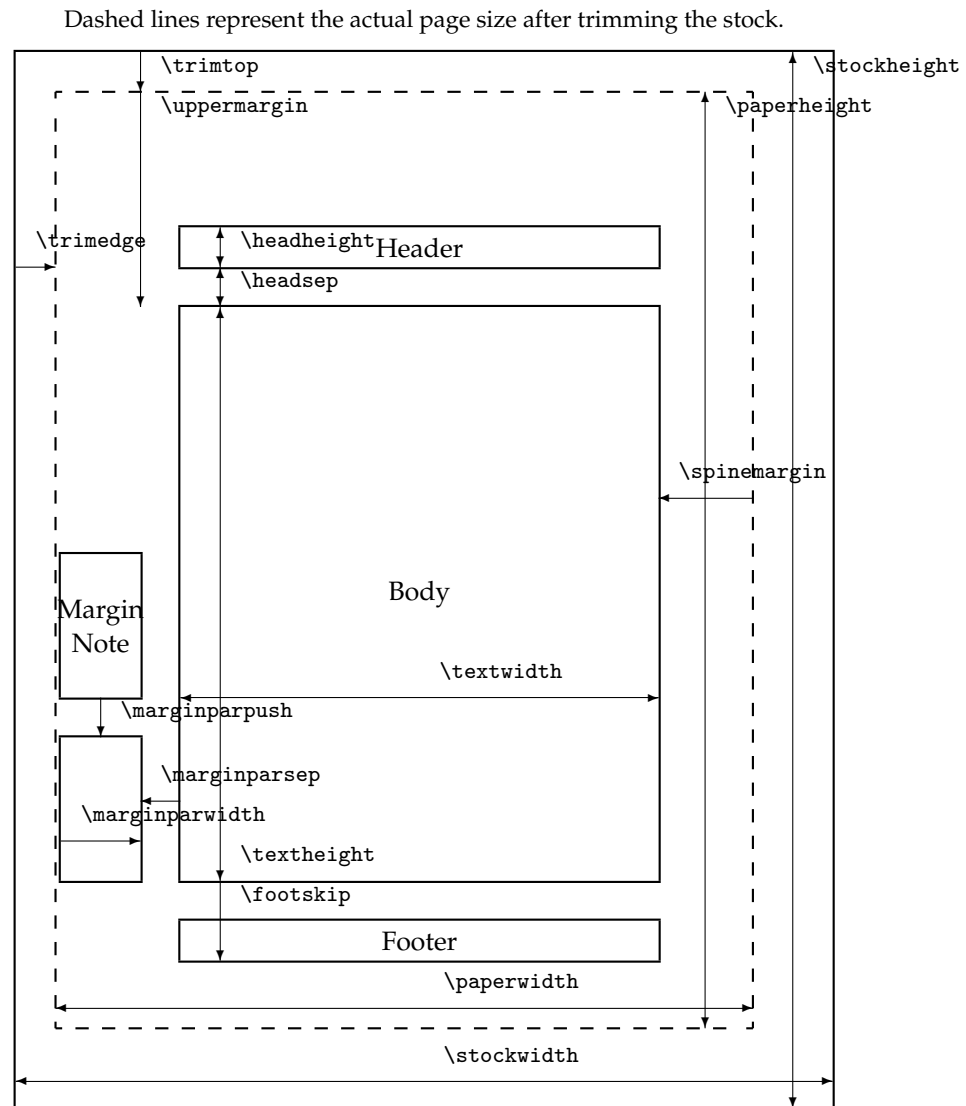


Figure 6.2: The memoir class page layout parameters for a verso page

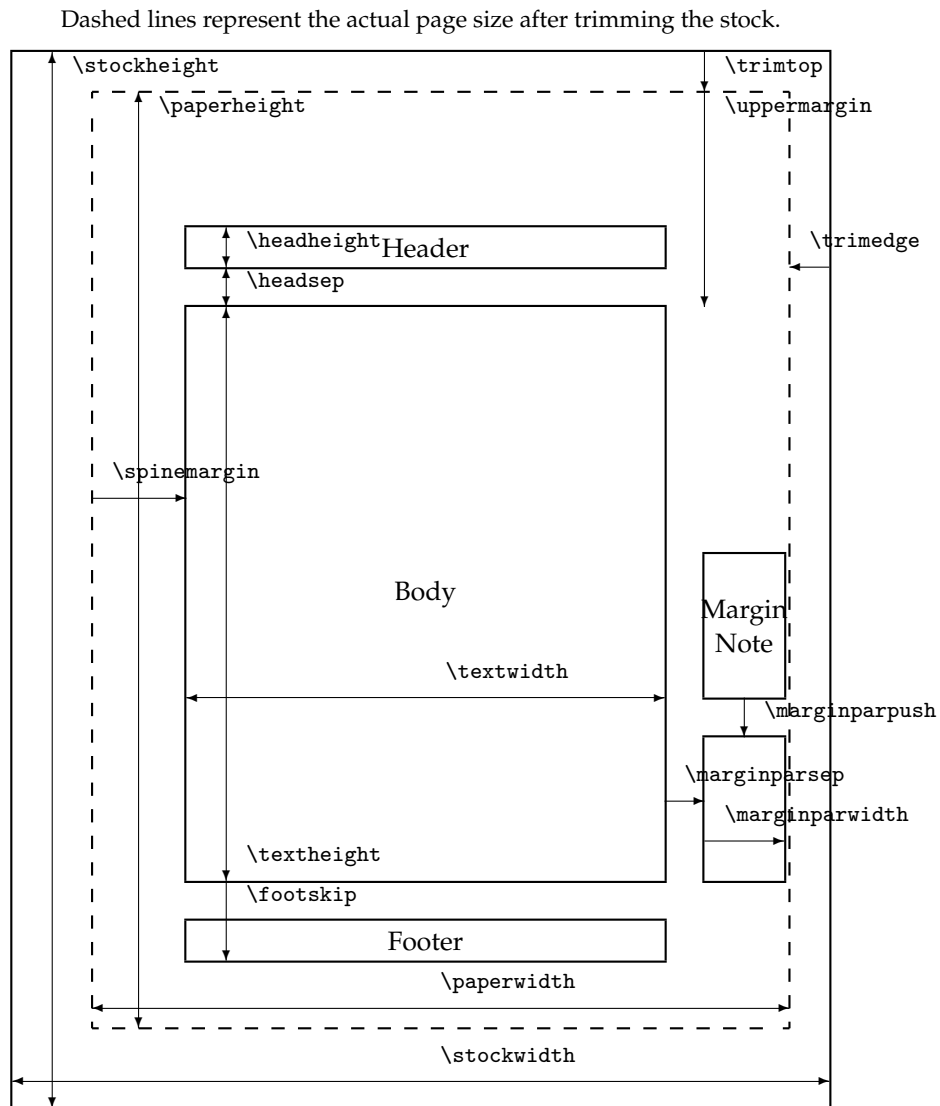


Figure 6.3: The memoir class page layout parameters for a recto page

Table 6.1: Arguments and results for `\settrimmedsize` and `\settypeblocksize`

$\langle height \rangle$	$\langle width \rangle$	$\langle ratio \rangle$	Result
H	W	r	ambiguous
H	W	*	H, W
H	*	r	$W = rH$
H	*	*	ambiguous
*	W	r	$H = rW$
*	W	*	ambiguous
*	*	r	ambiguous
*	*	*	ambiguous

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>}{<foreedge>}
```

The command `\settrims` can be used to specify the amount intended to be removed from the top (*<top>*) and fore-edge (*<foreedge>*) 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 yourself 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}
```

6.4 THE TYPEBLOCK

Like the page, the typeblock is normally rectangular with the height greater than the width.

Table 6.2 gives the lowercase alphabet lengths for some typefaces over a range font sizes; this may be used in conjunction with Table 2.2 on page 33 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, 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. Brighurst [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 6.2, 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` respectively for the font *<fontspec>*.

Table 6.2: Lowercase alphabet lengths in points

	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	108	122	136	149	162	194	234	281
Palatino	107	120	133	146	160	192	230	276
Times	96	108	120	131	143	172	206	247
Utopia	107	120	107	146	146	193	232	277
Avantgard	113	127	142	155	169	193	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 = 190.1926pt`.

Morten Høgholm¹ 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 inadvisable to put any commands before `\documentclass`.

The height of the typeblock should be equivalent to an integral number of lines.

`\settypeblocksize{\height}{\width}{\ratio}`

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 6.1 on page 74. For instance, here are three ways of specifying a 6in by 3in typeblock:

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

¹Private communication

Table 6.3: Arguments and results for `\setlrmargins`

$\langle spine \rangle$	$\langle edge \rangle$	$\langle ratio \rangle$	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	$E + S = K_w, E = rS$
*	*	*	$E + S = K_w, E = S$

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.

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{ $\langle spine \rangle$ }{ $\langle edge \rangle$ }{ $\langle ratio \rangle$ }`

The command `\setlrmargins` can be used to specify the side margins² with the width of the page and the typeblock being fixed.

Not more than 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 6.3.

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.

²Only the spine margin is noted in Figure 6.3 and 6.2; the fore-edge margin is at the opposite side of the typeblock.

Table 6.4: Arguments and results for `\setlrmarginsandblock`

$\langle spine \rangle$	$\langle edge \rangle$	$\langle ratio \rangle$	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

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 is half as big again as the spine margin) this can be accomplished in three ways (with the `\pagewidth` 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{\langle spine \rangle}{\langle edge \rangle}{\langle ratio \rangle}`

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 6.4. 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 `\pagewidth` 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}{*}
```

Table 6.5: Arguments and results for `\setulmargins`

$\langle upper \rangle$	$\langle lower \rangle$	$\langle ratio \rangle$	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	$L + U = K_h, L = rU$
*	*	*	$L + U = K_h, L = U$

```
% 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}{*}
```

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

`\setulmargins{ $\langle upper \rangle$ }{ $\langle lower \rangle$ }{ $\langle ratio \rangle$ }`

The command `\setulmargins` can be used to specify the upper and lower margins³ 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 6.5. The `\setlrmargins` command maintains the relationship

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

`\setulmarginsandblock{ $\langle upper \rangle$ }{ $\langle lower \rangle$ }{ $\langle ratio \rangle$ }`

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 6.6. 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 $B_h = P_h - U - L$.

³Only the upper margin is noted in Figure 6.3 and 6.2; the lower margin is the distance between the bottom of the typeblock and the bottom of the page.

Table 6.6: Arguments and results for `\setulmarginsandblock`

$\langle upper \rangle$	$\langle lower \rangle$	$\langle ratio \rangle$	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

`\setcolsepandrule{\langle colsep \rangle}{\langle thickness \rangle}`

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 $\langle colsep \rangle$ and the thickness of the rule, `\columnseprule`, to $\langle thickness \rangle$. A $\langle thickness \rangle$ 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.

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.

6.5 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 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{\langle headheight \rangle}{\langle footskip \rangle}`

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

`\setheaderspaces{\langle headdrop \rangle}{\langle headsep \rangle}{\langle ratio \rangle}`

Table 6.7: Arguments and results for `\setheaderspaces`

$\langle\text{headdrop}\rangle$	$\langle\text{headsep}\rangle$	$\langle\text{ratio}\rangle$	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	$H_s + D = C_h, H_s = rD$
*	*	*	$H_s + D = C_h, H_s = D$

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 header⁴, 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 6.7. 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 6.2 shows the marginal note parameters on a verso page, and also illustrates that some parameters control different positions on the stock.

`\setmarginnotes{ $\langle\text{separation}\rangle$ }{ $\langle\text{width}\rangle$ }{ $\langle\text{push}\rangle$ }`

The command `\setmarginnotes` sets the parameters for marginal notes. The distance `\marginparsep` between the typeblock and any note is set to $\langle\text{separation}\rangle$, the maximum width for a note, `\marginparwidth`, is set to $\langle\text{width}\rangle$ and the minimum vertical distance between notes, `\marginparpush`, is set to $\langle\text{push}\rangle$.

6.6 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 6.1. 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.

⁴The head drop is not shown in Figure 6.3 or 6.2.

- Use `\setstocksize` to set the values of `\stockheight` and `\stockwidth`, followed by `\settrimmedsize` to specify the values of `\paperheight` and `\paperwidth`.
- 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`.
- Decide on the size of the typeblock and use `\settypeblocksize` to specify the values of `\textheight` and `\textwidth`.
- 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 6.3 and 6.1 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 6.8.

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.

<pre>\checkandfixthelayout[<i><algorithm></i>] \checkthelayout[<i><algorithm></i>] \fixthelayout \baselineskip \topskip</pre>

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

Table 6.8: The class and LaTeX page layout parameters

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

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

$$T = (N - 1)b + t \quad (6.1)$$

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.⁵ The permissible values for `\algorithm` are:

fixed The `\textheight` is not altered.

$$T = H \quad (6.2)$$

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

⁵In 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 6.9: Results from sample `\textheight` adjustments

Requested height	Algorithm			
	fixed	classic	lines	nearest
	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)

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

$$T = b\lfloor H/b \rfloor + t \quad (6.3)$$

The relationship (6.1) is maintained.

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

$$T = b\lfloor (H - b)/b \rfloor + t \quad (6.4)$$

The relationship (6.1) is maintained.

`nearest` The calculated value is the nearest to the given value while still maintaining the relationship (6.1).

$$T = b\lfloor (H - t + b/2)/b \rfloor + t \quad (6.5)$$

Table 6.9 shows the results from the various `\textheight` adjustment calculations⁶ 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 and calculates the values for all the standard LaTeX layout parameters. 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.

```
\typeoutlayout
\typeoutstandardlayout
```

⁶For comparison the optimum heights from equation 6.1 for 10, 11 and 12 lines are respectively 118pt, 130pt and 142pt.

`\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.

6.7 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...
...
```

6.8 PRINTING AND VIEWING

When pdfLaTeX is used to generate a PDF version of a memoir document some special setup must be done.

```
\fixpdflayout
```

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

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

The first settings (`\pdfpage...`) ensure that pdfLaTeX knows the size of the physical sheet for printing. The `\...origin` settings set the PDF origin per the TeX origin, provided that their values are 0pt. If you have set the origin values yourself, either in a pdfLaTeX configuration file or earlier in the preamble, the `\fixpdflayout` macro will not alter them (if you need an origin value to be 0 then set it to 1sp, which is visually indistinguishable from 0pt).

```
\fixdvipslayout
```

The macro `\fixdvipslayout` is automatically called after the preamble when PDF output is not being produced. It provides the dvips processor with information about the stock size, which a viewer or printer may use.

With a landscape document and using the processing route `latex -> dvips` the resulting ps PostScript file may appear upside-down when viewed with, say, `ghostview` (also known as `gsview32`). If this happens try putting the following in your preamble:

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

If required, additional code can be added to `\fixdvipslayout` by repeated applications of `\addtodef`. Some other potential specials for PostScript printing may be⁷

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

6.9 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, and 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 `\trimedged` are easily specified by

```
\setlength{\trimtop}{0pt}  
\setlength{\trimedged}{\stockwidth}  
\addtolength{\trimedged}{-\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}
```

⁷At least for an HP 5SiMx LaserJet duplex printer.

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
```

6.9.1 The page layout of this manual

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

```
\settrimmedsize{11in}{210mm}{*}
\setlength{\trimtop}{0pt}
\setlength{\trimedged}{\stockwidth}
\addtolength{\trimedged}{-\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 6.4 which also lists the parameter values resulting from the code above, to the nearest point.

I initially used the layout defined in §6.9, 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.

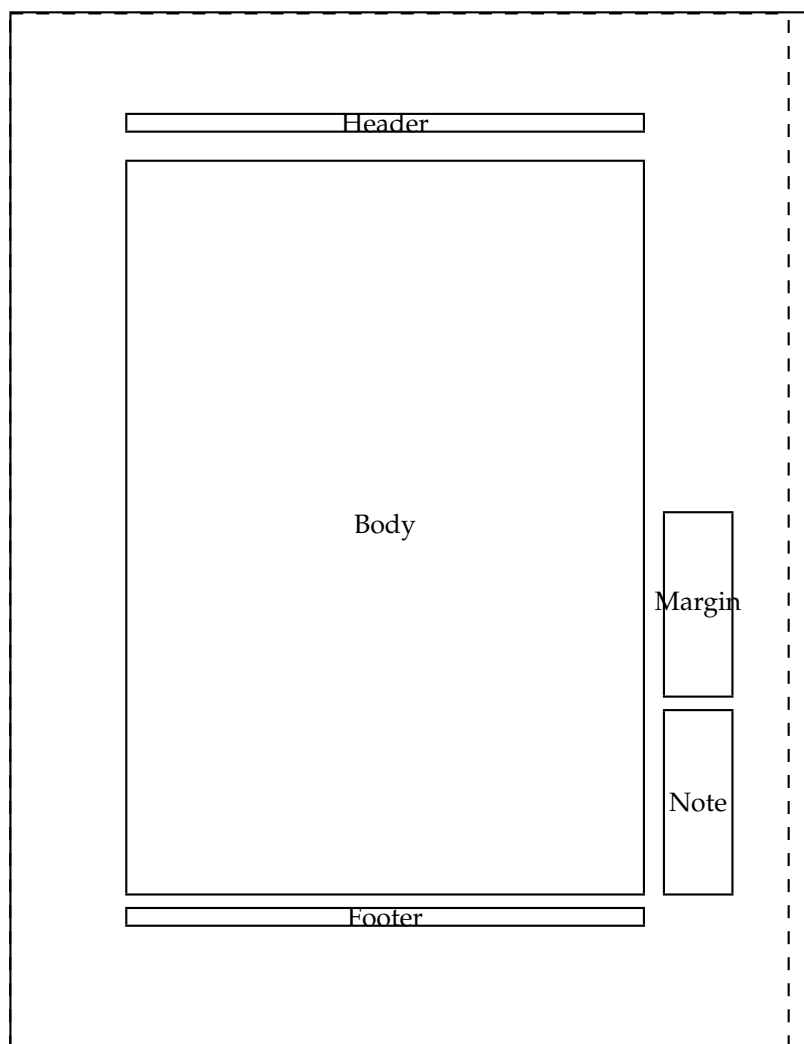
Andreas Mathias, via Rolf Niepraschk,⁸ has suggested that the following might be better for typesetting the manual on A4 paper.

```
\documentclass[a4paper]{memoir}
...
\settrimmedsize{297mm}{210mm}{*}
\setlength{\trimtop}{0pt}
\setlength{\trimedged}{\stockwidth}
\addtolength{\trimedged}{-\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.

⁸Email from niepraschk@ptb.de on 2002/02/05.

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



Lengths are to the nearest pt.

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

Figure 6.4: The recto page layout for this manual

6.10 PREDEFINED LAYOUTS

The class, like the standard classes, will automatically produce working layouts for letter-paper and a4paper size options. They might be a bit problematic, though, when the page is much smaller, particularly with respect to the space allotted for marginal notes. You perhaps will find the layouts package [Wil03a] 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).

```
\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]; some details about this were given earlier in §2.2.1. 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.

```
\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 6.5 through 6.12. These were produced with the aid of the layouts package.

```
\setpagebl{⟨height⟩}{⟨width⟩}{⟨ratio⟩}
\setpageml{⟨height⟩}{⟨width⟩}{⟨ratio⟩}
\setpagetl{⟨height⟩}{⟨width⟩}{⟨ratio⟩}
```

When your page is smaller than the stock it has to be positioned on the stock by specifying the trims to give the desired size and location. The macro `\setpagebl`, which takes the same arguments as `\settrimmedsize` (see Table 6.1 on page 74), calculates the

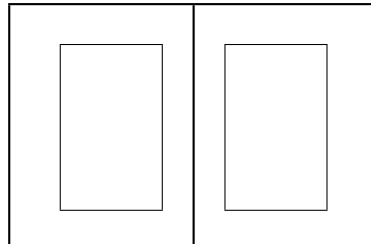


Figure 6.5: Default layout for letterpaper



Figure 6.6: Letterpaper layout: Left `\medievalpage`, Right `\medievalpage[12]`



Figure 6.7: Letterpaper layout: Left `\isopage`, Right `\isopage[12]`



Figure 6.8: Letterpaper layout: Left `\semiisopage`, Right `\semiisopage[12]`

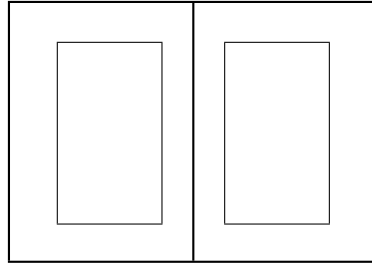


Figure 6.9: Default layout for a4paper



Figure 6.10: A4paper layout: Left `\medievalpage`, Right `\medievalpage[12]`



Figure 6.11: A4paper layout: Left `\isopage`, Right `\isopage[12]`



Figure 6.12: A4paper layout: Left `\semiisopage`, Right `\semiisopage[12]`

trims so that the page is located at the bottom left of the stock. Similarly `\setpageml` and `\setpagetl` will locate the page at the middle left and the top left, respectively, of the stock. 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.

```
\pagebv % sets page height and width for A5 paper
\setpagebl{\pageheight}{\pagewidth}{*}
...
```

```
\checkandfixthelayout
```

The 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. To reposition the page to the center of the stock the following will halve the top and edge trims.

```
\settrims{0.5\trimtop}{0.5\trimedg}
...
```

```
\checkandfixthelayout
```

Remember that after you have finished defining the layout you want you have to call `\checkandfixthelayout` for all the changes to take effect.

Seven

Text and fonts

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

7.1 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, created by many others, available 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 a little bit of work, 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]. There is less detailed, but 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.

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 7.1 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`.

<code>\normalfont</code>

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

There is a set of font declarations, as shown in Table 7.2, that correspond to the commands listed in Table 7.1. 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.

Table 7.1: Font categorisation and commands

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

Table 7.2: Font declarations

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

Typeset example 7.1: Badly mixed fonts

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

Do not go wild seeing how many different kinds of font you can cram into your work as in example 7.1.

Source for example 7.1

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

On the other hand there are occasions when several fonts may be used for a reasonable effect, as in example 7.2.

Source for example 7.2

```
\begin{center}
\textsc{Des Dames du Temps Jardis}
\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}
```

<code>\emph{<text>}</code>

The `\emph` command is a font changing command that does not fit into the above scheme

Typeset example 7.2: Sometimes mixed fonts work

DES DAMES DU TEMPS JARDIS

*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?]

Typeset example 7.3: 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.

of things. What it does is to typeset its *⟨text⟩* argument using a different font than the surrounding 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 7.3

```
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.
```

Table 7.3: Standard font size declarations

<code>\tiny</code>	tiny	<code>\scriptsize</code>	scriptsize
<code>\footnotesize</code>	footnotesize	<code>\small</code>	small
<code>\normalsize</code>	normalsize	<code>\large</code>	large
<code>\Large</code>	Large	<code>\LARGE</code>	LARGE
<code>\huge</code>	huge	<code>\Huge</code>	Huge

Table 7.4: Standard font sizes

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

7.2 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 specified 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 7.3, 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.

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

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

Table 7.5: The memoir class font size declarations

<code>\miniscule</code>	<small>miniscule</small>	<code>\tiny</code>	<small>tiny</small>
<code>\scriptsize</code>	<small>scriptsize</small>	<code>\footnotesize</code>	<small>footnotesize</small>
<code>\small</code>	<small>small</small>	<code>\normalsize</code>	<small>normalsize</small>
<code>\large</code>	<small>large</small>	<code>\Large</code>	<small>Large</small>
<code>\LARGE</code>	<small>LARGE</small>	<code>\huge</code>	<small>huge</small>
<code>\Huge</code>	<small>Huge</small>	<code>\HUGE</code>	<small>HUGE</small>

Table 7.6: The memoir class font sizes

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

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 names, namely `\miniscule` and `\HUGE`, one at each end of the range.

The memoir class font size names are given in Table 7.5 together with the name set in the specified size for this manual’s normal font. The corresponding actual sizes are given in Table 7.6.

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
60pt normal
size

to the
132pt

size
(the
60pt

\HUGE
size).

This extended range, though, is only accessible if you are using outline fonts. 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.

7.3 SPACES

7.3.1 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.

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 1½ em. To make the first line of a paragraph flushleft set this to zero:

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

7.3.2 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 read³ 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. 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.

²Except that all values except zero are unreasonable.

³I 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.

```
\OnehalfSpacing \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.

```
\SingleSpacing \SetSingleSpace{<factor>}
```

The `\SetSingleSpace` command is meant to be used to adjust *slightly* the normal spacing between lines, perhaps because the font being used looks too cramped or loose. The effect is that the normal `\baselineskip` spacing will be multiplied by `<factor>`, which should be close to 1.0. The declaration `\SingleSpacing` returns everything to normal, or at least the setting from `\SetSingleSpace` if it has been used.

```
\begin{SingleSpace} ... \end{SingleSpace}
\begin{Spacing}{<factor>} ... \end{Spacing}
\begin{OnehalfSpace} ... \end{OnehalfSpace}
\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 original 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.

7.4 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.

7.5 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 orphan⁴ 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:

...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)

⁴Knuth uses the term 'club' instead of the normal typographers' terminology.

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.
\clubpenalty=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.

Eight

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 layout¹ 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...

Figures 2.9 and 2.17, for example, show title pages created using normal LaTeX, without bothering with `\maketitle`.

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.

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

8.1 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 8.1 shows the title page style mandated by Rensselaer 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 8.2. 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].

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

Source for example 8.1

```
\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 reimplementaion 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
  {\prettitle \title \posttitle}
  \maketitlehookb
  {\preauthor \author \postauthor}
  \maketitlehookc
  {\predate \date \postdate}
  \maketitlehookd
  \thispagestyle{title}
}
```

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

**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 8.1: Example of a mandated title page style for a docotoral thesis

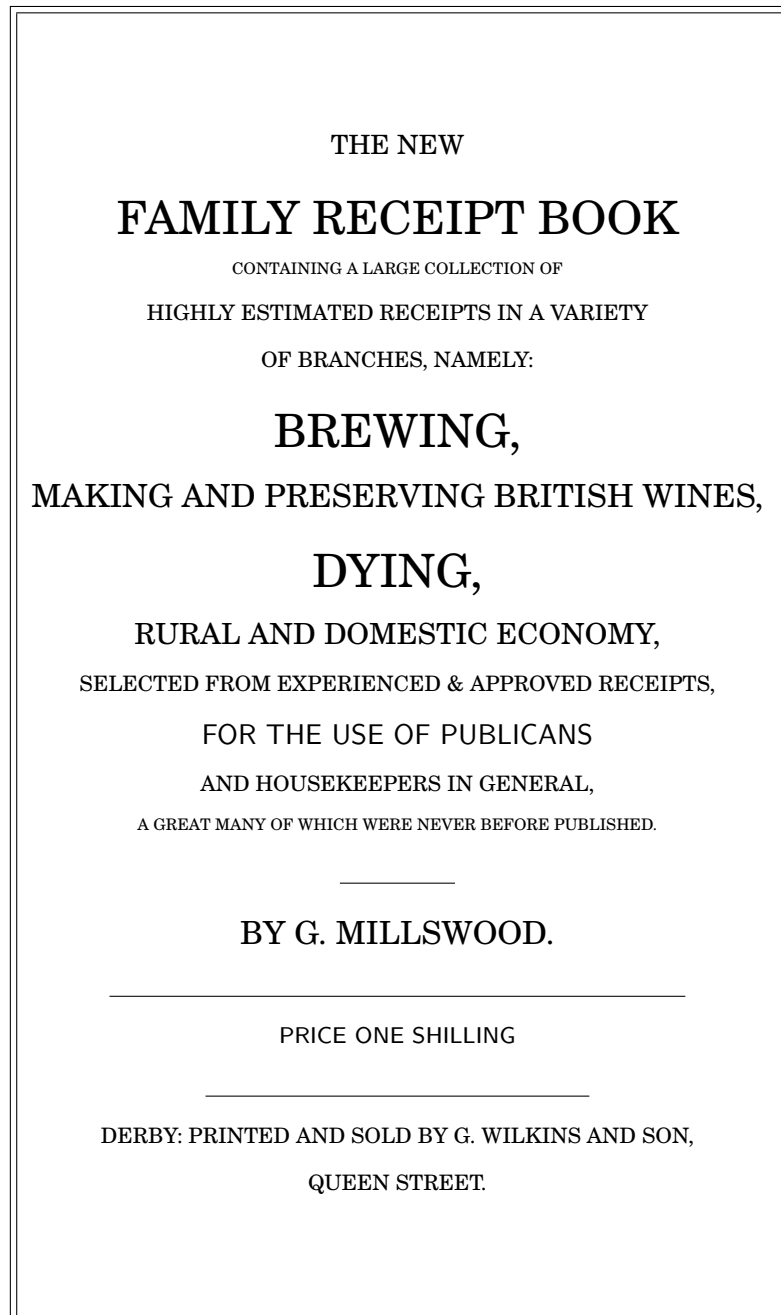


Figure 8.2: Example of a Victorian title page

Typeset example 8.1: 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

```
\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 justified 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}
```

<pre>\maketitlehooka \maketitlehookb \maketitlehookc \maketitlehookd</pre>
--

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.

<pre>\begin{titlingpage} text \end{titlingpage}</pre>

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. The *titlingpage* pagestyle is initially defined to be the same as the *empty* pagestyle.

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, I suggest that you ignore the `titlingpage` environment and just 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

```

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}

```

8. TITLES

```
\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.

8.2 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 $\langle pre \rangle$ and $\langle post \rangle$ before and after the thanks markers in the titling block. By default $\langle pre \rangle$ and $\langle post \rangle$ are empty. For example, to put parentheses round the titling markers do:

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

```
\thanksmark{\langle n \rangle}
```

It is sometimes desirable 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{\langle n \rangle}` is similar to `\footnotemark[\langle n \rangle]` in that it prints a thanks mark identical to that of the $\langle n \rangle$ '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{\langle defn \rangle}
```

By default the thanks mark at the foot is typeset as a superscript. In the class this is specified 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{\thanksfootpre \thefootnote \thanksfootpost}}}
```

where `\thanksfootpre` and `\thanksfootpost` are specified via the `\thanksfootextra` macro. 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.

<pre>\makethanksmark \makethanksmarkhook</pre>
--

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 `baselineskip` for `\footnotesize` in a 10pt document is 9.5pt); adjust these numbers to suit.

<pre>\thanksrule \usethanksrule \cancelthanksrule</pre>

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 16.1 on page 277). You will have to change one or more of these if the vertical spacings of footnotes and thanks notes are meant to be different.

Nine

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.

9.1 STYLING

Much of this part of the class is a reimplementaion 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
```

¹The `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 12.2.

```
\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 `\abstitlekip` is typeset before the heading. For example, if `\absparindent` is non-zero, then:

```
\setlength{\abstitlekip}{-\\absparindent}
```

will typeset the heading flush left.

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

9.2 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.

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.

10.1 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.

```
\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.

10.2 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. Different 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*{<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
\bottomsectionsip
```

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 `\bottomsectionsip` controls the amount of stretch on the short page. Setting it to zero allows the last line to be flush with the bottom of the typeblock. The default setting of 10mm appears to remove any stretch.

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

10.2.1 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.

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

Table 10.1: Division levels

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

```
\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}  
\namesubappendices \anonsubappendices
```

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

10.3 NUMBERING

Each type of sectional division has an associated *level* as shown in Table 10.1. 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 $\langle secname \rangle$ and above will be numbered. The argument $\langle secname \rangle$ 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 $\langle secname \rangle$ 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`.

The number setting commands come from the `tocvsec2` package [Wil99b].

10.4 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\partnamenumber\printpartnum
  \midpartskip               % space after Name and Number
  \printparttitle{#1}        % print the title
  \afterpartskip             % a ‘space’ after the title}
```

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 10.1 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.

```
\beforebookskip \afterbookskip
\beforepartskip \afterpartskip
```

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}{}  
\renewcommand{\printpartnum}
```

```
\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.

10.4.1 Leadpage

```
\newleadpage[<page-style>]{<cmdname>}{<title>}
\newleadpage*[<page-style>]{<cmdname>}{<title>}
\renewleadpage[<page-style>]{<cmdname>}{<title>}
\renewleadpage*[<page-style>]{<cmdname>}{<title>}
```

`\newleadpage` and `associates` are variants of the `\newcommand` and companions.² The `\newleadpage` command defines a macro `\cmdname` that when called will typeset an Appendixpage-like page (see §10.2.1) 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. 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}
```

If you wished them to be entered like a `\part` header then simply:

```
\renewcommand*{\leadpagetoclevel}{part}
```

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

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.

10.5 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.

²The suggestions for these came from Danie Els and Lars Madsen.

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
  \printchaptertertitle{#1}  % title
  \afterchaptertertitle      % \afterchapskip space after title
```

The general layout is shown in Figure 10.1.

`\clearforchapter`

The actual macro that sets the opening page for a chapter is `\clearforchapter`. The `\openright`, `\openleft` and `\openany` define `\clearforchapter` to be respectively (see §22.13) `\cleartorecto`, `\cleartoverso` and `\clearpage`. You can obviously change `\clearforchapter` to do more than just start a new page.

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
\afterchaptertertitle \afterchapskip`

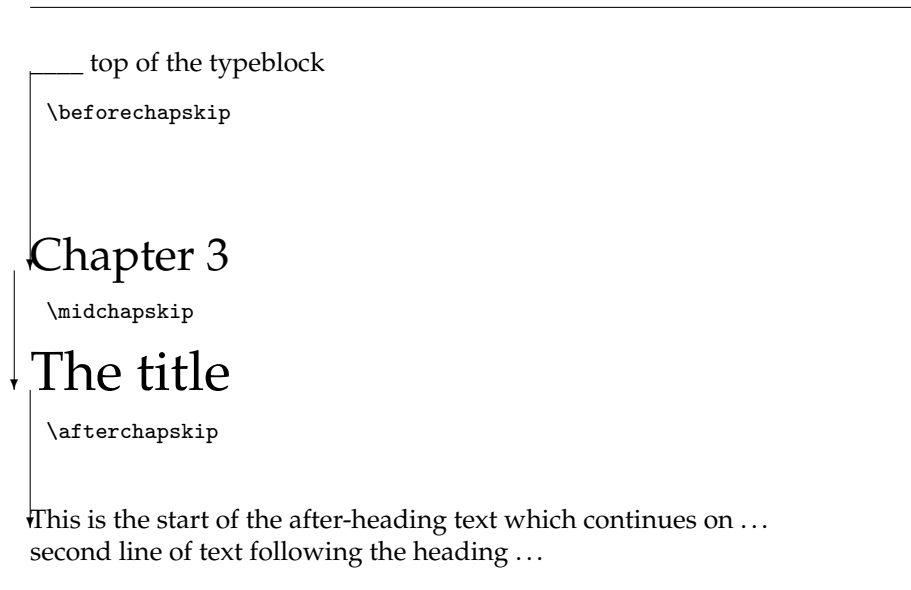


Figure 10.1: Class layout parameters for chapter titles

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`.

```
\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>} \chaptitelfont
```

The title is typeset by `\printchaptertitle` using the font specified by `\chaptitelfont`. 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
```

`\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.

10.5.1 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 10.2.

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 10.3.

hangnum Like the *section* style except that the chapter number is put in the margin, as shown in Figure 10.4.

Chapter 1

Demonstration of the default chapter style

The above is a demonstration of the *default* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure 10.2: The default chapterstyle

2 Demonstration of the section chapter style

The above is a demonstration of the *section* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure 10.3: The section chapterstyle

3 Demonstration of the hangnum chapter style

The above is a demonstration of the *hangnum* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure 10.4: The hangnum chapterstyle

companion This produces chapter headings like those of the *LaTeX Companion* series of books. An example is in Figure 10.5.

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 10.6.

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:

```
\newcommand{\chapterheadstart}{\vspace*{\beforechapskip}}
\newcommand{\printchaptername}{\chapnamefont \@chapapp}
\newcommand{\chapternamenum}{\space}
\newcommand{\printchapternum}{\chapnumfont \thechapter}
\newcommand{\afterchapternum}{\par\nobreak\vskip \midchapskip}
\newcommand{\printchapternonum}{\space}
\newcommand{\printchaptertitle}[1]{\chaptitlefont #1}
\newcommand{\afterchaptertitle}{\par\nobreak\vskip \afterchapskip}
```

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 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 10.5: 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 10.6: The article chapterstyle

Chapter styles are defined via the `\makechapterstyle` command, where *style* is the style being defined and *text* is the LaTeX code defining the style.

To start things off, here is the code for the *default* chapter style, which mimics the chapter heads in the standard book and report classes, as it appears in `memoir.cls`.

```
\makechapterstyle{default}{%
  \def\chapterheadstart{\vspace*{\beforechapskip}}
  \def\printchaptername{\chapnamefont \@chapap}
  \def\chapternamenum{\space}
  \def\printchapternum{\chapnumfont \thechapter}
  \def\afterchapternum{\par\nobreak\vskip \midchapskip}
  \def\printchapternonum{}
  \def\printchaptertitle##1{\chaptitlefont ##1}
  \de\afterchaptertitle{\par\nobreak\vskip \afterchapskip}
}
\newcommand{\chapnamefont}{\normalfont\huge\bfseries}
\newcommand{\chapnumfont}{\normalfont\huge\bfseries}
\newcommand{\chaptitlefont}{\normalfont\Huge\bfseries}
\newlength{\beforechapskip}\setlength{\beforechapskip}{50pt}
\newlength{\midchapskip}\setlength{\midchapskip}{20pt}
\newlength{\afterchapskip}\setlength{\afterchapskip}{40pt}
\chapterstyle{default}
```

(The mysterious `\@chapap` is the internal macro that LaTeX uses to store normally the chapter name.¹ 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.)

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

¹Remember, if you use a macro that has an @ in its name it must be in a place where @ is treated as a letter.

of vertical space taken by any ‘Chapter N’ line. For example, as `\printchapternonum` is vacuous 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}{\chapttitlefont}
  % allow for 99 chapters!
  \settowidth{\chapindent}{\chapnumfont 999}
  \renewcommand*{\printchaptername}{}
  \renewcommand*{\chapternamenum}{}
  \renewcommand*{\chapnumfont}{\chapttitlefont}
  \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 \chapttitlefont ##1\par\nobreak
    \end{adjustwidth}}
}
```

As shown in Figure 10.5 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` environment² to make LaTeX think that the typeblock is locally wider than it actually is.

10.5.2 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

²See §12.4.

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 Bianchi³ 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 10.7.
- bringhurst* The *bringhurst* chapterstyle described in the manual and illustrated in Figure 10.8.
- brotherton* A very simple style designed by William Adams¹ 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 10.9. 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 10.10.
- crosshead* The number and title are centered and set with a large bold font. It is illustrated in Figure 10.11.

³CTT, *New chapter style: chapter vs chapter**, 2003/12/09

¹CTT, *An example of a novel?*, 2006/12/09

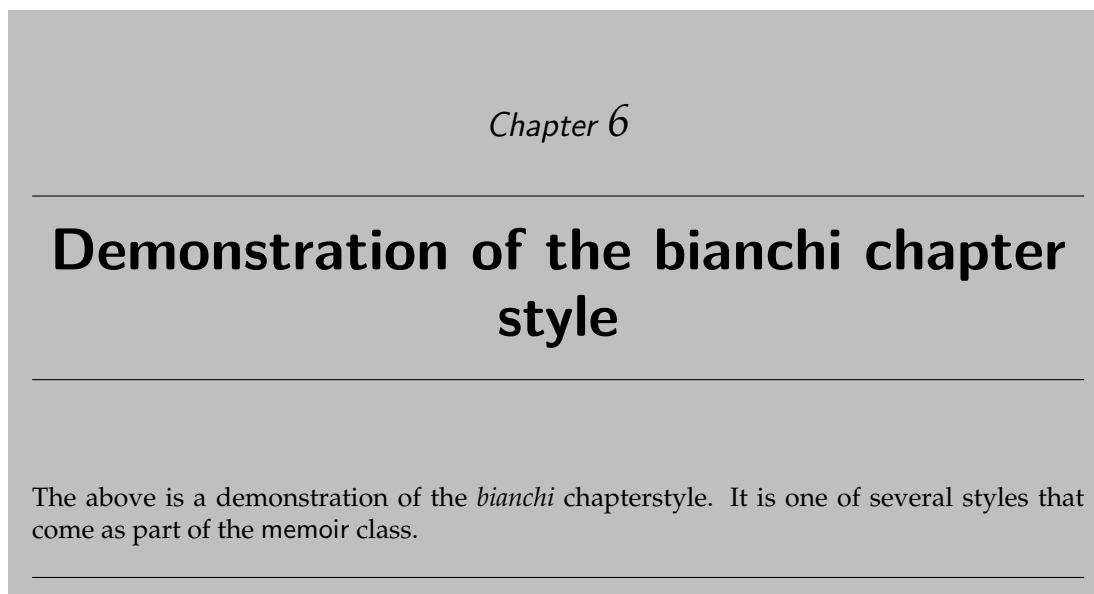


Figure 10.7: The bianchi chapterstyle

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 10.8: The bringhurst chapterstyle

Chapter Eight

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 10.9: 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 10.10: 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 10.11: The crosshead chapterstyle

culver A chapter style I created for Christopher Culver¹ 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 10.12.

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 10.13.

¹CTT, “Biblical” formatting, how?, 2004/03/29

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 10.12: 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 10.13: The dash chapterstyle

- 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 10.14.
- 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 10.15.
- 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 10.16. I will probably use this in my next book.
- ger* This style was created by Gerardo Garcia¹ and is a two line, raggedleft, large bold style with rules above and below. It is demonstrated in Figure 10.17.
- 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 10.18.

¹CTT, *Fancy Headings, Chapter Headings*, 2002/04/12

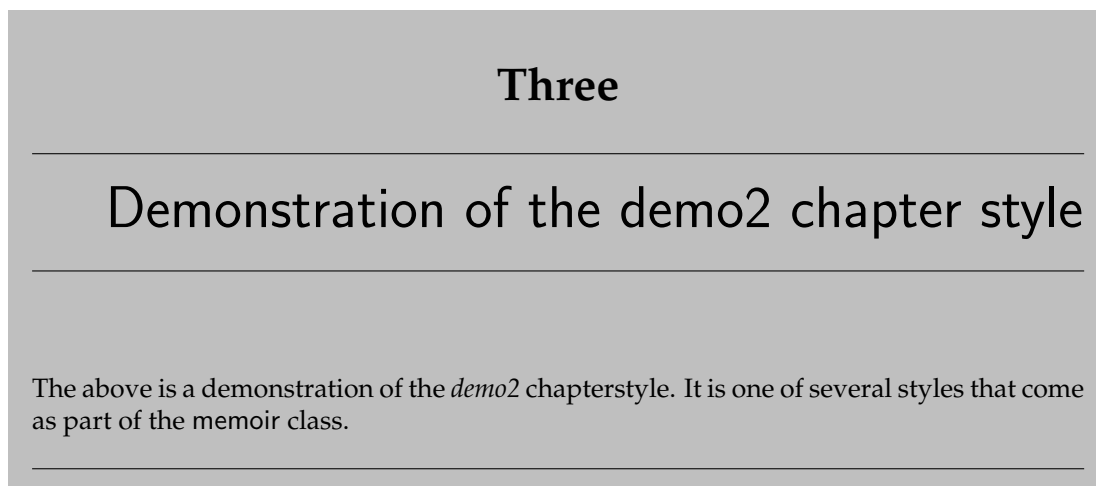


Figure 10.14: The demo2 chapterstyle

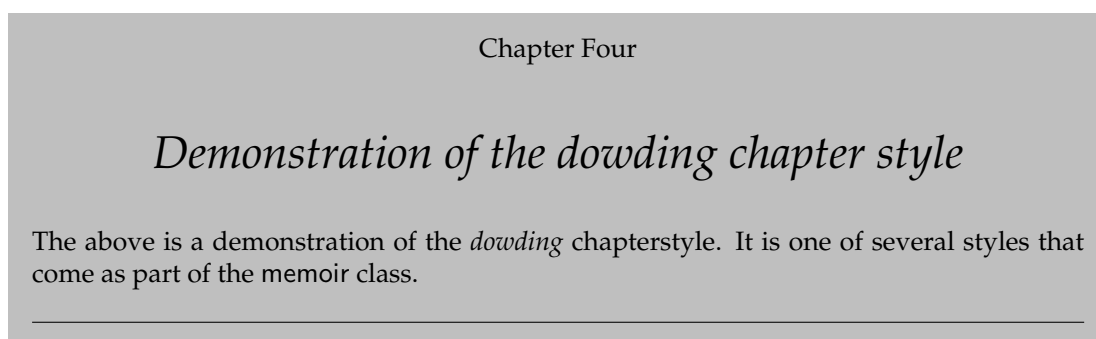


Figure 10.15: The dowing chapterstyle

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 10.16: 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 10.17: 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 10.18: 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 10.19: The lyhne chapterstyle

- lyhne* A style created by Anders Lyhne¹ and shown in Figure 10.19 where the raggedleft sanserif title is between two rules, with the name and number above. I modified the original to cater for unnumbered chapters. It requires the `graphicx` package.
- madsen* This was created by Lars Madsen¹ and is shown in Figure 10.20. It is a large sanserif raggedleft style with the number in the margin and a rule between the number and title lines. It requires the `graphicx` package.
- 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 10.21.
- pedersen* This was created by Troels Pedersen¹ 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.

¹CTT, *Glossary*, 2006/02/09

¹CTT, *New chapter style: chapter vs chapter**, 2003/12/09

¹CTT, *Chapter style*, 2006/01/31

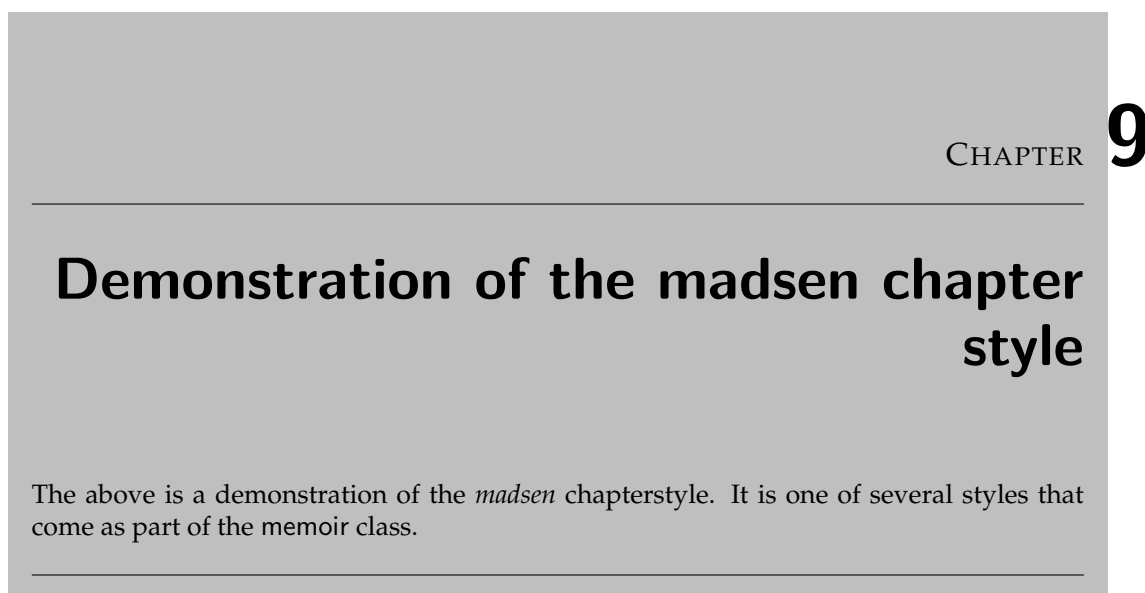


Figure 10.20: The madsen chapterstyle

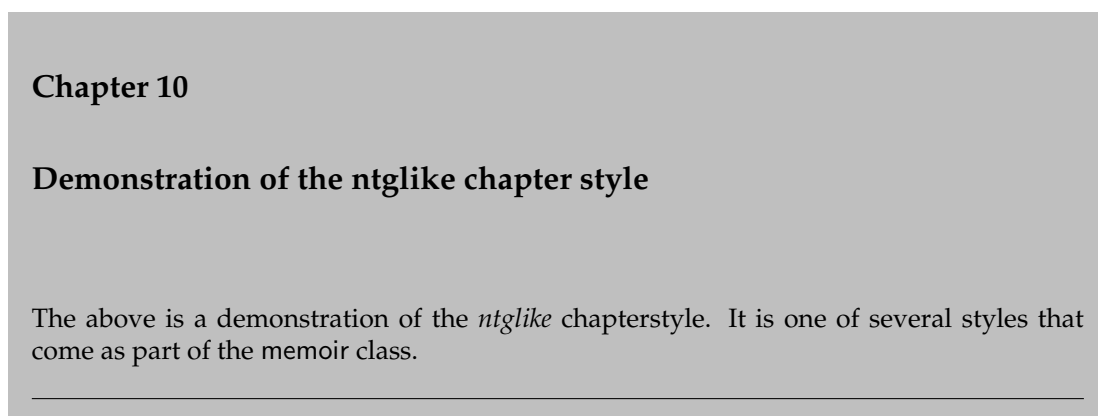


Figure 10.21: The ntglke chapterstyle

- 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 10.22.
- 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 10.23.
- thatcher* A style created by Scott Thatcher¹ 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 10.24. 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 10.25 and is raggedleft, large, bold and with a black square in the margin by the number line. It requires the `graphicx` package.
- verville* A chapterstyle I created for Guy Verville¹. It is a single line, large centered style with rules above and below, as in Figure 10.26. 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 10.27.

The code for some of these styles is given 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.

¹CTT, *memoir: chapter headings capitalize math symbols*, 2006/01/18

¹CTT, *Headers and special formatting of sections*, 2005/01/18

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 10.22: 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 10.23: 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 10.24: The thatcher chapterstyle

10.5.3 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 10.10, 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*{\chaptitelfont}{\Large\itshape}
  \renewcommand*{\printchaptertitle}[1]{%
    \hrule\vskip\onelineskip \centering\chaptitelfont ##1}}
```

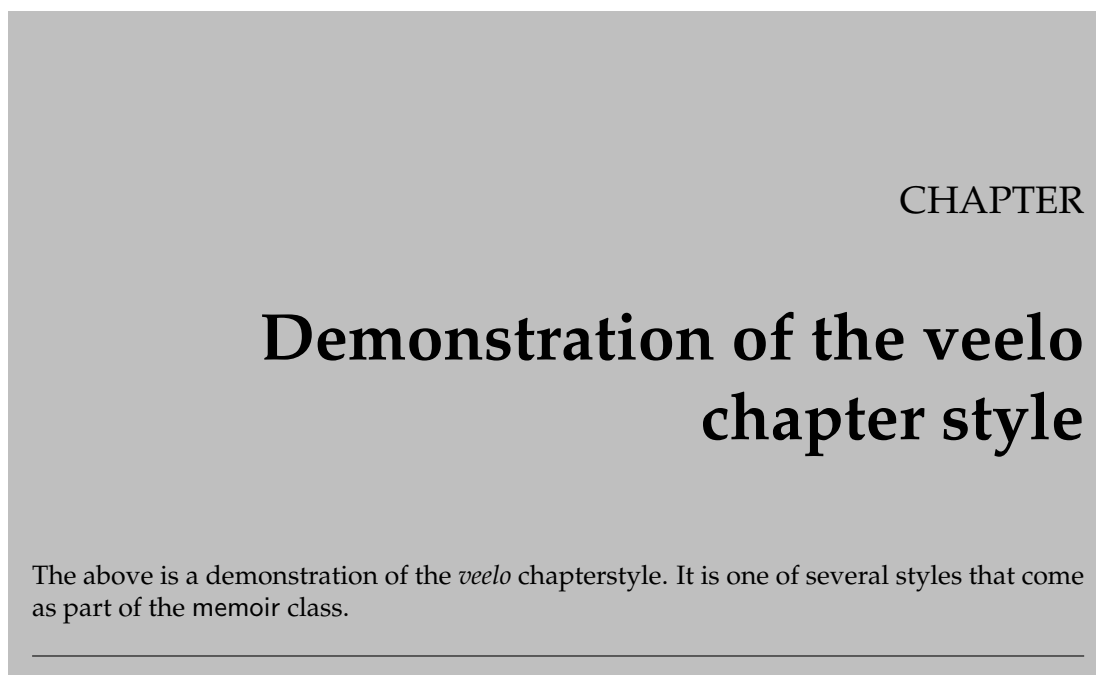


Figure 10.25: The veelo chapterstyle

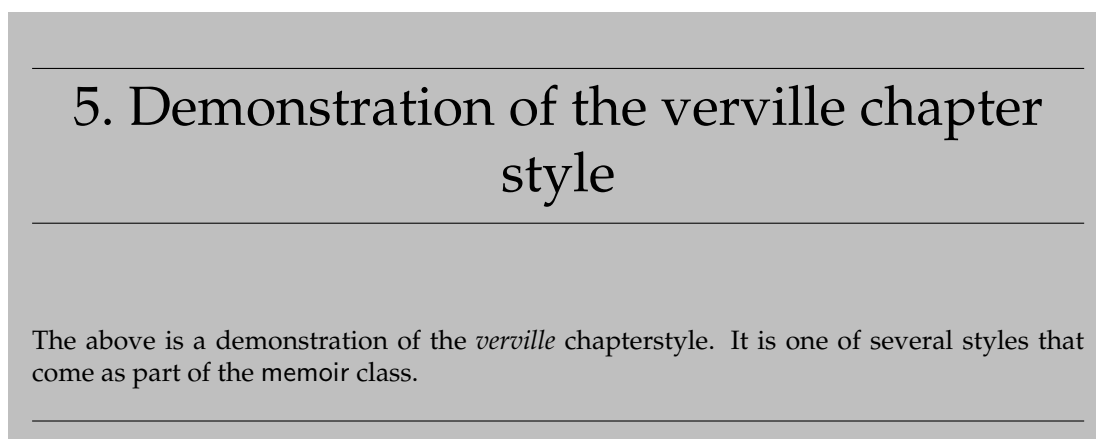


Figure 10.26: The verville chapterstyle

6 *Demonstration of the wilsondob chapter style*

The above is a demonstration of the *wilsondob* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure 10.27: The wilsondob chapterstyle

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).

10.5.4 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*{\chapttitlefont}{\normalfont\Huge\sffamily}
  \renewcommand*{\printchapttitle}[1]{%
    \hrule\vskip\onelineskip \raggedleft \chapttitlefont ##1}
  \renewcommand*{\afterchapttitle}{%
    {\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*{\chapttitlefont}{\normalfont\Huge\sffamily}
```

```

\renewcommand*{\printchaptertitle}[1]{%
  \hrule\vskip\onelineskip \raggedleft \chaptitelfont ##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.

The *demo* chapterstyle is still available in the class for backward compatibility reasons, but I strongly advise against anyone using it.

By chance I inadvertently typed 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}

```

10.5.5 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*{\chaptitelfont}{\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]{\chaptitelfont ##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.

10.5.6 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 book¹ 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}}
  \renewcommand*{\chaptitelfont}{\huge\rmfamily\raggedright}
  \renewcommand*{\chapnumfont}{\chaptitelfont}
  \renewcommand*{\printchaptername}{}
  \renewcommand*{\chapternamenum}{}
  \renewcommand*{\afterchapternum}{}
  \renewcommand*{\printchapternum}{%
    \begin{minipage}[t][\baselineskip][b]{\beforechapskip}
      {\vspace{0pt}\chapnumfont%%\figureversion{lining}
        \thechapter}
    \end{minipage}}
  \renewcommand*{\printchaptertertitle}[1]{%
    \hfill\begin{minipage}[t]{\midchapskip}
      {\vspace{0pt}\chaptitelfont ##1\par}\end{minipage}}
  \renewcommand*{\afterchaptertertitle}{%
    \par\vspace{\baselineskip}%
    \hrulefill \par\nobreak\noindent \vskip\afterchapskip}}

```

The resulting style is shown in Figure 10.22.

10.5.7 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

¹Which I haven't seen

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*{\chapttitlefont}{\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}{\}{\}{\thechapter}}
```

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.

10.5.8 Chapter precis

Some old style novels, and even some modern text books,² 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>}`

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.
```

²For example, Robert Sedgewick, *Algorithms*, Addison-Wesley, 1983.

```
\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*{-2\baselineskip}%
  \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.

```
\precistotext{<text>} \precistocfont
```

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

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

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

...end of last line of preceding text.
 $||beforekip|| + \backslash parskip \text{ (of text font)} + \backslash baselineskip \text{ (of heading font)}$

$\xrightarrow{\text{indent}}$ 3.5 Heading Title
 $\xrightarrow{\text{afterskip} + \backslash parskip \text{ (of heading font)} + \backslash baselineskip \text{ (of text font)}}$

This is the start of the after-heading text, which continues on ...
second line of text following the heading ...

Figure 10.28: Displayed sectional headings

...end of last line of preceding text.
 $||beforekip|| + \backslash parskip \text{ (of text font)} + \backslash baselineskip \text{ (of heading font)}$

$\xrightarrow{\text{indent}}$ 3.5 Heading Title $\xrightarrow{\text{afterskip} (< 0)}$ Start of text ...
second line of text following the heading ...

Figure 10.29: Run-in sectional headings

10.6 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 10.28 shows the parameters controlling a displayed sectional heading and Figure 10.29 shows the parameters for a run-in heading. The default values of the parameters for the different heads are in Table 10.1 for the display heads and Table 10.2 for the runin heads.

In the following I will use S to stand for one of sec, subsec, subsubsec, para or

Table 10.1: 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 10.2: Default runin 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 10.3: Values for S in section styling macro names.

S	sec	subsec	subsubsec	para	subpara
	section	subsection	subsubsection	paragraph	subparagraph

subpara, which are in turn shorthand for section through to subparagraph, as summarised in Table 10.3.

```
\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 default *<skip>* depends on the particular level of heading, but for a `\section` (i.e., when $S = \text{sec}$) 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 ** argument depends on the level of the heading. For a `\subsection` (i.e., $S = \text{subsec}$) 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 `` 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

```
\setsubsubseheadtstyle{\normalfont\centering\MakeUppercase}
```

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}}
\setseheadstyle{\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 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:

```
\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

Typeset example 10.1: 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**

```
\setsecnumformat{\csname the#1\endcsname:\quad}
```

Note that you have to use #1 where you want the argument (sectional number) of \@secntformat to go.

```
\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>}
```

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.

Here are some example lower level heads and the code used to produce them.

Source for example 10.1

```
\setsubsubseheadstyle{\bfseries\raggedright}  
  \subsubsection*{Bold raggedright}  
\setsubsubseheadstyle{\scshape\raggedright}  
  \subsubsection*{Small caps raggedright}
```

```

\setsubsubseheadstyle{\itshape\raggedright}
  \subsubsection*{Italic raggedright}
\setsubsubseheadstyle{\Large\centering}
  \subsubsection*{Large centered}
\setsubsubseheadstyle{\large\centering\MakeUppercase}
  \subsubsection*{large centered uppercase}
\setsubsubseheadstyle{\bfseries\centering}
  \subsubsection*{Bold centered}
\setsubsubseheadstyle{\scshape\centering}
  \subsubsection*{Small caps centered}
\setsubsubsecindent{2\parindent}
\setsubsubseheadstyle{\scshape\raggedright}
  \subsubsection*{Small caps indented}
\setsubsubsecindent{0pt}
\setsubsubseheadstyle{\itshape\raggedleft}
  \subsubsection*{Italic flushright}
\newcommand*{\shortcenter}[1]{%
  \sethangfrom{\noindent #1}%
  \normalfont\boldmath\bfseries
  \centering
  \parbox{3in}{\centering #1}\par}
\setsubsubseheadstyle{\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\mbox{ } #1}}
\newcommand{\marginhead}[1]{%
  {\llap{\marginbox{#1}\kern0.5em}}}
\setparaindent{0em}
\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`.

10.7 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 spaces³ 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
\makeatletter
\def\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.

10.8 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?}}
```

10.9 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 10.4 lists the default fonts used for the sectional headings. These fonts are all bold but in different sizes depending on the division level.

<pre>\makeheadstyles{<name>}{<code>} \headstyles{<name>}</pre>
--

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.

³It is usually a matter for experiment to find the right values for the kerning.

Table 10.4: Default fonts for sectional headings

<code>\booknamefont</code>	<code>\huge\bfseries</code>	huge
<code>\booknumfont</code>	<code>\huge\bfseries</code>	huge
<code>\booktitlefont</code>	<code>\Huge\bfseries</code>	Huge
<code>\partnamefont</code>	<code>\huge\bfseries</code>	huge
<code>\partnumfont</code>	<code>\huge\bfseries</code>	huge
<code>\parttitlefont</code>	<code>\Huge\bfseries</code>	Huge
<code>\chapnamefont</code>	<code>\huge\bfseries</code>	huge
<code>\chapnumfont</code>	<code>\huge\bfseries</code>	huge
<code>\chapttitlefont</code>	<code>\Huge\bfseries</code>	Huge
<code>\secheadstyle</code>	<code>\Large\bfseries</code>	Large
<code>\subsecheadstyle</code>	<code>\large\bfseries</code>	Large
<code>\subsubsecheadstyle</code>	<code>\normalsize\bfseries</code>	normal
<code>\paraheadstyle</code>	<code>\normalsize\bfseries</code>	normal
<code>\subparaheadstyle</code>	<code>\normalsize\bfseries</code>	normal

```

\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

```

Table 10.5: Fonts used by different headstyles

Headstyles	chapter	section	subsec	subsubsec	para	subpara
<i>bringhurst</i>	CAPS	S. CAPS	<i>Italic</i>	S. CAPS	<i>Italic</i>	<i>Italic</i>
<i>crosshead</i>	Bold	CAPS	Bold	S. CAPS	<i>Italic</i>	S. CAPS
<i>default</i>	Bold	Bold	Bold	Bold	Bold	Bold
<i>dowding</i>	<i>Italic</i>	CAPS	S. CAPS	<i>Italic</i>	<i>Italic</i>	<i>Italic</i>
<i>komalike</i>	Sans	Sans	Sans	Sans	Sans	Sans
<i>memman</i>	Sans	S. CAPS	Bold	<i>Italic</i>	<i>Italic</i>	<i>Italic</i>
<i>ntglike</i>	Bold	Bold	Bold	<i>Slanted</i>	<i>Slanted</i>	<i>Slanted</i>
<i>tandh</i>	Bold	CAPS	<i>Italic</i>	Bold	<i>Italic</i>	<i>Italic</i>
<i>wilsondob</i>	<i>Italic</i>	CAPS	<i>Italic</i>	S. CAPS	<i>Italic</i>	<i>Italic</i>

```

\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 *pederson* 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 10.5 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 *bringhurst* chapterstyle (Figure 10.8).
- crosshead* This set uses the *crosshead* chapterstyle and the lower level division titles are set as crossheads.
- default* The default set corresponding the the LaTeX book class.
- dowding* A set based on Dowding's *Finer Points* [Dow96]. It uses the *dowding* chapterstyle (Figure 10.15).
- 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 10.18).
- 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 10.21) 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 10.23)
- wilsondob* A set based on those used in Adrian Wilson's *Design of Books* [Wil93]. It uses the *wilsondob* chapterstyle (Figure 10.27).

Eleven

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.

11.1 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.

```
\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.

11.2 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.

<pre>\pagestyle{<style>} \thispagestyle{<style>}</pre>
--

`\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 11.1.

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 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.

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.

Table 11.1: The use of `\thispagestyle`

Called from	Style
<code>\book</code>	<i>book</i>
<code>\chapter</code>	<i>chapter</i>
<code>\cleardoublepage</code>	<i>cleared</i>
<code>\cleartorecto</code>	<i>cleared</i>
<code>\cleartoverso</code>	<i>cleared</i>
<code>\epigraphhead</code>	<i>epigraph</i>
<code>\listoffigures</code>	<i>chapter</i>
<code>\listoftables</code>	<i>chapter</i>
<code>\maketitle</code>	<i>title</i>
<code>\part</code>	<i>part</i>
<code>\tableofcontents</code>	<i>chapter</i>
<code>thebibliography</code>	<i>chapter</i>
<code>theindex</code>	<i>chapter</i>
<code>titlingpage</code>	<i>titlingpage</i>

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 uppcased. The default is `\uppercaseheads` which specifies that the titles are to be automatically uppcased.

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>}`

`\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>`.

Table 11.2: Mark macros for page headers

Main macro	default mark definition
<code>\book(*)</code>	<code>\newcommand*{\bookpagemark}[1]{}</code>
<code>\part(*)</code>	<code>\newcommand*{\partmark}[1]{}</code>
<code>\chapter(*)</code>	<code>\newcommand*{\chaptermark}[1]{}</code>
<code>\section(*)</code>	<code>\newcommand*{\sectionmark}[1]{}</code>
<code>\subsection(*)</code>	<code>\newcommand*{\subsectionmark}[1]{}</code>
<code>\subsubsection(*)</code>	<code>\newcommand*{\subsubsectionmark}[1]{}</code>
<code>\paragraph(*)</code>	<code>\newcommand*{\paragraphmark}[1]{}</code>
<code>\subparagraph(*)</code>	<code>\newcommand*{\subparagraphmark}[1]{}</code>
<code>\tableofcontents(*)</code>	<code>\newcommand*{\tocmark}[1]{}</code>
<code>\listoffigures(*)</code>	<code>\newcommand*{\lofmark}[1]{}</code>
<code>\listoftables(*)</code>	<code>\newcommand*{\lotmark}[1]{}</code>
<code>\thebibliography</code>	<code>\newcommand*{\bibmark}{}{}</code>
<code>\theindex</code>	<code>\newcommand*{\indexmark}{}{}</code>
<code>\theglossary</code>	<code>\newcommand*{\glossarymark}{}{}</code>
<code>\PoemTitle</code>	<code>\newcommand*{\poemtitlemark}[1]{}</code>
<code>\PoemTitle*</code>	<code>\newcommand*{\poemtitlestarmark}[1]{}</code>

`\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 11.2. 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.

11.3 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 11.1.

The class itself uses the commands from this section. For example, the *plain* pagestyle is defined as

```
\makepagestyle{plain}
```

Figure 11.1: Header and footer slots

```
\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 $\langle style \rangle$ which is initially equivalent to the *empty* pagestyle. On the other hand, `\aliaspagestyle` defines the $\langle alias \rangle$ pagestyle to be the same as the $\langle original \rangle$ 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.

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>}{<length>}
\headwidth
```

The macro `\makerunningwidth` sets the width of the `<style>` pagestyle headers and footers to be `<length>`. The `\makepagestyle` initialises the width 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>}
```

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.

```
\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>}
{<theadpos>}{<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>`. This is normally used for specifying non-default code (i.e., code not specifiable via any of the previous macros) for the particular pagestyle. The code normally defines the marks, if any, that will be used in the headers and footers.

11.3.1 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 memoir 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. \ %
        }
      }
    }
  }
  \def\sectionmark##1{%
    \markright{\MakeUppercase{%
      \ifnum\c@secnumdepth > \z@
        \thesection. \ %
      }
    }
  }
}
```

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 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{\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}}}%
  \def\tocmark{\markboth{\MakeUppercase{\contentsname}}{}}
  \def\lofmark{\markboth{\MakeUppercase{\listfigurename}}{}}
  \def\lotmark{\markboth{\MakeUppercase{\listtablename}}{}}
  \def\bibmark{\markboth{\MakeUppercase{\bibname}}{}}
  \def\indexmark{\markboth{\MakeUppercase{\indexname}}{}}
  \def\glossarymark{\markboth{\MakeUppercase{\glossaryname}}{}}
```

As you can see, defining the marks for a pagestyle is not necessarily the simplest thing in the world. However, courtesy of Lars Madsen, help is at hand.

```
\createplainmark{<type>}{<marks>}{<text>}
\memUHead{<text>}
\uppercaseheads \nouppercaseheads
\createmark{<sec>}{<marks>}{<show>}{<prefix>}{<postfix>}
```

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{\memUHead{\contentsname}}{}}
\def\lotmark{\markright{\memUHead{\listtablename}}}
\def\lofmark{\markboth{\memUHead{\bibname}}{\memUHead{\bibname}}}
```

Following the declaration `\uppercaseheads` the `\memUHead` command is equivalent to `\MakeUppercase` but after the `\nouppercaseheads` it is equivalent to `\relax` (which does nothing). The `\createplainmark` macro wraps `\memUHead` around the `<text>` argument within the generated `\mark(both/right)` macro. By using the `\(no)uppercaseheads` declarations you can control the uppercasing, or otherwise, of

the mark texts. The default is `\uppercaseheads`.

The macro `\createmark{<sec>}{<marks>}{<show>}{<prefix>}{<postfix>}` defines the `\<sec>mark` macro where `<sec>` is a sectional division such as `part`, `chapter`, `section`, etc., and `<show>` (`shownumber` or `nonnumber`) controls whether the division number will be displayed within `\mainmatter`. The `<marks>` argument is `left`, `both` or `right`, and `<prefix>` and `<postfix>` are affixed before and after the division number. For example:

```
\createmark{section}{left}{nonnumber}{}{}
\createmark{section}{both}{nonnumber}{}{}
\createmark{section}{right}{nonnumber}{}{}
```

is equivalent to, respectively

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

Using these macros memoir's current definition of `\makepsmarks{headings}` is much simpler (it also leads to a slightly different result as the `toc` etc., marks set both the `\leftmark` and `\rightmark` instead of just the `\leftmark`):

```
\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}}
```

This next 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 ToC¹ and other similar elements, and this is where the `\makepsmarks` macro comes into play.

```
\makepsmarks{companion}{%  
  \nouppercaseheads  
  \createmark{chapter}{both}{nonnumber}{}{}  
  \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}
```

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 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}  
\makeevenfootruled{\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
```

¹The ToC and friends are described in detail in Chapter 13.

```

\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}
}

```

```

\addtopmarks{<pagestyle>}{<prepend>}{<append>}

```

`\addtopmarks{<pagestyle>}{<prepend>}{<append>}` is the last of this group of helper macros. It inserts `<prepend>` and `<append>` before and after the current definition of `\makepsmarks` for `<pagestyle>`. 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:

```

\addtopmarks{companion}{}{%
  \createmark{subsection}{right}{shownumber}{. \space}}

```

11.3.2 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, but the standard `fixltx2e` package corrects this.

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.

11.3.3 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}%
  {\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@tempwa
  \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.
```

Twelve

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.

12.1 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 12.1. 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.

12.1.1 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.

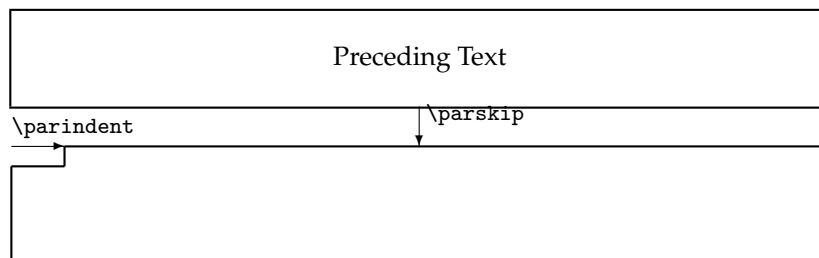


Figure 12.1: Paragraphing parameters

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.

12.1.2 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 elsewhere 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
```

12.2 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 similarly 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.

```
\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[Opt]   % 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.

12.3 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.

12.4 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 `chngpage` package [Wil01b], 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 `lequotation` 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 \lazypagecheck
```

Odd/even page checking may be either strict (`\strictpagecheck`) or lazy (`\lazypagecheck`). Lazy 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 desirable 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 14.1 on page 223, 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 §6.4 (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 inches¹ 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>}

```

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

¹On A4 paper the result would be different.

spine margin and subtracted from the foreedge 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).

12.5 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` environment²).

This class provides the normal `description` list but the `itemize` and `enumerate` lists are extended versions of the normal ones.

```
\begin{description} \item[⟨label⟩] ... \end{description}  
\descriptionlabel⟨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 `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 ...  
...
```

²The `quote` and `quotation` environments are also defined in terms of the general `list` environment. You may be surprised where it crops up.

```
\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)]
```

...

```
\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 slightly less space around the lists when they are preceded by a blank line than does the unstarred `\firmlists`.

```
\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 12.2 shows the parameters controlling the layout of the `list` environment.

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.

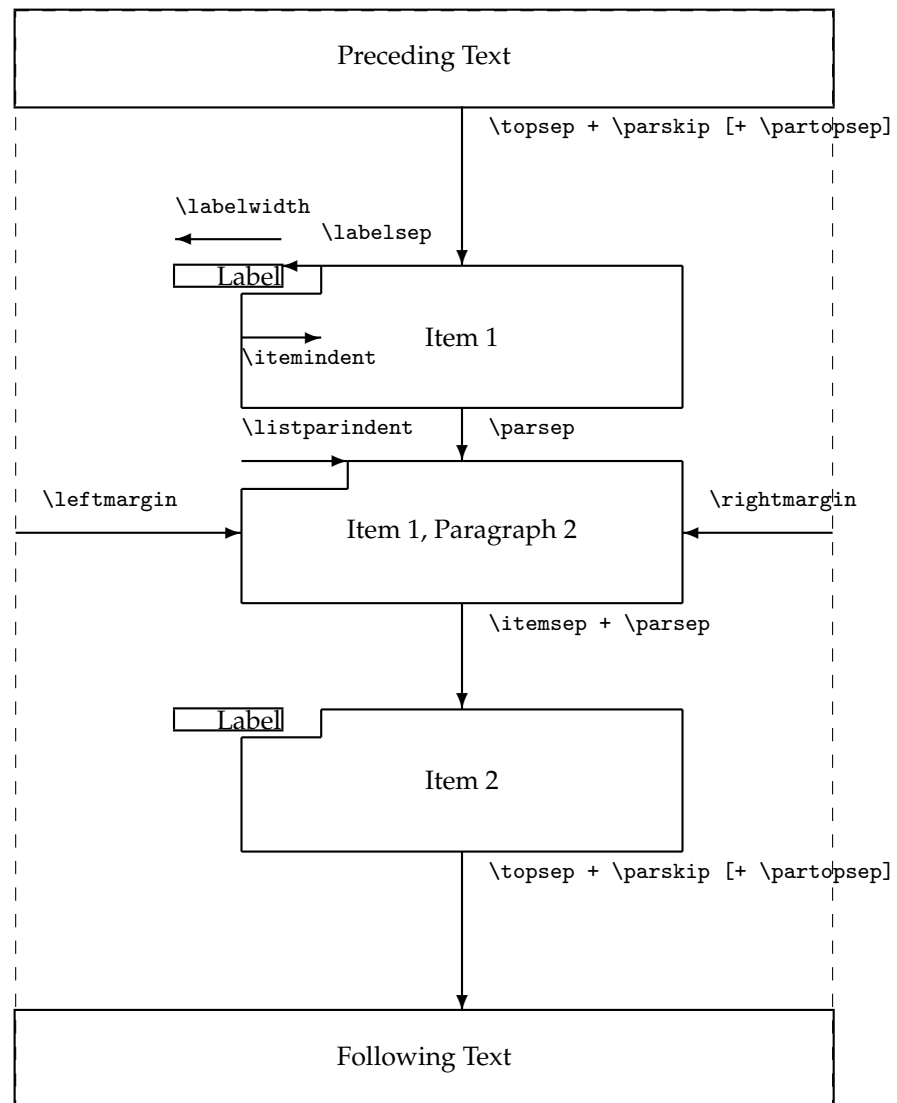


Figure 12.2: The layout parameters for general lists

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{}}\noprofilebreak}
```

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

Typeset example 12.1: 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.

`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 12.2.

`\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 12.1

```
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 (12.1) shows how it is possible to change the spacing around the `center` environment, but it applies equally to the other environments.

Thirteen

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, respectively, 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 appearance 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.

13.1 GENERAL TOC METHODS

To provide 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 Table 10.1 on page 124. 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>}
```

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 13.1: Example extracts from toc, lof and lot files

LaTeX generates a file if the document contains a `\tableofcontents` command. The sectioning commands¹ put entries into the 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 *<title>* 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 file:

```
\contentsline{section}{\numberline{3.4} Head text}{27}
```

Extracts from toc, lof and lot files are shown in Figure 13.1.

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.

¹For figures and tables it is the `\caption` command that populates the `lof` and `lot` files.

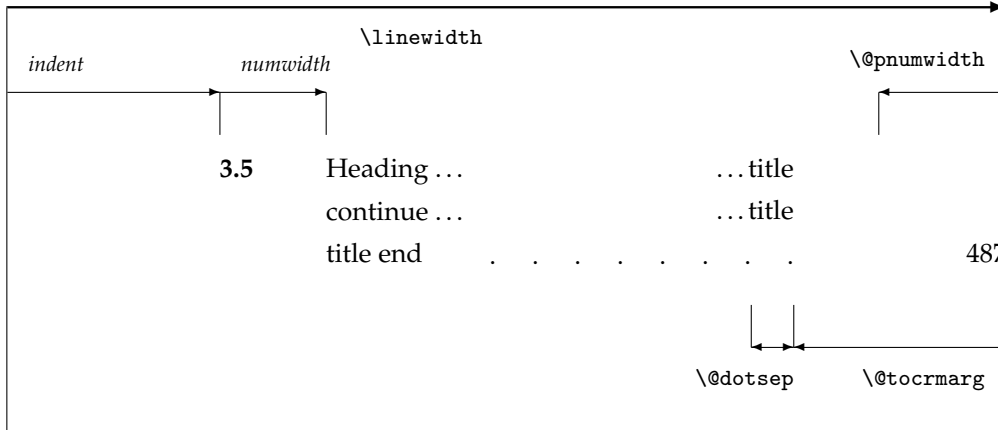


Figure 13.2: Layout of a ToC (LoF, LoT) entry

```
\@pnumwidth{<length>}
\@tocrmarg{<length>}
\@dotsep{<number>}
```

The general layout of a typeset entry is illustrated in Figure 13.2. 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 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

²There are 18mu to 1em.

Table 13.1: 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 $(\text{linewidth} - \text{indent} - \text{numwidth} - \text{@tocrmarg})$.

Table 13.1 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 13.1, and `<indent>` and `<numwidth>` are the *indent* and *numwidth* as illustrated in Figure 13.2. 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 \protects 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 §B.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}
...
```

13. CONTENTS LISTS

13.2 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. 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.

You can use `\tableofcontents`, `\listoffigures`, etc., more than once in a memoir class document.

```
\maxtocdepth{<secname>}
\settocdepth{<secname>}
```

The class `\maxtocdepth` command sets the maximum allowable value for the `tocdepth` counter. If used, the command must appear before the `\tableofcontents` command. By default, the class sets `\maxtocdepth{section}`.

The memoir class command `\settocdepth` is somewhat analogous to the `\setsecnumdepth` command described in §10.3. It sets the value of the `tocdepth` counter and puts it into the ToC to (temporarily) modify what will appear. The command can only be used after the preamble but may be used before calling the `\tableofcontents`. 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`.

13.2.1 Changing the titles

Commands are provided for controlling the appearance of the ToC, LoF and LoT titles.

```
\contentsname \listfigurename \listtablename
```

Table 13.2: Values for X in macros for styling the titles of ‘List of...’

toc	lof	lot	...
-----	-----	-----	-----

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 13.2, 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:

```
\toheadstart
\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.

13.2.2 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 [Wil03a] or in [MG⁺04, p. 51], and are repeated in Figure 13.2.

Most of the commands in this section start as `\cft...`, where *cft* is intended as a mnemonic for *Table of Contents*, *List of Figures*, *List of Tables*.

`\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

Table 13.3: Value of K in macros for styling entries in a ‘List of...’

K	Kind of entry	K	Kind of entry
book	<code>\book title</code>	subparagraph	<code>\subparagraph title</code>
part	<code>\part title</code>	figure	figure caption
chapter	<code>\chapter title</code>	subfigure	subfigure caption
section	<code>\section title</code>	table	table caption
subsection	<code>\subsection title</code>	subtable	subtable caption
subsubsection	<code>\subsubsection title</code>

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 13.3; that is, any of the following:

- book for `\book` titles.
- 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 13.2). It can be changed using `\setlength`.

`\cftKnumwidth`

This length controls the space allowed for typesetting title numbers (*numwidth* in Figure 13.2). 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\cftaftersnum\hfil}
```

is typeset within a box of width `\cftKnumwidth`.

`\cftKfont`

This controls the appearance of the title (and its preceding number, if any). It may be changed using `\renewcommand`.

`\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`

³Suggested by Danie Els.

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.

```
\booknumberline{<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. 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]{}%
```

```
\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 13.2). 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.

```
\cftKformatpnum{<pnum>}
\cftKpagefont
```

The macro `\cftKformatpnum` typesets an entry's page number, using the `\cftKpagefont`.⁴ The default definition is essentially:

```
\newcommand*{\cftKformatpnum}[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]{%
  \hbox to \@pnumwidth{{\cftpartpagefont #1}}}
```

```
\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

⁴This addition to the class was suggested by Dan Luecking, CTT Re: *setting numbers in toc in their natural width box*, 2007/08/15.

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}
```

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 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}{\cftparskip}
```

`\cftparskip`

By default the `\parskip` 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 `\cftparskip` command is provided just so that the above effect can be achieved.

`\cftpagenumbersoff{<kind>}`  
`\cftpagenumberon{<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 `\cftpagenumberon` 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}{\cftpagenumberon{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 yourself 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`.

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`.

```
\precistotext{text} \precistocfont
```

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

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

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

### 13.2.3 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 original \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.

#### 13.2.4 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 [Mit98]. 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{multicols}}}
```

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.

#### 13.2.5 Example: Multiple contents

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\oldprecistoc\precistoc
 \renewcommand{\precistoc}[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@section}[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.

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\precistoc\oldprecistoc
 \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`).

### 13.3 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}{\thechapter.\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:

|                                          |
|------------------------------------------|
| <p><b>Answer 1. Hard</b><br/>The ...</p> |
|------------------------------------------|

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 §13.2.1. Then in addition to `\listofcom` the following new commands will be made available.

The four commands, `\Xmark`, `\Xheadstart`, `\printXtitle`, and `\afterXtitle`, are analogous to the commands of the same names described in §13.2.1 (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.

|                            |
|----------------------------|
| <p><code>Xdepth</code></p> |
|----------------------------|

The counter `Xdepth` is analogous 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}
```

|                                                                                          |
|------------------------------------------------------------------------------------------|
| <pre>\insertchapterspace \addtodef{&lt;macro&gt;}{&lt;prepend&gt;}{&lt;append&gt;}</pre> |
|------------------------------------------------------------------------------------------|

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 §22.10.

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.

|                                                                                      |
|--------------------------------------------------------------------------------------|
| <pre>\newlistentry[&lt;within&gt;]{&lt;cntr&gt;}{&lt;ext&gt;}{&lt;level-1&gt;}</pre> |
|--------------------------------------------------------------------------------------|

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,  $\langle ext \rangle$ , is the file extension for the entry listing. The last argument,  $\langle level-1 \rangle$ , is a number specifying the numbering level minus one, of the entry in a listing.

Calling `\newlistentry` creates several new commands. Assuming that it 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 analogous to the commands of the same names described in §13.2.2. Their default values are also as described earlier.

The default values of `\cftKindent` and `\cftKnumwidth` are set according to the value of the  $\langle level-1 \rangle$  argument (i.e.,  $N$  in this example). For  $N=0$  the settings correspond to those for figures and tables, as listed in Table 13.1 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  $\langle within \rangle$  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 be-

low 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 §B.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.

### 13.3.1 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<sup>5</sup> 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}\themempref}
\newcommand{\nextpageref}[1]{%
 \setcounter{mempref}{\pageref{#1}}\addtocounter{mempref}{1}\themempref}
```

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`.

## 13.4 CHAPTER PRECIS

Some old style novels, and even some modern text books,<sup>6</sup> 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>}`

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.}
...
```

`\chapterprecishere{<text>}`  
`\chapterprecistoc{<text>}`

<sup>5</sup>These only work for arabic page numbers.

<sup>6</sup>For example, Robert Sedgewick, *Algorithms*, Addison-Wesley, 1983.

The `\chapterprecis` command calls these two commands to print the  $\langle text \rangle$  in the document (the `\chapterprecishere` command) and to put it into the ToC (the `\chapterprecistoc` command). These can be used individually if required.

`\prechapterprecis \postchapterprecis`

The `\chapterprecishere` macro is intended for use immediately after a `\chapter`. The  $\langle text \rangle$  argument is typeset in italics in a quote environment. The macro's definition is:

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

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

```
\newcommand{\prechapterprecis}{%
 \vspace*{-2\baselineskip}%
 \begin{quote}\normalfont\itshape}
\newcommand{\postchapterprecis}{\end{quote}}
```

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

`\precistotext{\langle text \rangle} \precistocfont`

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

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

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

# Fourteen

---

## 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.

### 14.1 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 preceded 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
\listofdiagrams
...
\begin{diagram}
\caption{A diagram} \label{diag1}
...
\end{diagram}
As diagram~\ref{diag1} shows ...
```

#### 14.2 SETTING OFF A FLOAT

Sometimes it is desirable to set off a float, more probably an illustration than a tabular, from its surroundings. The `framed` environment, described later in Chapter 19, might come in handy for this.

The following code produces the example Figures 14.1 and 14.2.

```
\begin{figure}
\centering
```

FRAMED FIGURE

Figure 14.1: Example framed figure

FRAMED FIGURE AND CAPTION

Figure 14.2: Example framed figure and caption

```

\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 14.3 and 14.4.

```

\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}
\hrule
\end{figure}

```

---

RULED FIGURE

---

Figure 14.3: Example ruled figure

---

RULED FIGURE AND CAPTION

---

Figure 14.4: Example ruled figure and caption

---

## ILLUSTRATION 1

## ILLUSTRATION 2

Figure 14.5: Example float with two illustrations

## 14.3 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 14.5 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}
```

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 14.6 and 14.7 which are examples of two separately captioned illustrations in one float.

```
\begin{figure}
\centering
```

GRAPHIC 1

Figure 14.6: Graphic 1 in a float

GRAPHIC 2

Figure 14.7: Graphic 2 in same float

```

\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 minipages. 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 minipages.

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 14.6 and 14.7) and you want to align them horizontally. Here are some examples.

This code produces Figures 14.8 and 14.9. 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
}\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}\hhrule

```

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 14.8: Left center aligned

Figure 14.9: Right figure. This has  
more text than the adjacent caption  
(14.8) so the heights are unequal

```
\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 14.10 and 14.11.

```
\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}\hhrule
\null\hfill\parbox[t]{0.4\linewidth}{%
 \caption{Left top aligned}\label{fig:left2}%
}\hfill
\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 14.12 and 14.13.

|                                        |                                                                                                             |
|----------------------------------------|-------------------------------------------------------------------------------------------------------------|
| 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 14.10: Left top aligned         | Figure 14.11: Right figure. This has more text than the adjacent caption (14.10) so the heights are unequal |

|                                           |                                                                                                             |
|-------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| 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 14.12: Left bottom aligned         | Figure 14.13: Right figure. This has more text than the adjacent caption (14.12) so the heights are unequal |

```

\begin{figure}
\hrule \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
\hrule
\end{figure}

```

`\newsubfloat{<float>}`

The `\newsubfloat` command creates subcaptions (`\subcaption`, `\subtop` and `\subbottom`) for use within the float environment  $\langle fenv \rangle$  previously defined via `\newfloat[\langle...\rangle]{\langle fenv \rangle}{\langle...\rangle}`. Subcaptions are discussed below.

#### 14.4 WHERE L<sup>A</sup>T<sub>E</sub>X PUTS FLOATS

The general format for a float environment is:

`\begin{float}[\langle loc \rangle] ... \end{float}` or for double column floats:

`\begin{float*}[\langle loc \rangle] ... \end{float*}`

where the optional argument  $\langle loc \rangle$ , 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  $\langle loc \rangle$  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  $\langle loc \rangle$  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*<sup>1</sup>. 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 succeeding floats will be held up, and L<sup>A</sup>T<sub>E</sub>X 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 to the float placement parameters. A `\clearpage` or `\cleardoublepage` or `\end{document}` flushes out all unprocessed floats, irrespective of the  $\langle loc \rangle$  and float parameters, putting them on float-only pages.

`\setfloatlocations{\langle float \rangle}{\langle locs \rangle}`

You can set the location for all floats of type  $\langle float \rangle$  to  $\langle locs \rangle$  with the `\setfloatlocations` declaration. The class initialises these using:

```
\setfloatlocations{figure}{tbp}
```

```
\setfloatlocations{table}{tbp}
```

`\suppressfloats[\langle pos \rangle]`

You can use the command `\suppressfloats` to suppress floats at a given  $\langle pos \rangle$  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.

---

<sup>1</sup>This little quirk is fixed by the `fixltx2e` package, at least for tables and figures. The package is part of a normal L<sup>A</sup>T<sub>E</sub>X distribution.

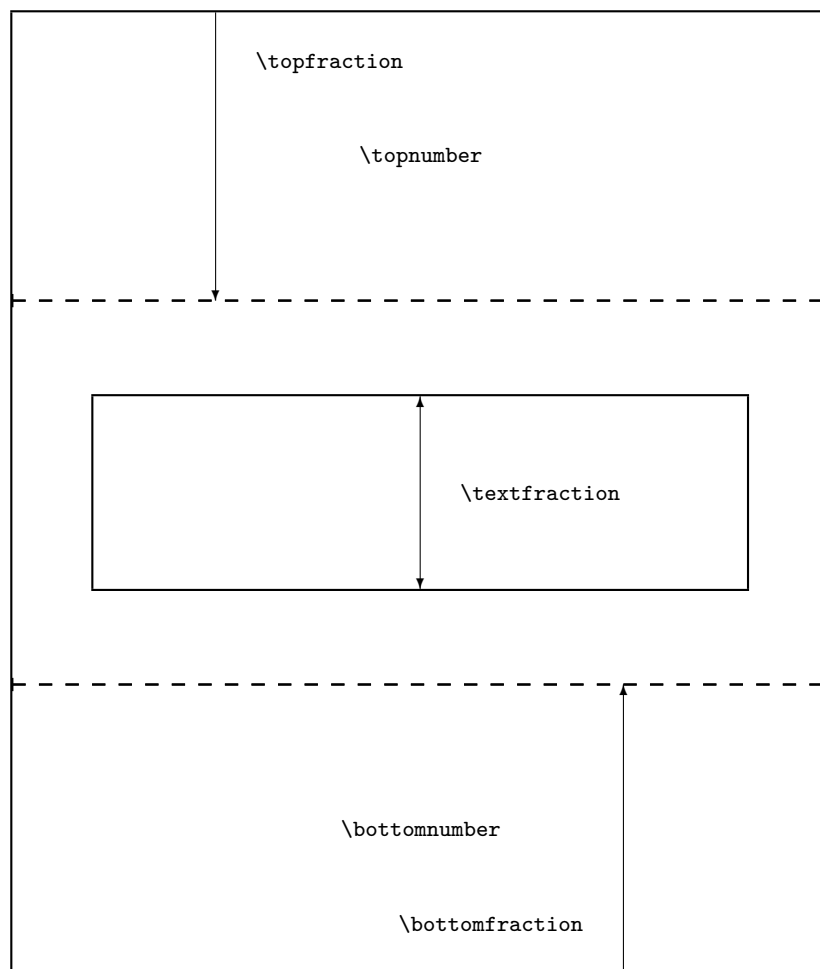


Figure 14.14: Float and text page parameters

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 14.14 and 14.15 illustrate the many float parameters and Table 14.1 lists the float parameters and the typical standard default values. The lengths controlling the spaces surrounding floats are listed in Table 14.2; 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

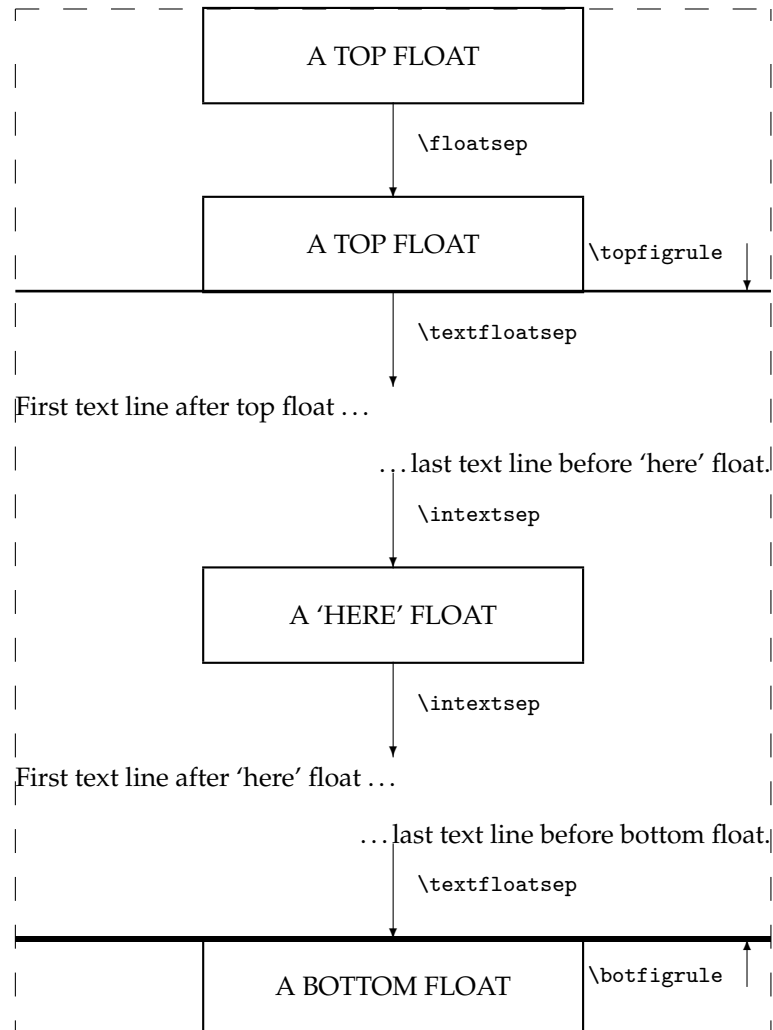


Figure 14.15: Float parameters

Table 14.1: Float placement parameters

| Parameter                                         | Controls                                                      | Default |
|---------------------------------------------------|---------------------------------------------------------------|---------|
| Counters — change with <code>\setcounter</code>   |                                                               |         |
| <code>topnumber</code>                            | max number of floats at top of a page                         | 2       |
| <code>bottomnumber</code>                         | max number of floats at bottom of a page                      | 1       |
| <code>totalnumber</code>                          | max number of floats on a text page                           | 3       |
| <code>dbltopnumber</code>                         | like <code>topnumber</code> for double column floats          | 2       |
| Commands — change with <code>\renewcommand</code> |                                                               |         |
| <code>\topfraction</code>                         | max fraction of page reserved for top floats                  | 0.7     |
| <code>\bottomfraction</code>                      | max fraction of page reserved for bottom floats               | 0.3     |
| <code>\textfraction</code>                        | min fraction of page that must have text                      | 0.2     |
| <code>\dbltopfraction</code>                      | like <code>\topfraction</code> for double column floats       | 0.7     |
| <code>\floatpagefraction</code>                   | min fraction of a float page that must have float(s)          | 0.5     |
| <code>\dblfloatpagefraction</code>                | like <code>\floatpagefraction</code> for double column floats | 0.5     |

Table 14.2: Float spacing parameters

| Parameter                                                | Controls                                                                     | Default       |
|----------------------------------------------------------|------------------------------------------------------------------------------|---------------|
| Text page lengths — change with <code>\setlength</code>  |                                                                              |               |
| <code>\floatsep</code>                                   | vertical space between floats                                                | 12pt          |
| <code>\textfloatsep</code>                               | vertical space between a top (bottom) float and succeeding (preceeding) text | 20pt          |
| <code>\intextsep</code>                                  | vertical space above and below an h float                                    | 12pt          |
| <code>\dblfloatsep</code>                                | like <code>\floatsep</code> for double column floats                         | 12pt          |
| <code>\dbltextfloatsep</code>                            | like <code>\textfloatsep</code> for double column floats                     | 20pt          |
| Float page lengths — change with <code>\setlength</code> |                                                                              |               |
| <code>\@fptop</code>                                     | space at the top of the page                                                 | 0pt plus 1fil |
| <code>\@fpsep</code>                                     | space between floats                                                         | 8pt plus 2fil |
| <code>\@fpbot</code>                                     | space at the bottom of the page                                              | 0pt plus 1fil |
| <code>\@dblfpptop</code>                                 | like <code>\@fptop</code> for double column floats                           | 0pt plus 1fil |
| <code>\@dblfpsep</code>                                  | like <code>\@fpsep</code> for double column floats                           | 8pt plus 2fil |
| <code>\@dblfpbot</code>                                  | like <code>\@fpbot</code> for double column floats                           | 0pt plus 1fil |

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.

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 `\@dblftop` 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.

## 14.5 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].

## 14.6 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 delimiter in the form of a colon between the caption number and the caption title. The command `\captiondelim` can be used to change the delimiter. For example, to have an en-dash instead of the colon, `\captiondelim{-- }` will do the trick. Notice that no space is put between the delimiter and the title unless it is specified in the `<delim>` parameter. The class initially specifies `\captiondelim{: }` to give the normal delimiter.

```
\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 delimiter (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{...}`. 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>}
\captiontitlefinaltext
\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>]
```

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}
```

Table 14.3  
REDESIGNED TABLE CAPTION STYLE

|       |      |
|-------|------|
| three | III  |
| five  | V    |
| eight | VIII |

```
\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 14.3, 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 14.3:

```

\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}}

```

Remember that any code that involves the @ sign must be either in a package (sty) file or enclosed between a \makeatletter ... \makeatother pairing (see §B.4).

The code for the Table 14.3 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 appearance.

## 14.7 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 14.4 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}

\begin{table}
\centering
\contcaption{Continued}
\begin{tabular}{lc} \toprule
 just a single line & 2 \\\bottomrule
\end{tabular}
\end{table}

```

Table 14.4: A multi-part table

|                    |   |
|--------------------|---|
| just a single line | 1 |
|--------------------|---|

Table 14.4: Continued

|                    |   |
|--------------------|---|
| just a single line | 2 |
|--------------------|---|

Table 14.4: Concluded

|                    |   |
|--------------------|---|
| just a single line | 3 |
|--------------------|---|

```
\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 14.5. The `\legend` command can be used within a float independently of any `\caption` command.

```
\begin{table}
\centering
\caption{Another table} \label{tab:legend}
```

Table 14.5: Another table

|                   |   |
|-------------------|---|
| A legendary table | 5 |
| with two lines    | 6 |

The legend

Legendary table

|                    |   |
|--------------------|---|
| An anonymous table | 5 |
| with two lines     | 6 |

```

\begin{tabular}{lc} \toprule
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.

```

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{}
 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}

```

## LEGEND

This is a  
marginal  
note with a  
legend.

Table: *Named legendary table*

|       |      |
|-------|------|
| seven | VII  |
| eight | VIII |

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 exemplars 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>}
\providfixedcaption[<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 to the page number in the typeset document, *not* your source input order).

#### 14.8 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...’.

```
\bitwouncaption[⟨label⟩]{⟨short1⟩}{⟨long1⟩}%
{⟨NAME⟩}{⟨short2⟩}{⟨long2⟩}
\bionenumcaption[⟨label⟩]{⟨short1⟩}{⟨long1⟩}%
{⟨NAME⟩}{⟨short2⟩}{⟨long2⟩}
```

Bilingual captions can be typeset by the `\bitwouncaption` 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:

```
\bitwouncaption{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:

```
\bitwouncaption[fig:bi1]{}{Long}{Bild}{ }{Lang}
```

Both language texts are entered into the appropriate ‘List of...’, and both texts are numbered.

Figure 14.16, typeset from the following code, is an example.

```
\begin{figure}
\centering
EXAMPLE FIGURE WITH BITWOUNCAPTION
\bitwouncaption[fig:bi1]%
 {}{Long \cs{bitwouncaption}}%
 {Bild}{ }{Lang \cs{bitwouncaption}}
\end{figure}
```

Both `\bionenumcaption` and `\bitwouncaption` 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 14.17, typeset from the following, is an example of this.

```
\begin{figure}
```

## EXAMPLE FIGURE WITH BITWONUMCAPTION

Figure 14.16: Long \bitwonumcaption

Bild 14.16: Lang \bitwonumcaption

## EXAMPLE FIGURE WITH BIONENUMCAPTION

Figure 14.17: Long English \bionenumcaption

Bild 14.17: Lang Deutsch \bionenumcaption

```

\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.

Figure 14.18 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}

```

## EXAMPLE FIGURE WITH A RULED BICAPTION

Figure 14.18: Longingly

Bild 14.18: Langlauf

```
\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{}`.

## 14.9 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<sup>2</sup> has been specified.

```
\subcaption[⟨list-entry⟩]{⟨subtitle⟩}
```

<sup>2</sup>See §14.1.

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⟩*.

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`, `\contsubtop` 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 `\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 `\labeled` 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 parentheses 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 14.19 and the paragraph immediately following this one were produced by the code shown below.

Figure 14.19 has two subfigures, namely 14.19(a) and (b).

```
Figure \ref{fig:twosubfig} has two subfigures,
namely \ref{sf:1} and \subcaptionref{sf:2}.
\begin{figure}
```

SUBFIGURE ONE

(a) Subfigure 1

SUBFIGURE TWO

(b) Subfigure 2

Figure 14.19: Figure with two subfigures

```
\centering
\subbottom[Subfigure 1]{\fbox{SUBFIGURE ONE}\label{sf:1}}
\hfill
\subbottom[Subfigure 2]{\fbox{SUBFIGURE TWO}\label{sf:2}}
\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.

#### 14.10 SIDE CAPTIONS

Typically captions are put either above or below the the element they are describing. Sometimes it is desirable to put a caption at the side of the element instead.

```
\begin{sidecaption}[\langle fortoc \rangle]{\langle title \rangle}[\langle label \rangle]
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{\langle sep \rangle}{\langle width \rangle}
```

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{\langle margin \rangle}
\ifscapmargleft \scapmarglefttrue \scapmargleftfalse
```

If the float is a single column float in a twocolumn document then the caption is always<sup>3</sup> placed in the adjacent margin, otherwise the `\sidecapmargin` command controls the margin where the sidecaption will be placed. The possible values for `\langle margin \rangle` 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.

---

<sup>3</sup>Well, nearly always. See the `\overridescapmargin` command later.

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.

#### 14.10.1 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}
```

Table 14.6: Permitted arguments for some sidecaption related commands

| <code>\sidecapmargin</code> | <code>\overridescapmargin</code> |
|-----------------------------|----------------------------------|
| left                        | left                             |
| right                       | right                            |
| inner                       |                                  |
| outer                       |                                  |

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 14.6 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.

## 14.11 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 `lfloat.dtx`) defines the caption command via

```

\def\caption{%
 \refstepcounter{@captype} \@dblarg{\@caption\@captype}}

```

`\@capytype`

`\@capytype` 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\@capytype{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 responsibility 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
```

## A THOUSAND WORDS...

FIGURE 14.20: A picture is worth a thousand words

```

\sbbox\@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 §B.4). So, if you modify the `\fnum@figure` command anywhere in your document it has to be done as:

```

\makeatletter
\renewcommand{\fnum@figure}{.....}
\makeatother

```

As an example, Figure 14.20 was created by the following code:

```

\makeatletter
\renewcommand{\fnum@figure}{\textsc{\figurename~\thefigure}}
\makeatother
\begin{figure}

```

ANOTHER THOUSAND WORDS...

**Figure 14.21** — A different kind of figure caption

```
\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 14.21. 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.

#### 14.12 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.



# Fifteen

---

## Rows and columns

---

The class provides extensions to the standard `array` and `tabular` environments. These are based partly on a merging of the `array` [MC98], `dcolumn` [Car01], `delarray` [Car94], `tabularx` [Car99], and `booktabs` [Fea03] packages. Much of the material in this chapter strongly reflects the documentation of these packages.

Additionally, new kinds of tabular environments are also provided.

### 15.1 GENERAL

```
\[\begin{array}[\langle pos \rangle]{\langle preamble \rangle} rows \end{array} \]
\begin{tabular}[\langle pos \rangle]{\langle preamble \rangle} rows \end{tabular}
\begin{tabular*}[\langle width \rangle]{\langle pos \rangle}{\langle preamble \rangle} rows \end{tabular*}
\begin{tabularx}[\langle width \rangle]{\langle pos \rangle}{\langle preamble \rangle} 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 `\langle pos \rangle` 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 `\langle preamble \rangle`. 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 `\langle width \rangle` argument.

### 15.2 THE PREAMBLE

You use the `\langle preamble \rangle` 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 15.1.

Examples of the options include:

- A flush left column with bold font can be specified with `>\bfseries`1.  

```
\begin{center}
\begin{tabular}{>\large c >\large \bfseries l >\large \itshape c}
A & B & C \\
```

Table 15.1: The array and tabular preamble options.

|                                                            |                                                                                                                                                                                                                                                                                  |
|------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>l</code>                                             | Left adjusted column.                                                                                                                                                                                                                                                            |
| <code>c</code>                                             | Centered adjusted column.                                                                                                                                                                                                                                                        |
| <code>r</code>                                             | Right adjusted column.                                                                                                                                                                                                                                                           |
| <code>p{&lt;width&gt;}</code>                              | Equivalent to <code>\parbox[t]{&lt;width&gt;}</code> .                                                                                                                                                                                                                           |
| <code>m{&lt;width&gt;}</code>                              | Defines a column of width <code>&lt;width&gt;</code> . Every entry will be centered in proportion to the rest of the line. It is somewhat like <code>\parbox{&lt;width&gt;}</code> .                                                                                             |
| <code>b{&lt;width&gt;}</code>                              | Coincides with <code>\parbox[b]{&lt;width&gt;}</code> .                                                                                                                                                                                                                          |
| <code>&gt;{&lt;decl&gt;}</code>                            | Can be used before an <code>l</code> , <code>r</code> , <code>c</code> , <code>p</code> , <code>m</code> or a <code>b</code> option. It inserts <code>&lt;decl&gt;</code> directly in front of the entry of the column.                                                          |
| <code>&lt;{&lt;decl&gt;}</code>                            | Can be used after an <code>l</code> , <code>r</code> , <code>c</code> , <code>p{. . .}</code> , <code>m{. . .}</code> or a <code>b{. . .}</code> option. It inserts <code>&lt;decl&gt;</code> right after the entry of the column.                                               |
| <code> </code>                                             | Inserts a vertical line. The distance between two columns will be enlarged by the width of the line.                                                                                                                                                                             |
| <code>@{&lt;decl&gt;}</code>                               | Suppresses inter-column space and inserts <code>&lt;decl&gt;</code> instead.                                                                                                                                                                                                     |
| <code>!{&lt;decl&gt;}</code>                               | Can be used anywhere and corresponds with the <code> </code> option. The difference is that <code>&lt;decl&gt;</code> is inserted instead of a vertical line, so this option doesn't suppress the normally inserted space between columns in contrast to <code>@{. . .}</code> . |
| <code>D{&lt;ssep&gt;}{&lt;osep&gt;}{&lt;places&gt;}</code> | Column entries aligned on a 'decimal point'                                                                                                                                                                                                                                      |

```
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 &
2 2 2 2 2 2 2 &
3 3 3 3 & \\ \bottomrule
\end{tabular}
\end{center}
```

|         |         |         |
|---------|---------|---------|
| 1 1 1 1 | 2 2 2 2 | 3 3 3 3 |
| 1 1 1 1 | 2 2 2 2 |         |
| 1 1 1 1 |         |         |

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     | 1 2 3 4 5 6 7 8 |
| 7 8 9 0 1 2 3 4 | 9 0 1 2 3 4 5 6 |
| 5 6 7 8 9 0     | 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.

### 15.2.1 D column specifiers

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.

$D\langle ssep \rangle\langle osep \rangle\langle places \rangle$

- $\langle ssep \rangle$  is the single character which is used as the separator in the source `.tex` file. Thus it will usually be `'.'` or `'.'`.
- $\langle osep \rangle$  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.
- $\langle places \rangle$  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 `\newcolumnntype` macro described later.

```
\newcolumnntype{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.

```
\newcolumnntype{.}{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 & & \\\
& & & & & & & & \\\
& & & & & & & & \\\
\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 15.1

```
\begin{center}\small
\begin{tabular}[t]{|D{.}{.}{-1}|D{.}{.}{1}|D{.}{.}{5.1}||}
\multicolumn{1}{|c|}{head} & &
\multicolumn{1}{|c|}{head} & &
\end{tabular}
```

Typeset example 15.1: 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      |              |              |              |

```

\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 \\\
.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  $\pm$  sign.

### 15.2.2 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 `\newcolumn` type lets you define a new column option like, say

```
\newcolumn{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 15.1 on page 250), 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.

### 15.2.3 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 overridden.

- 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 `|1|1|1|1|` 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 `|`.

### 15.3 THE ARRAY ENVIRONMENT

Mathematical arrays are usually produced using the `array` environment.

```
\[\begin{array}[\langle pos \rangle]{\langle preamble \rangle} rows \end{array} \]
```

```
\[\begin{array}[\langle pos \rangle]{\langle left \rangle}{\langle preamble \rangle}{\langle right \rangle} rows \end{array} \]
```

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 delimiters are often large and have to be indicated using `\left` and `\right` commands.

```
\[\left[\begin{array}{cc}
x_{\{1\}} & x_{\{2\}} \\
x_{\{3\}} & x_{\{4\}}
\end{array} \right] \]
```

$$\begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \end{bmatrix}$$

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:

```
\[\begin{array}{(cc)} a & b \\ c & d \end{array} \]
```

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Or, a little more complex

```
\[\begin{array}({c})
 \begin{array}|{cc}|
 x_{1} & x_{2} \\
 x_{3} & x_{4} \\
 \end{array} \\
 y \\
 z \\
\end{array} \]
```

$$\left( \begin{array}{cc|c} x_1 & x_2 & \\ x_3 & x_4 & \\ y & & \\ z & & \end{array} \right)$$

And you can do things like this:

```
\[a = {\begin{array}|{*{20}{c}}|
 x-\lambda & 1 & & 0 \\
 0 & & x-\lambda & 1 \\
 0 & & & x-\lambda \\
\end{array}
}^{\sim{2}} \]
```

$$a = \left| \begin{array}{ccc} x - \lambda & 1 & 0 \\ 0 & x - \lambda & 1 \\ 0 & & x - \lambda \end{array} \right|^2$$

As another example, a construct equivalent to plain TeX's `\cases` could be defined by:

```
\[f(x)=\begin{array}\{{lL}.
 0 & \text{if } x=0 \\
 \sin(x)/x & \text{otherwise} \\
\end{array} \]
```

$$f(x) = \begin{cases} 0 & \text{if } x = 0 \\ \sin(x)/x & \text{otherwise} \end{cases}$$

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\sqrt{2}\sqrt{3} \end{array}
\begin{array}[c]({c}) 1\sqrt{2}\sqrt{3} \end{array}
\begin{array}[b]({c}) 1\sqrt{2}\sqrt{3} \end{array}
\quad \mbox{not} \quad
\left(\begin{array}[t]({c}) 1\sqrt{2}\sqrt{3} \end{array}\right)
\left(\begin{array}[c]({c}) 1\sqrt{2}\sqrt{3} \end{array}\right)
\left(\begin{array}[b]({c}) 1\sqrt{2}\sqrt{3} \end{array}\right)
```

Table 15.2: Demonstrating the parts of a table

| stub        | spanner         |                 |      |      |
|-------------|-----------------|-----------------|------|------|
|             | head<br>subhead | head<br>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    |

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \text{ not } \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

## 15.4 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 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 15.2.

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.

Table 15.3: Two views of one table

| (a) Before |         |         | (b) After |             |            |
|------------|---------|---------|-----------|-------------|------------|
| gnats      | gram    | \$13.65 | Item      |             |            |
|            | each    | .01     | Animal    | Description | Price (\$) |
| gnu        | stuffed | 92.50   | Gnat      | per gram    | 13.65      |
| emu        |         | 33.33   |           | each        | 0.01       |
| armadillo  | frozen  | 8.99    | Gnu       | stuffed     | 92.50      |
|            |         |         | Emu       | stuffed     | 33.33      |
|            |         |         | Armadillo | frozen      | 8.99       |

For example, in Table 15.3 which was produced from the code below, Table 15.3(a) is from the LaTeX book and Table 15.3(b) is how Simon Fear [Fea03] 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}
\subtop[Before]{\label{tab:before}%
\begin{tabular}{|l|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 15.4.

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.

Table 15.4: Micawber's law

|             |           |         |
|-------------|-----------|---------|
| Income      | £20-0-0   | £20-0-0 |
| Expenditure | £19-0-6   | £20-0-6 |
| Result      | happiness | misery  |

Table 15.5: A narrow table split half and half

| Relative contents of odd isotopes for heavy elements |    |          |         |    |          |
|------------------------------------------------------|----|----------|---------|----|----------|
| Element                                              | Z  | $\gamma$ | Element | Z  | $\gamma$ |
| Sm                                                   | 62 | 1.480    | W       | 74 | 0.505    |
| Gd                                                   | 64 | 0.691    | Os      | 76 | 0.811    |
| Dy                                                   | 66 | 0.930    | Pt      | 78 | 1.160    |
| ...                                                  |    |          | ...     |    |          |

- 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 Dickens's *David Copperfield* (1849–1850) Mr Wilkins Micawber states:  
'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<sup>1</sup> as in Table 15.4.

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 15.5.

## 15.5 FEAR'S RULES

Simon Fear disapproves of the default LaTeX table rules and wrote the `booktabs` package [Fea03] 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

<sup>1</sup>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.

rule has an optional length argument,  $\langle width \rangle$ , 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.

`\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[$\langle width \rangle$]($\langle trim \rangle$){ $\langle m-n \rangle$ }
\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  $\langle m \rangle$  to  $\langle n \rangle$  inclusive (where  $\langle m \rangle$  and  $\langle n \rangle$  are column numbers, with  $\langle m \rangle$  not greater than  $\langle n \rangle$ ).

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 `\cmidrules`, 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[$\langle width \rangle$] \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  $\langle width \rangle$ . 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{ $\langle width \rangle$ }{ $\langle abovespace \rangle$ }{ $\langle belowspace \rangle$ }
```

You can, but should not, generate peculiar spacing between rules by using `\specialrule`. The three required arguments are the width,  $\langle width \rangle$ , 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 = 0.5em
\cmidrulewidth = 0.3em
\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 15.3(b) on page 258:

```
\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}
```

### 15.5.1 Fills

The rules described previously go between rows. Sometimes it may be desirable to put a rule or something similar within a row.

`\downbracefill \hrulefill \upbracefill`

Examples of `\downbracefill`, `\hrulefill`, and `\upbracefill` are illustrated in Table 15.6, 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 \ \
\end{tabular}
```

Table 15.6: Example table with fills

|   |     |     |      |        |
|---|-----|-----|------|--------|
| 1 | 2   | 3   | 4    | 5      |
| Q |     | fgh | jklm | qwerty |
| v | as  | x   | y    | z      |
| g | nnn |     |      |        |
|   |     |     | pq   | dgh    |
| k | j   | t   | co   | ytrewq |
| 1 | 2   | 3   |      |        |

```

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 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             |
| <i>Q</i> |            | <i>fgh</i> | <i>jklm</i> | <i>qwerty</i> |
| <i>v</i> | <i>as</i>  | <i>x</i>   | <i>y</i>    | <i>z</i>      |
| <i>g</i> | <i>nnn</i> |            |             |               |
|          |            |            | <i>pq</i>   | <i>dgh</i>    |
| <i>k</i> | <i>j</i>   | <i>t</i>   | <i>co</i>   | <i>ytrewq</i> |
| 1        | 2          | 3          |             |               |

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
\end{array}

```

```
A & B & C & D
\end{array}
\end{displaymath}
```

|     |     |     |     |
|-----|-----|-----|-----|
| 1   | 2   | 3   | 4   |
| $a$ |     |     | $d$ |
| $A$ | $B$ | $C$ | $D$ |

## 15.6 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 accommodate 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>` argument.

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.

### 15.6.1 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}
```

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:

Figure 15.1: 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. |

Figure 15.2: 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. <sup>2</sup> | 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. |  |

Figure 15.3: 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. |

- `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.<sup>3</sup>
- `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`

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

---

<sup>3</sup>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.

```
\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

```
{>{\hspace=.5\hsize}X>{\hspace=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 sure that the real definition is restored afterwards.

This version of `\verb` and `\verb*` are subject to the following restrictions:

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.

## 15.7 SPACES AND RULES

### 15.7.1 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 15.1.

`\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.

### 15.7.2 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  
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  
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.

### 15.7.3 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 `l|l` 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.)

## 15.8 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.

### 15.8.1 Continuous tabulars

```
\begin{ctabular}[\langle pos \rangle]{\langle format \rangle} 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  $\langle format \rangle$  argument is the same as for the previous `array` and `tabular` environments, but the optional  $\langle pos \rangle$  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 its translation, without having to be concerned about page breaks.

|                                    |                                      |
|------------------------------------|--------------------------------------|
| Je suis François, 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 15.7: Example automatic row ordered table

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

### 15.8.2 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 15.7 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 accommodate 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 15.4 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 }
...
\autorows[0pt]{c}{5}{c}{one, two, three,
 ... fourteen }
\end{figure}
```

|                                                |        |          |          |      |
|------------------------------------------------|--------|----------|----------|------|
| $\langle width \rangle = -1pt$                 |        |          |          |      |
| one                                            | two    | three    | four     | five |
| six                                            | seven  | eight    | nine     | ten  |
| eleven                                         | twelve | thirteen | fourteen |      |
| $\langle width \rangle = 0pt$ (the default)    |        |          |          |      |
| one                                            | two    | three    | four     | five |
| six                                            | seven  | eight    | nine     | ten  |
| eleven                                         | twelve | thirteen | fourteen |      |
| $\langle width \rangle = 0.9\text{\textwidth}$ |        |          |          |      |
| one                                            | two    | three    | four     | five |
| six                                            | seven  | eight    | nine     | ten  |
| eleven                                         | twelve | thirteen | fourteen |      |

Figure 15.4: Changing the width of a row ordered table

```

...
\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}

```

|                                                                                                                                              |
|----------------------------------------------------------------------------------------------------------------------------------------------|
| $\backslash\text{autocols}[\langle width \rangle]{\langle pos \rangle}{\langle num \rangle}{\langle style \rangle}{\langle entries \rangle}$ |
|----------------------------------------------------------------------------------------------------------------------------------------------|

The `\autocols` macro lists its  $\langle entries \rangle$  in columns, proceeding top to bottom and left to right. The arguments, are the same as for `\autorows`, except that a negative  $\langle width \rangle$  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  $\langle width \rangle$  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 15.5, 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}
\end{figure}

```

|                                             |       |       |        |          |
|---------------------------------------------|-------|-------|--------|----------|
| $\langle width \rangle = 0pt$ (the default) |       |       |        |          |
| one, two                                    | five  | eight | eleven | thirteen |
| three                                       | six   | nine  | twelve | fourteen |
| four                                        | seven | ten   |        |          |

|                                                        |       |       |        |          |
|--------------------------------------------------------|-------|-------|--------|----------|
| $\langle width \rangle = 0.9\text{\texttt{textwidth}}$ |       |       |        |          |
| one, two                                               | five  | eight | eleven | thirteen |
| three                                                  | six   | nine  | twelve | fourteen |
| four                                                   | seven | ten   |        |          |

Figure 15.5: Changing the width of a column ordered table

# Sixteen

---

## 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 `\marginpars` on the page. The class additionally supplies commands for fixed marginal notes and sidebars.

### 16.1 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 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 desirable 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{... adults or babies. \label{fn:rabbits}}
...
... The footnote \footref{fn:rabbits} on \pref{fn:rabbits} ...
```

In this manual, the last line above prints:

```
... The footnote3 on page 17 ...
```

```
\multfootsep
```

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

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

```
\feetabovelfloat
\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 `\feetabovelfloat` declaration, which is the default.

### 16.1.1 A variety of footnotes

```
\verbfootnote[⟨num⟩]{⟨text⟩}
```

The macro `\verbfootnote` is like the normal `\footnote` except that its *⟨text⟩* argument 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.
```

---

<sup>1</sup>One footnote

<sup>2</sup>Immediately followed by another

Below, footnote 3 is a `\footnote` while footnote 4 is a `\verbfootnote`.

The `\verbfootnote` command should appear<sup>3</sup> to give identical results as the normal `\footnote`, but it can include some verbatim text<sup>4</sup> in the `<text>` 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.

```
\footnoteA{<text>}
\footnoteB{<text>}
\footnoteC{<text>}
```

<sup>3</sup>There may be some problems if color is used.

<sup>4</sup>The `\footnote` macro, like all other macros except for `\verbfootnote`, can not contain verbatim text in its argument.

In addition to the regular `\footnote` the class provides three further series of footnotes, namely the ‘A’, ‘B’, and ‘C’ series which are distinguished by appending the series’ uppercase letter at the end of the command, like `\footnoteB` for the ‘B’ series. Perhaps the normal footnotes are required, marked flagged with arabic numerals, and another kind of footnote flagged with roman numerals. Each series has its own `\footnotemarkB`, `\footnotetextB` and so on matching the regular commands.

```
\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{\@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.

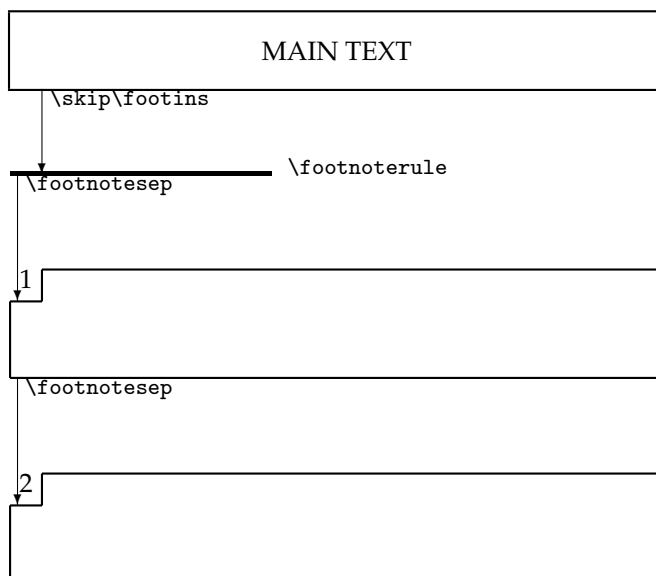


Figure 16.1: Footnote layout parameters

### 16.1.2 Styling

The parameters controlling the vertical spacing of footnotes are illustrated in Figure 16.1.

There is a discussion in §8.2 starting on page 114 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[1]{\hbox{#1}}
```

You can change this if, for example, you wanted the marks to be in parentheses at the baseline.

```
\renewcommand*\@makefnmark[1]{\footnotesize (#1)}
```

or, somewhat better to take account of the size of the surrounding text

```
\renewcommand*\@makefnmark[1]{\slashfracstyle{(#1)}}
```

```
\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{#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:

```
\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{#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 16.1 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:

Table 16.1: Some footnote text styles

| <code>\footmarkwidth</code> | <code>\footmarksep</code> | 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      |

Some

```
\wrapfootnote{disciplines}{For example, Celtic studies.}
require double marks in the text.
```

Some<sup>5</sup>disciplines<sup>5</sup> 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}, \dots \@fnsymbol{9}` produces the series:

$*$ ,  $\dagger$ ,  $\ddagger$ ,  $\S$ ,  $\P$ ,  $\|$ ,  $**$ ,  $\dagger\dagger$ , and  $\ddagger\ddagger$ .

Robert Bringhurst quotes the following as the traditional ordering (at least up to  $\P$ ):

$*$ ,  $\dagger$ ,  $\ddagger$ ,  $\S$ ,  $\|$ ,  $\P$ ,  $**$ ,  $\dagger\dagger$ , and  $\ddagger\ddagger$ .

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}}
```

---

<sup>5-5</sup>For example, Celtic studies.

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`

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
```

## 16.2 MARGINAL NOTES

Some marginalia can also be considered to be kinds of floats. The class provides the standard margin notes via `\marginpar`.

```
\marginpar[<left-text>]{<text>}
\reversemarginpar
\normalmarginpar
```

Just as a reminder, the `\marginpar` macro puts *<text>* into the margin alongside the type-block — the particular margin depends on the document style and the particular page. Following the declaration `\normalmarginpar`, which is the default, the selected margin is:

- Two columns: the note appears in the margin next to the column.
- Single column, two sided: the note appears in the outside margin (left on verso pages and right on odd numbered pages).
- Single column, one sided: the note appears in the right margin.

If the optional argument *<left-text>* is given then this will be used for the text if the note appears in the left margin, otherwise *<text>* is used no matter which margin. The note text is typeset in a parbox.

Following the `\reversemarginpar` declaration the default margins are reversed. The placement is determined by the declaration in effect at the end of the paragraph containing the `\marginpar` command. LaTeX tries to align the the top line of the note with the line in the main text where the command was given. However, if this would overlap a previous note it will be moved down, and possibly below, the page; a warning message is printed on the terminal if a note is moved.

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

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.

### 16.3 SIDE NOTES

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

```
\sidepar[<left>]{<right>}
\sideparvshift
```

The `\sidepar` macro is similar to `\marginpar` except that it produces side notes that do not float — they may overlap.

The same spacing is used for both `\marginpar` and `\sidepar`, namely the lengths `\marginparsep` and `\marginparwidth`.

The length `\sideparvshift` can be used to make vertical adjustments to the position of `\sidepar` notes. By default this is set to a value of 0pt which should align the top of the note with the text line.

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

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

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

```
\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, 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%
 \checkoddpages%
 \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`.

## 16.4 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.<sup>6</sup>

```
\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.

```
\ifsidebaroneside
\sidebarmargin{<margin>}
```

On two column pages sidebars are put in the margin next to the column. On one column pages sidebars are put into the right hand margin when `\ifsidebaroneside` is true. When it is false the margin is controlled by the `\sidebarmargin`, the argument to which may be `left`, `right`, `inner` or `outer` with the obvious meanings (`\sidebarmargin{right}` and `\sidebarmargintrue` are equivalent).

```
\sidebarfont \sidebarform
```

The sidebar `<text>` is typeset using the `\sidebarfont`, whose initial definition is

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

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

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

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 \sidebarmhsep \sidebarmvsep
```

The `<text>` of a `\sidebar` is typeset in a column of width `\sidebarwidth` and there is a horizontal gap of `\sidebarmhsep` between the main text and the sidebar. The length `\sidebarmvsep` 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>}
```

<sup>6</sup> Donald Arseneau's help has been invaluable in getting the sidebar code to work.

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  $\langle height \rangle$ , without making any allowance for `\sidebartopsep`. The  $\langle length \rangle$  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{ $\langle hsep \rangle$ }{ $\langle width \rangle$ }{ $\langle vsep \rangle$ }{ $\langle topsep \rangle$ }{ $\langle font \rangle$ }{ $\langle height \rangle$ }`

The `\setsidebars` command can be used to set the sidebar parameters. `\sidebarhsep` is set to  $\langle hsep \rangle$ , `\sidebarwidth` is set to  $\langle width \rangle$ , `\sidebarvsep` is set to  $\langle vsep \rangle$ , `\sidebartopsep` is set to  $\langle topsep \rangle$ , `\sidebarfont` is set to  $\langle font \rangle$ , and finally `\setsidebarheight` is used to set the height to  $\langle height \rangle$ . The default is:

```
\setsidebars{\marginparsep}{\marginparwidth}{2\onelineskip}%
{0pt}{}{\textheight}
```

The empty  $\langle font \rangle$  argument means that the normal body font will be used. 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'.



# Seventeen

---

## 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.

### 17.1 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` (analogous to the `\item` command in ordinary list environments). By default, the  $\langle source \rangle$  is typeset at the bottom right of the  $\langle text \rangle$ .

## 17.2 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{ $\langle flush \rangle$ }
```

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  $\langle flush \rangle$  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{ $\langle flush \rangle$ }
```

In order to avoid bad line breaks, the epigraph  $\langle text \rangle$  is normally typeset `raggedright`. The  $\langle flush \rangle$  argument to the `\epigraphtextposition` declaration controls the  $\langle text \rangle$  typesetting style. By default this is `flushleft` (which produces `raggedright` text). The sensible values are `center` for centered text, `flushright` for `raggedleft` text, and `flushleft` for normal justified text.

If by any chance you want the  $\langle text \rangle$  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  $\langle text \rangle$  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{ $\langle flush \rangle$ }
```

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 `flushleft`.

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 desirable 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}}
```

### 17.3 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  $\langle text \rangle$  for printing at  $\langle distance \rangle$  below the header on a page.  $\langle text \rangle$  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  $\langle distance \rangle$  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  $\langle distance \rangle$  value. The value for  $\langle distance \rangle$  can be either a integer or a real number. The units are in terms of the current value for `\unitlength`. A typical value for  $\langle distance \rangle$  for a single line quotation and source for a `\chapter*` might be about 70 (points). A positive value of  $\langle distance \rangle$  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  $\langle text \rangle$  argument is put into a minipage of width `\epigraphwidth`. If you use something other than `\epigraph` or `epigraphs` for the  $\langle text \rangle$  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[\langle distance \rangle]{\langle text \rangle}
\epigraphpicture
```

The `\epigraphforheader` macro takes the same arguments as `\epigraphhead` but puts  $\langle text \rangle$  into a zero-sized picture at the coordinate position  $(0, -\langle distance \rangle)$ ; 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  $\langle text \rangle$  argument for `\epigraphforheader` need not be an `\epigraph`, it can be arbitrary text.

```
\dropchapter{\langle length \rangle}
\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
```

### 17.3.1 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}
...
```

# *Eighteen*

---

## 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 undesirable 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.,

```
\noindent\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<sup>1</sup>:

```
\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.

## 18.1 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}[\langle length \rangle] ... \end{verse}
\versewidth
```

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

<sup>1</sup> See the *TeXbook* for how to use this.

`\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 18.1

```
\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.
```

## Typeset example 18.1: Phantom text in verse

|           |                                                 |
|-----------|-------------------------------------------------|
|           | ...                                             |
| Ingeborg  | His judgement rendered, he dissolved the Thing. |
| Fridthjof | 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.                            |
| Ingeborg  | And Ingeborg leave?                             |
| Fridthjof | Nay, nay, I leave thee not, thou goest too.     |
| Ingeborg  | Impossible!                                     |
| Fridthjof | O! hear me, ere thou answerest.                 |

`\vleftofline{<text>}`

A verse line may start with something, for example open quote marks, where it is desirable that it is ignored as far as the alignment of the rest of the line is concerned<sup>2</sup> — 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 18.2

```
\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 18.3

```
\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. \
```

<sup>2</sup> Requested by Matthew Ford who also provided the example text.

## Typeset example 18.2: 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 18.3: 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 ... ” ’ ”

---

```
Your sons and daughters will prophesy, \\
\ldots \\
And everyone who calls \ldots ’ ’\,’
\end{verse}
```

**18.1.1 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  $n$ th 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  $\langle digits \rangle$  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.

### 18.1.2 Numbering

```
\flagverse{ $\langle flag \rangle$ }
\leftskip
```

Putting `\flagverse` at the start of a line will typeset  $\langle flag \rangle$ , for example the stanza number, ending at a distance `\leftskip` before the line. The default for `\leftskip` is 3em.

The lines in a poem may be numbered.

```
\linenumberfrequency{ $\langle nth \rangle$ }
\setverselinenums{ $\langle first \rangle$ }{ $\langle startat \rangle$ }
```

The declaration `\linenumberfrequency{ $\langle nth \rangle$ }` will cause every  $\langle nth \rangle$  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  $\langle first \rangle$  and the first printed number shall be  $\langle startat \rangle$ . 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{ $\langle font-decl \rangle$ }
```

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
\rightskip
```

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 `\rightskip` (default 1em) into the margin.

## 18.2 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.

### 18.2.1 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.

### 18.2.2 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}
```

### 18.3 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}
```

`\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,<br>Excavating for a mine,<br>Lived a miner, forty-niner,<br>And his daughter, Clementine. | 2<br>4 |
| CHORUS | Oh my darling, Oh my darling,<br>Oh my darling Clementine.<br>Thou art lost and gone forever,                       | 6      |

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 Carroll, *Alice's Adventures in Wonderland*, 1865

# Nineteen

---

## 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.

### 19.1 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{snugshaded} text \end{snugshaded}
```

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<sup>+</sup>07]. In the `snugshaded` environment the box clings more closely to its contents than it does in the `shaded` environment.

Be aware that these boxes are somewhat delicate; they do not work in all circumstances. For example they will not work with the `multicol` package [Mit98], 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}{\fboxsep}
```

```
\setlength{\FrameSep}{3\fboxsep}
```

The length `\FrameHeightAdjust` is the height of the top of the frame above the baseline at the top of a page; its default value is 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{\colorbox{\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 desirable. Doing this takes a bit more work than I have shown so far. This first part is from a posting to CTT by Donald Arseneau.<sup>1</sup>

<sup>1</sup> On 2003/10/24 in the thread *framed.sty w/heading?*

```

\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{\FrameCurrent@Lab}{##1}%
 \global\let\FrameCurrent@Lab\FrameNext@Lab
 \global\let\FrameNext@Lab\FrameCont@Lab
 }%
 \global\let\FrameCurrent@Lab\FrameFirst@Lab
 \global\let\FrameNext@Lab\FrameFirst@Lab
 \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 succeeding 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:

```

\rlap{\centerline{\strut#1}}\nobreak\nointerlineskip% centered

```

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{\FrameCurrent@Lab}{##1}
 \global\let\FrameCurrent@Lab\FrameNext@Lab
 \global\let\FrameNext@Lab\FrameCont@Lab
 }%
 \global\let\FrameCurrent@Lab\FrameFirst@Lab

```

```

\global\let\FrameNext@Lab\FrameFirst@Lab
\MakeFramed{\hsize\textwidth
 \advance\hsize -2\FrameRule
 \advance\hsize -2\FrameSep
 \FrameRestore}}%
{\endMakeFramed}

```

```
\begin{titledframe}[\langle cont-title \rangle]{\langle title \rangle} 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.<sup>2</sup> 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 `qframe` 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 expectations better than `framed` or `shaded`. Of course, it does depend on what your expectations are.

## 19.2 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.

<sup>2</sup> Donald Arseneau has said that he may put something similar in a later version of the `framed` package.

`\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:

```
\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].

### 19.3 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 outputting 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 `\setverbatim` declaration 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<sup>3</sup> 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 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'\%}` which will make `'/'` the end marker.

---

<sup>3</sup> This version of the `verbatim` environment is heavily based on the `verbatim` package [SRR99] but does provide some extensions.

### 19.3.1 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 page-break, 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}{%
 \bverpage>true
 \renewcommand{\bvtoprulehook}{\hrule \nobreak \vskip-.1pt}%
 \renewcommand{\bvleftsidehook}{\vrule}%
 \renewcommand{\bvrightsidehook}{\vrule}%
 \renewcommand{\bvendrulehook}{\hrule}%
 \renewcommand{\bvtopmidhook}{\rule{0pt}{2\fbboxsep} \hss}%
}
```

The `\nobvbox` turns off all boxing, and is defined as

```
\newcommand{\nobvbox}{%
 \bverpage>false
 \renewcommand{\bvtoprulehook}{}%
 \renewcommand{\bvleftsidehook}{}%
 \renewcommand{\bvrightsidehook}{}%
 \renewcommand{\bvendrulehook}{}%
 \renewcommand{\bvtopmidhook}{\rule{0pt}{2\fbboxsep} \hss}%
}
```

The definitions of the other styles, `\bvtopandtail` and `\bvslides`, are intermediate between `\bvbox` and `\nobvbox` in the obvious manner.

|                                                                                                                                                          |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>\linenumberfrequency{&lt;nth&gt;} \resetbvlinenumber \setbvlinenums{&lt;first&gt;}{&lt;startat&gt;} \linenumberfont{&lt;font declaration&gt;}</pre> |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|

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 `n`th 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.

### 19.3.2 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.

### 19.3.3 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 `\normalfont`.

```
\newenvironment{smallverbatim}%
 {\setverbatimfont{\normalfont\ttfamily\small}}%
 \verbatim}%
\endenvironment
```

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{\hspace{\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.

## 19.4 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 an 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
```

```
\newcommand{\atstreamopenmyout}{...}
\newoutputstream{myout}
\newinputstream{myin}
```

```
\newcommand{\atstreamclosemyout}{}
\newcommand{\atstreamopenmyin}{}
\newcommand{\atstreamclosemyvin}{}

```

$$\backslash \text{IfStreamOpen}\{\langle stream \rangle\}\{\langle true\text{-code} \rangle\}\{\langle false\text{-code} \rangle\}$$

### 19.4.1 Writing to a file

```
\openoutputfile{<filename>}{<stream>}
\closeoutputstream{<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}`

Writing verbatim text to a file is treated specially as it is likely to be the most common usage.

```
\begin{verbatim}\{\langle file \rangle\} anything \end{verbatim}
\begin{writetext}\{\langle stream \rangle\} anything \end{writetext}
```

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.

### 19.4.2 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>}`.

```
\readline{<stream>}
```

The `\readline` 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 `\readline` 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.

### 19.4.3 Example: endnotes

Books like biographies often quote sources for quotations by the subject, or sources for statements of fact and so on, at the end of the book or chapter. These are often like a collected set of footnotes. The example shows a somewhat rough and ready approach to implementing endnotes.

Typically endnotes come in one of two forms: they are like a normal footnote, except that the note text is on another page, or; there is no mark in the body of the text and the note is identified via a small quote from the text and its page number. The example is for the footnote-like form and for endnotes collected at the end of the document, with an appropriate heading to distinguish notes from different chapters.

We have to be careful in choosing names for the macros we will be defining for endnotes. Remember, you cannot use `\newcommand` to define a new command whose name starts `\end...`, so `\endnote` appears to be out. However, the TeX primitive `\def` command does let you define a command starting with `\end...`. The syntax of the `\def` command, like that of many of TeX macros, looks strange to LaTeX eyes. The major disadvantage in using `\def` is that it will merrily overwrite any previous definition with the same name (the LaTeX `\newcommand` won't let you do that). I could use `\def` for an `\endnote` macro, like

```
\long\def\endnote#1{...}
```

I won't do that, though, as there is at least one LaTeX class that includes a `note` environment and that means that `\endnote` is already defined in that class. To avoid potential pitfalls like that I'll use `\enote` rather than the more evocative `\endnote`.

We need a new counter for the endnotes, starting afresh with each chapter, and to print in arabic numerals.

```
\newcounter{enote}[chapter]
\renewcommand{\theenote}{\arabic{enote}}
```

And we need a macro to typeset the text of the note. This will take two arguments, the number of the note, and the text.

```
\DeclareRobustCommand{\enotetext}[2]{%
 \par\noindent #1 #2\par
 \vspace{\baselineskip}}
```

This makes sure that it starts a new non-indented paragraph, then typesets the first argument (the number) as a superscript and then processes the second argument (the text of the note). After that it makes sure that any paragraph is ended and puts some vertical space in case there is another note following.

The basic idea is to define a command, `\enote{<text>}`, like `\footnote`, that will write `<text>` to a file which will be read in later to typeset the `<text>`.

To this end, we need an output stream, and we will use a file with extension `ent`, the first part of the file name being the name of the LaTeX source file; this is available via the `\jobname` macro.

```
\newoutputstream{notesout}
\openoutputfile{\jobname.ent}{notesout}
\newcommand{\printendnotes}{%
 \closeoutputstream{notesout}%
 \input{\jobname.ent}}
```

The `\printendnotes` macro can be called at the appropriate place in the document to print any endnotes. It closes the output file and then inputs it to print the endnotes.

As well as putting the notes into the file we are also going to add a heading indicating the chapter. Rather than invent a completely new kind of heading I'll simply use `\subsection*` — the starred form so that there will be no ToC entry.

```
\DeclareRobustCommand{\enotehead}[1]{%
 \subsection*{Notes for chapter #1}}
```

The argument to the `\enotehead` macro is the number of a chapter. Also needed is a method for determining when this heading should be added to the endnote file. One simple way is using a counter holding the chapter number. Initialise the counter to something that is an invalid chapter number.

```
\newcounter{savechap}
```

```
\setcounter{savechap}{-1000}
```

We have the pieces ready, and all that remains is to define the `\enote` macro, which will take one argument — the text of the note.

```
\newcommand{\enote}[1]{%
 \refstepcounter{enote}% increment the counter
 \theenote% typeset it as a superscript
 \ifnum\value{savechap}=\value{chapter}\else % in a new chapter
 \setcounter{savechap}{\value{chapter}% save the number
 \addtostream{notesout}{\enotehead{\thechapter}}% the heading
 \fi
 \addtostream{notesout}{\enotetext{\theenote}{#1}}}
```

`\enote`, which is used just like `\footnote`, increments the counter for endnotes, typesets that as a superscript, and then writes the `\enotetext` command to the endnotes file. Entries in the `ent` file will look like:

```
...
\enotehead{3} % for chapter 3
\enotetext{1}{First end note in chapter 3.}
\enotetext{2}{The next end note.}
...
```

You can try this, perhaps changing the definition of `\enotetext` to give a better looking presentation of an endnote. There is, however, a caveat if you use `\enote`.

**Question 3.** What is the caveat?

If you can't think what it might be, don't worry as it will be dealt with in another example.

#### 19.4.4 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 preceded 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 *n*th 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 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}
```

#### 19.4.5 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 `\ifdraft` is true for a draft mode document).

```

\ifdraft % 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 \ifdraft ... \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.

## 19.5 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.

**Question 3.** If LaTeX writes text out to an external file which will be read by LaTeX at some time, any fragile commands in the text must be `\protected`.



# Twenty

---

## 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.

### 20.1 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 current 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*<sup>1</sup> `\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.

---

<sup>1</sup> 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 16.1 on page 277.
```

```
\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 §20.1 in Chapter 20.

## 20.2 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 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 20.1: 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 335 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 20.1

```
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?.
```

```
\headnameref \tocnameref
\namerefon \namerefoff
```

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.

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 `\nameref on`, which is the default, enables name referencing.

# Twenty-one

---

## 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 some features of such matter that is not in the standard classes.

### 21.1 BIBLIOGRAPHY

Just as a reminder the bibliography is typeset within the `thebibliography` environment.

```
\bibname
\begin{thebibliography}{\langle exlabel \rangle}
\bibitem ...
\end{thebibliography}
```

The environment takes one required argument,  $\langle exlabel \rangle$ , 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[\langle detail \rangle]{\langle labstr-list \rangle}
```

Within the text you call out a bibliographic reference using the `\cite` command, where  $\langle labstr-list \rangle$  is a comma-separated list of identifiers for the cited works; there must be no spaces in this list. The optional  $\langle detail \rangle$  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 \tex\ 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 by 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 overridden 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}
```

### 21.1.1 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<sup>+</sup>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.

## 21.2 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.

### 21.2.1 Printing an index

|                                                                 |
|-----------------------------------------------------------------|
| <pre>\makeindex[<i>file</i>]<br/>\printindex[<i>file</i>]</pre> |
|-----------------------------------------------------------------|

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 `file` which has not been declared via `\makeindex`. After the declaration `\reportnoidxfile` LaTeX will whinge about any attempts to write to an unopened file.

### 21.2.2 Preparing an index

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 omission 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:

|                                   |                 |
|-----------------------------------|-----------------|
| <code>\item</code> bridge, 2,3,7  | bridge, 2, 3, 7 |
| <code>\subitem</code> railway, 24 | railway, 24     |
| <code>\subsubitem</code> Tay, 37  | Tay, 37         |

if the Tay Bridge<sup>1</sup> was mentioned on page 37.

<sup>1</sup> A railway (railroad) bridge in Scotland that collapsed in 1879 killing 90 people. The disaster lives for ever in the

### 21.2.3 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.<sup>2</sup> 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 containing 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  $\langle stuff \rangle$  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<sup>+</sup>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 21.1. 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}
```

*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:

---

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.
```

<sup>2</sup> This deficiency in xindy was discovered by Frederic Connes, who also provided the `\xindyindex` command.

Table 21.1: MakeIndex configuration file input parameters

| Keyword                          | Default                     | Description                                                                                                                                                     |
|----------------------------------|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>keyword (s)</code>         | <code>"\\indexentry"</code> | The argument to this command is a MakeIndex index entry                                                                                                         |
| <code>arg_open (c)</code>        | <code>'{'</code>            | Argument start delimiter                                                                                                                                        |
| <code>arg_close (c)</code>       | <code>'}'</code>            | Argument end delimiter                                                                                                                                          |
| <code>range_open (c)</code>      | <code>' ('</code>           | Start of an explicit page range                                                                                                                                 |
| <code>range_close (c)</code>     | <code>') '</code>           | End of an explicit page range                                                                                                                                   |
| <code>level (c)</code>           | <code>'!'</code>            | Character denoting a new subitem level                                                                                                                          |
| <code>actual (c)</code>          | <code>'@'</code>            | Character denoting that the following text is to appear in the actual index file                                                                                |
| <code>encap (c)</code>           | <code>' '</code>            | Character denoting that the rest of the argument is to be used as an encapsulating command for the page number                                                  |
| <code>quote (c)</code>           | <code>'\"'</code>           | Character that escapes the following character                                                                                                                  |
| <code>escape (c)</code>          | <code>'\\'</code>           | 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. |
| <code>page_compositor (s)</code> | <code>"-"</code>            | Composite number separator                                                                                                                                      |

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

|       |                                       |                                               |
|-------|---------------------------------------|-----------------------------------------------|
| p. v: | <code>\index{Alf}</code>              | <code>\indexentry{Alf}{v}</code>              |
| p. 1: | <code>\index{Alf}</code>              | <code>\indexentry{Alf}{1}</code>              |
| p. 2: | <code>\index{Alf}</code>              | <code>\indexentry{Alf}{2}</code>              |
| p. 3: | <code>\index{Alf}</code>              | <code>\indexentry{Alf}{3}</code>              |
| p. 5: | <code>\index{Alfabet see{Bet}}</code> | <code>\indexentry{Alfabet see{Bet}}{5}</code> |
| p. 7: | <code>\index{Alf@\textit{Alf}}</code> | <code>\indexentry{Alf@\textit{Alf}}{7}</code> |
|       | <code>\index{Bet textbf}</code>       | <code>\indexentry{Bet textbf}{7}</code>       |
| p. 8: | <code>\index{Alf!Bet!Con}</code>      | <code>\indexentry{Alf!Bet!Con}{8}</code>      |
| p. 9: | <code>\index{Alf!Dan}</code>          | <code>\indexentry{Alf!Dan}{9}</code>          |

Figure 21.1: Raw indexing: (left) index commands in the source text; (right) idx file entries

|                                          |                         |
|------------------------------------------|-------------------------|
| <code>\begin{theindex}</code>            |                         |
| <code>\item Alf, v, 1-3</code>           | Alf, v, 1-3             |
| <code>\subitem Bet</code>                | Bet                     |
| <code>\subsubitem Con, 8</code>          | Con, 8                  |
| <code>\subitem Dan, 9</code>             | Dan, 9                  |
| <code>\item \textit{Alf}, 7</code>       | <i>Alf</i> , 7          |
| <code>\item Alfabet, \see{Bet}{5}</code> | Alfabet, <i>see</i> Bet |
| <code>\indexspace</code>                 |                         |
| <code>\item Bet, \textbf{7}</code>       | <b>Bet</b> , 7          |
| <code>\end{theindex}</code>              |                         |

Figure 21.2: Processed index: (left) alphabeticized ind file; (right) typeset index

`\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 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
```

### 21.2.4 Controlling MakeIndex output

Table 21.2 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:

```
% MakeIndex style file memman.ist

% @ is a valid character in some entries, use ? instead
actual '??'
```

Table 21.2: 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.

```
% 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.

### 21.2.5 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 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
```

### 21.3 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[\langle file \rangle]`

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 `\langle file \rangle` argument the file `file.glo` will be opened. A glossary `glo` file is analagous to an index `idx` file.

`\printglossary[\langle file \rangle]`

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[\langle file \rangle](\langle key \rangle){\langle term \rangle}{\langle desc \rangle}`

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

By using the optional `\langle file \rangle` 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 facilities 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.

### 21.3.1 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 "\\end{theglossary}\\n"
item_0 "\\n\\glossitem"
delim_0 "{\\memglnum{"
encap_suffix "}}}"
%%% Input style parameters
keyword "\\glossaryentry"
```

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

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

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

The preamble and postamble lines tell the program to start and end its output file with `\begin{theglossary}` and `\end{theglossary}`, respectively. The `item_0` tells the program to start each output entry with `\glossitem`. The `delim_0` says that `{\\memglnum{` could 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 21.1 and 21.2 with the exception of the output file page number setting keywords.

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  $\langle num \rangle$  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 #4}%
 \par\hangindent 2em
 \ifdim\wd\@tempboxa<0.8\linewidth
 #1 \space #2 #3 \dotfill #4\relax
 \else
 #1 \dotfill #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

```

## 21.4 ENDNOTES

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. The following was originally supplied as the `pagenote` package [Wil04b].

```
\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 note file which is called `\jobname.ent`.

In the body of the text 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. 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 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 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 21.3.

For notes at the end of each chapter:

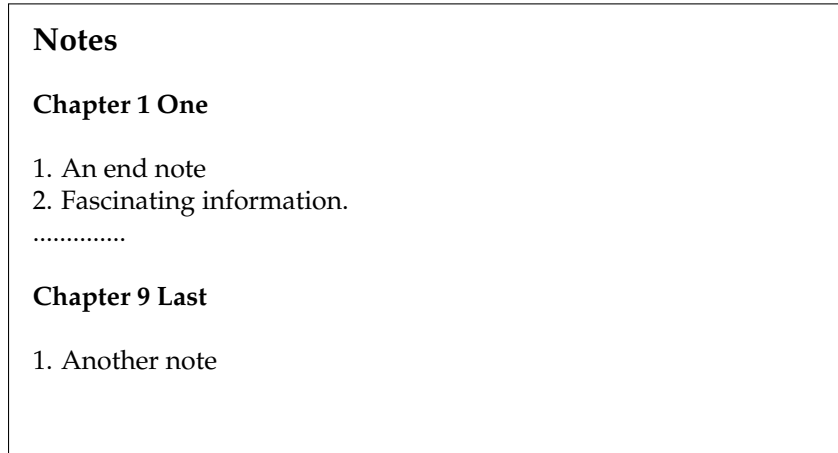


Figure 21.3: Example endnote listing

```

\documentclass[...]{memoir}
...
\makepagenote
...
\begin{document}
\chapter{One}
... \pagenote{An end note.} ...
...
\printpagenotes*
\chapter{Last}
... \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. These declarations should be put in your preamble.

Because of how TeX writes information to files, when the `\notepageref` declaration is used there must be no notes on the page where `\printnotes` or `\printnotes*` closes the file. If necessary, a `\clearpage` or similar must be used before the print command.

```
\notesname
\notedivision
```

When `\printnotes` (or `\printnotes*`) is called the first thing it does is call the macro `\notedivision`. By default this is defined as:

```
\newcommand*\notesname{\Notes}
\newcommand*\notedivision{\chapter{\notesname}}
```

In other words, it will print out a heading for the notes that will be read from the `ent` file. `\print...` then closes the file for writing and after this `\inputs` it to get and process the notes.

#### 21.4.1 Changing the appearance

```
\notenumintext{<num>}
\notenuminnotes{<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. In the note listing `\notenuminnotes` is used to print the number of a note. The default definitions are:

```
\newcommand*\notenumintext[1]{#1}
\newcommand*\notenuminnotes[1]{\normalfont #1.\space}
```

```
\noteentry{<notenum>}{<id>}{<text>}{<pagenum>}
\prenoteinnotes
\postnoteinnotes
```

The `\pagenote` macro writes `\noteentry`, with the appropriate values for the arguments, to the file, where `<notenum>` is the note number (from the `pagenote` counter), `<id>` and `<text>` are as supplied to `\pagenote`, and if the `\notepageref` declaration option is used, `<pagenum>` is the page number, otherwise it is empty. The `\noteentry` macro controls the typesetting of the note.

The default definition of `\noteentry` is

```
\newcommand{\noteentry}[4]{%
 \prenoteinnotes
 \noteidinnotes{#1}{#2}\pageinnotes{#4}\noteinnotes{#3}%
 \postnoteinnotes}
```

and the definitions of other macros are:

```
\newcommand{\prenoteinnotes}{\par\noindent}
\newcommand{\postnoteinnotes}{\par}
```

so that (the first paragraph of) each note is printed as a non-indented paragraph.

If you would prefer, say, hanging paragraphs try:

```
\renewcommand{\prenoteinnotes}{\par\noindent\hangindent 2em}
```

```
\noteidinnotes{<notenum>}{<id>}
\idtextinnotes{<id>}
\notenuminnotes{<num>}
```

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. These are defined respectively as:

```
\newcommand*{\idtextinnotes}[1]{[#1]\space}
\newcommand*{\notenuminnotes}[1]{\normalfont #1.\space}
```

```
\pageinnotes{<pagenum>}
\printpageinnotes{<pagenum>}
```

The macro `\pageinnotes` controls the printing of a note's page reference. If the `\notepageref` declaration has been used it calls `\printpageinnotes` to do the actual printing. Its definition is:

```
\newcommand*{\printpageinnotes}[1]{%
 (\pagerefname\ #1)\space}
```

```
\noteinnotes{<text>}
```

The macro `\noteinnotes{<text>}` is simply:

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

and is used to print the text of a note.

```
\addtonotes{<text>}
```

The macro `\addtonotes` inserts *<text>* into the file. For example, before the first note in a chapter, `\addtonotes` is used to write the `\pagenotesubhead` command to the 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.

```
\pagenotesubhead{<chapapp>}{<num>}{<title>}
\pnchap \pnschap
```

The macro `\pagenotesubhead` typesets the subheadings in an endnote list. It is inserted into the file via `\addtonotes`. 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}}
```

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 `\printnotes*` at each place you want the current set of notes to be printed.

```
\foottopagenote \pagetofootnote
```

You can have both footnotes and endnotes in the same document. However you may start with all footnotes and later decide you would have preferred endnotes instead, or *vice-versa*. The `\foottopagenote` declaration makes `\footnotes` behave as `\pagenotes`, and `\pagetofootnote` has the opposite effect. 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.

# Twenty-two

---

## 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.

### 22.1 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
```

### 22.2 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  $\oplus$  as its symbol. The `\deleted` macro is for indicating that something has been deleted and uses the  $\neq$  symbol. The macro `\changed` uses the  $\Leftrightarrow$  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.

### 22.3 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.

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

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

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

```
\trimmarks
\marktl \marktr \markbr \markbl
\marktm \markmr \markbm \markml
```

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 `\marktl`, `\marktr`, `\markbr`, and `\markbl` at the top left, top right, bottom right and bottom left corners of the page. The pictures `\marktm`, `\markmr`, `\markbm`, and `\markml` are placed at the top middle, middle right, bottom middle and middle left of the edges of the page. All these `\mark..` 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\marktl\trimmark
 \let\marktr\trimmark
 \let\markbr\trimmark
 \let\markbl\trimmark}
```

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

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

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

`\quarkmarks`  
`\registrationColour{\<mark>}`

Following the declaration `\quarkmarks` and trim marks will be in the style of Quark Xpress registration marks.<sup>1</sup> 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.

## 22.4 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

<sup>1</sup> The code for this was donated by William Adams.

number is 1 for the first page and increases by 1 for each page thereafter. The sheet number is independent of the page number.

|                                                              |
|--------------------------------------------------------------|
| <code>sheetsequence</code><br><code>\thesheetsequence</code> |
|--------------------------------------------------------------|

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

|                                                 |
|-------------------------------------------------|
| <code>lastsheet</code><br><code>lastpage</code> |
|-------------------------------------------------|

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 400 of 561 sheets, and page 364 of 525.*

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\marktm\trimseqpage
```

### 22.5 GATHERINGS OR SIGNATURES

Sometimes publishers request that they be supplied with a total number of pages that meet their planned *gatherings*.<sup>2</sup> 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.

|                                               |
|-----------------------------------------------|
| <code>\leavespergathering{&lt;num&gt;}</code> |
|-----------------------------------------------|

The command `\leavespergathering` ensures that there will be exactly the right number of pages output to make a complete set of gatherings of  $\langle num \rangle$  leaves ( $2\langle num \rangle$  pages) each — if necessary blank pages will be output at the end to make up the correct tally. If  $\langle num \rangle$  is less than two (the default) then no additional pages will be output.

---

<sup>2</sup> There was a thread on CTT, *pagenumber mod 4?* about this in 2008.

## 22.6 TIME

```
\printtime \printtime*
\hmpunct \amname \pmname
```

The `\printtime` command<sup>3</sup> 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: July 12, 2008 at 19:13 (7:13 pm).

The punctuation between the hours and minutes is `\hmpunct` which defaults to a colon (:). The macros `\amname` and `\pmname` hold the abbreviations for *ante meridiem* and *post meridiem*, respectively; 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.

## 22.7 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.

## 22.8 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.

<sup>3</sup> I based the code on a similar macro in *TeX for the Impatient* [AHK90].

```
\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`.

### 22.9 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.

## 22.10 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\thedata{#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.

### 22.11 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
 ...
}
```

### 22.12 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 `chnpage` package [Wil01b]. (The class code and `chnpage` code is similar but not identical. There is a later package, `changepage` [Wil08] which contains code that is identical to the class.)

```
\checkoddpage
\ifoddpage
\strictpagecheck \lazypagecheck
```

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 `\lazypagecheck` 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.

### 22.13 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*{\vfill}]{THIS PAGE LEFT BLANK\vspace*{\vfill}}
```

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<sup>4</sup> 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:

```
\cleartoverso
\tableofcontents
```

## 22.14 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}}` → 370th
- The arabic representation of a counter, e.g., `\ordinal{\thepage}` → 370th

However, if the representation of a counter is not completely in arabic digits, such as `\thesection` which here prints as 22.14, it will produce odd errors or peculiar results if it is used as the argument. For instance:

```
\ordinal{\thesection} → .1422nd
```

### 22.14.1 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}{,}
```

---

<sup>4</sup> Perhaps more robust.

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
\renewcommand{\ordscript}[1]{#1}
\fordinal{1234567} -> 1, 234, 567th
\ordinal{1234567} -> 1234567th
```

This is the `\ordinal{\value{chapter}}` chapter. -> This is the 22<sup>nd</sup> 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 21<sup>st</sup>, `\iindstring` (default: nd) for ordinals like 22<sup>nd</sup>, and `\iiirdstring` (default: rd) for ordinals like 23<sup>rd</sup>.

### 22.14.2 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
```

## Typeset example 22.1: 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)

## Source for example 22.1

```
The minimum number in TeX is \numtoname{-2147483647}
(i.e., \fcardinal{-2147483647})
```

```
\ordinaltoname{<number>}
\ordinaltoName{<number>}
\OrdinalToName{<number>}
```

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

For example:

This is the `\ordinaltoname{\value{chapter}}` chapter. -> This is the twenty-second 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 22.2

```
\renewcommand*{\namenumberand}{ }
\renewcommand*{\namenumbercomma}{ }
The maximum number in TeX is \numtoname{2147483647}
(i.e., \cardinal{2147483647}).
```

## Typeset example 22.2: 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).

```
\minusname \lcminusname \ucminusname
```

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

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

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

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

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

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

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].

### 22.14.3 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  $3/4$ . 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.

## Typeset example 22.3: Varieties of fractions in text

---

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

---

## Typeset example 22.4: Super- and subscripts in text

---

In normal text you can typeset superscripts like H<sup>+</sup> and subscripts like H<sub>2</sub>O without going into math mode.

---

## Source for example 22.3

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.

```
<super>
\textsubscript{<sub>}
```

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

## Source for example 22.4

In normal text you can typeset superscripts like H`\textsuperscript{+}` and subscripts like H`\textsubscript{2}`O without going into math mode.

## 22.15 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` checks if `<num>` is an integer (not less than zero). If it is then `\ifinteger` is set `true`, otherwise it is set `false`.

## 22.16 CHECKING THE PROCESSOR

### 22.16.1 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` 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.

### 22.16.2 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.

```
\ifetex
```

`\ifetex` can be used to determine if `etex` is being used as the underlying engine; it is analogous to `\ifpdf` which tests for pdfLaTeX. For example:

```
\ifetex
 %%% code only processible by etex
\else
 \typeout{etex is not available}
\fi
```

### 22.16.3 Checking for XeTeX

You have been able to use `\ifpdf` to check if pdfLaTeX 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.

```
\RequireXeTeX
```

The class also provides `\RequireXeTeX`, which generates an error if XeTeX is not being used to process the document.

## 22.17 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<sup>5</sup>. 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.

---

5 This statement ignores any attempts to stretch the baseline.

## 22.18 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  $3/18\text{em}$ . The class `\medspace` specifies a medium space of  $4/18\text{em}$ . As mentioned, the kernel macro `\:` inserts a medium space in math mode. The class version can be used in both math and text mode to insert a medium space. Similarly, the class version of `\!` can be used to insert a negative thin space in both text and math mode.

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

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

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

## 22.19 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 `\subparagraph` titles. When sectional titles, like `\paragraph` are run-in, it is customary to end them with a period (or occasionally a colon).

## 22.20 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 22.1 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.

Table 22.1: Defined words and phrases

Macro	Default	Useage
<code>\abstractname</code>	Abstract	title for abstract environment
<code>\alsoname</code>	see also	used by <code>\seealso</code>
<code>\amname</code>	am	used in printing time of day
<code>\appendixname</code>	Appendix	name for an appendix heading
<code>\appendixpagename</code>	Appendices	name for an <code>\appendixpage</code>
<code>\appendixtocname</code>	Appendices	ToC entry announcing appendices
<code>\bibname</code>	Bibliography	title for <code>\thebibliography</code>
<code>\bookname</code>	Book	name for <code>\book</code> heading
<code>\bookrefname</code>	Book	used by <code>\Bref</code>
<code>\chaptername</code>	Chapter	name for <code>\chapter</code> heading
<code>\chapterrefname</code>	Chapter	used by <code>\Cref</code>
<code>\contentsname</code>	Contents	title for <code>\tableofcontents</code>
<code>\figurename</code>	Figure	name for figure <code>\caption</code>
<code>\figurerefname</code>	Figure	used by <code>\fref</code>
<code>\glossaryname</code>	Glossary	title for <code>\theglossary</code>
<code>\indexname</code>	Index	title for <code>\theindex</code>
<code>\lcminusname</code>	minus	used in named number formatting
<code>\listfigurename</code>	List of Figures	title for <code>\listoffigures</code>
<code>\listtablename</code>	List of Tables	title for <code>\listoftables</code>
<code>\minusname</code>	minus	used in named number formatting
<code>\namenumberand</code>	and	used in named number formatting
<code>\namenumbercomma</code>	,	used in named number formatting
<code>\notesname</code>	Notes	title of <code>\notedivision</code>
<code>\pagename</code>	page	for your use
<code>\pagerefname</code>	page	used by <code>\pref</code>
<code>\partname</code>	Part	name for <code>\part</code> heading
<code>\partrefname</code>	Part	used by <code>\Pref</code>
<code>\pmname</code>	pm	used in printing time of day
<code>\sectionrefname</code>	§	used by <code>\Sref</code>
<code>\seename</code>	see	used by <code>\see</code>
<code>\tablename</code>	Table	name for table <code>\caption</code>
<code>\tablerefname</code>	Table	used by <code>\tref</code>
<code>\ucminusname</code>	Minus	used in named number formatting

## 22.21 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 `\textsuperscript` macro typesets its argument like a superscript, so using

```
\textregistered
```

gave the required result®.

## 22.21.1 Two simple macros

There are two trivial macros that can be generally useful.

```
\memjustarg{<text>}
\mengobble{<text>}
```

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

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

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

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

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

## 22.21.2 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.

## 22.22 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.

## 22.22.1 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 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.

### 22.22.2 Inserting code before and after a file, package or class

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

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

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

`\AtBeginFile{<file>}{<code>}`  
`\AtEndFile{<file>}{<code>}`

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

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

`\AtBeginPackage{<pack>}{<code>}`  
`\AtEndPackage{<pack>}{<code>}`  
`\RequireAtEndPackage{<pack>}{<code>}`

The `\AtBeginPackage` command will insert `<code>` just before the `<pack>` package is used. Similarly `\AtEndPackage` will insert the `<code>` immediately after the `<pack>`. The `<pack>`

argument must be the same as used in the corresponding `\usepackage` command, that is, without any extension. The `\At...Package` commands must be issued *before* the corresponding `\pack` is used otherwise nothing will happen.

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

```
\AtBeginClass{\class}{\code}
\AtEndClass{\class}{\code}
\RequireAtEndClass{\class}{\code}
```

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

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

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

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

which caused all sorts of problems.

The kernel scheme looks like this:

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

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

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

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

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

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

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

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

but this seems more trouble than it's worth especially since Heiko Oberdiek has kindly updated `hyperref` so that versions after 2006/11/15 will automatically load the `memhfixc` package.

### 22.23 HEADING HOOKS

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

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

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

```
\membookinfo{<thebook>}{<fortoc>}{<title>}
\membookstarinfo{<title>}
```

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

```
\mempartinfo{<thepart>}{<fortoc>}{<title>}
\mempartstarinfo{<title>}
```

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

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

```
\memchapinfo{<thechapter>}{<fortoc>}{<forhead>}{<title>}
\memchapstarinfo{<fortoc>}{<title>}
\memappchapinfo{<thechapter>}{<fortoc>}{<forhead>}{<title>}
\memappchapstarinfo{<fortoc>}{<title>}
```

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

```
\memsecinfo{<name>}{<thename>}{<fortoc>}{<forhead>}{<title>}
\memsecstarinfo{<name>}{<title>}
```

Hooks for appendix-like page headings.

```
\memapppageinfo{<title>}
\memapppagestarinfo{<title>}
\memleadpageinfo{<pstyle>}{<cmdname>}{<title>}
\memleadpagestarinfo{<pstyle>}{<cmdname>}{<title>}
```

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

```
\mempoeminfo{<title>}
\mempoemstarinfo{<title>}
\memPoemTitleinfo{<thepoem>}{<fortoc>}{<forhead>}{<title>}
\memPoemTitlestarinfo{<fortoc>}{<title>}
```

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

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

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) '22.23', (3) 'Heading hooks', (4) 'Heading hooks', (5) 'Heading hooks'.

### 22.23.1 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.

## 23.1 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.

## 23.2 DESIGN REQUIREMENTS

The *Elements of Typographic Style* is typeset using Minion as the text font and Syntax (a sans font) for the captions. The page layout has been shown diagrammatically in Figure 2.2 on page 20, but further details need to be described for those not fortunate enough to have a copy of their own.

The trimmed page size is 23 by 13.3cm. The fore-edge 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\sffseries` 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.

### 23.3 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 fore-edge.

```
\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.

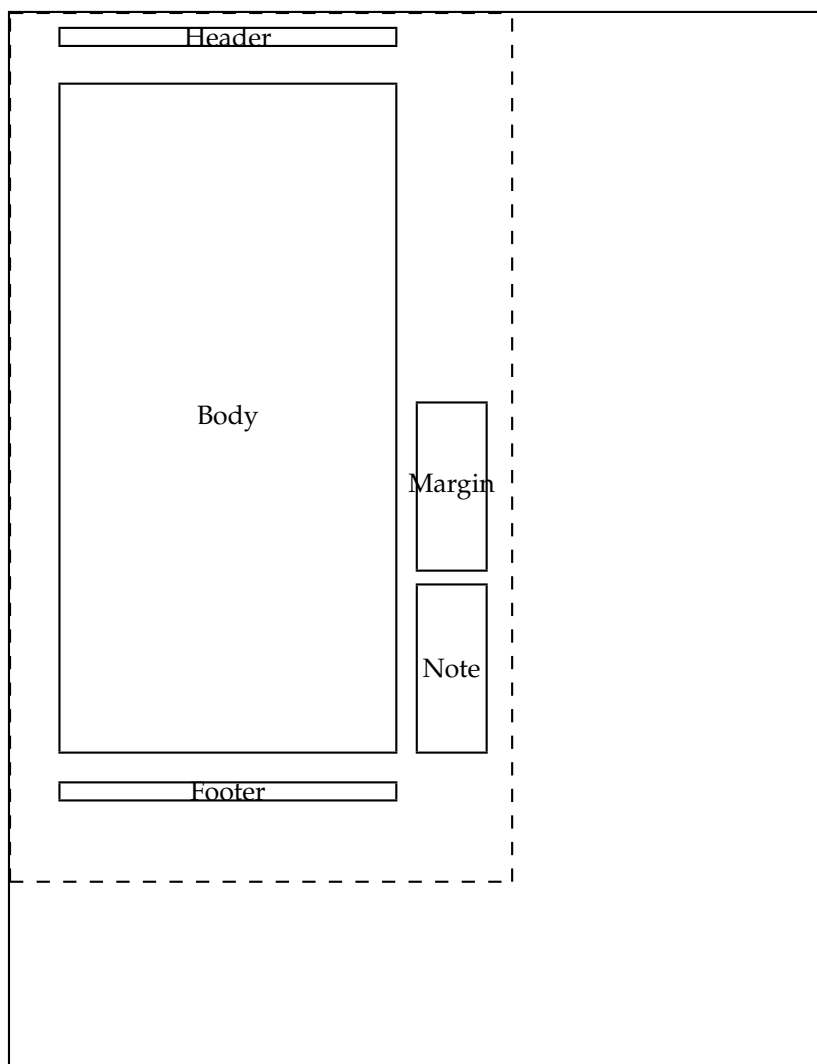
```
\setmarginnotes{17pt}{51pt}{\onelineskip}
```

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 387 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  
design*

Lengths are to the nearest pt.

<code>\stockheight = 795pt</code>	<code>\stockwidth = 614pt</code>
<code>\pageheight = 654pt</code>	<code>\pagewidth = 378pt</code>
<code>\textheight = 502pt</code>	<code>\textwidth = 252pt</code>
<code>\trimtop = 0pt</code>	<code>\trimedged = 236pt</code>
<code>\uppermargin = 54pt</code>	<code>\spinmargin = 38pt</code>
<code>\headheight = 12pt</code>	<code>\headsep = 30pt</code>
<code>\footskip = 36pt</code>	<code>\marginparsep = 17pt</code>
<code>\marginparpush = 12pt</code>	<code>\columnsep = 10pt</code>
<code>\columnseprule = 0.0pt</code>	

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.

## 23.4 SPECIFYING THE SECTIONAL TITLING STYLES

### 23.4.1 *The chapter style*

*Specifying  
the sectional  
titling styles*

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.

The definition of the `chapterstyle` is remarkably simple, as shown below.

```
%% Bringham chapter style
\makechapterstyle{bringham}{%
 \renewcommand{\chapterheadstart}{}
 \renewcommand{\printchaptername}{}
 \renewcommand{\chaptername}{}
 \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.

### 23.4.2 *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 (`\setsubseheadstyle`) 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  
design*

## 23.5 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 `\pwayi` and `\pwayii` 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{\pwayi}{\headsep}
\addtolength{\pwayi}{\topskip}
\addtolength{\pwayi}{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\pwayi){%
 \begin{minipage}[t]{\marginparwidth}
```

```

\raggedright\itshape #1
\end{minipage}}
\end{picture}
}
\makeatother

```

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, -\pwayi)` 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 `\pwayii` to calculate the sum of `\marginparsep` and `\marginparwidth`. Hence:

```

\makeatletter
\setlength{\pwayii}{\marginparsep}
\addtolength{\pwayii}{\marginparwidth}
\newcommand{\bringpicl}[1]{%
\setlength{\unitlength}{1pt}
\begin{picture}(0,0)
\put(-\strip@pt\pwayii, -\strip@pt\pwayi){%
\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, -\pwayi)` 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}}

```

## 23.6 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}
```

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 387, 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}
```

*An example  
design*

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}{\bfseries}
\renewcommand{\cftchapterafterpnum}{\cftparfillskip}
```

## 23.7 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 §B.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`.

You then use the bringhurst package just like any other by putting  
`\usepackage{bringhurst}`  
in your document's preamble.

*Preamble or  
package?*

## 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:

- `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.
- `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.

The package that I used and you most likely do not have is `layouts` [Wil03a]. I used it for all the layout diagrams. For example, Figure 10.28 and Figure 10.29 were drawn simply by:

```
\begin{figure}
\centering
\setlayoutscale{1}
\drawparameterstrue
\drawheading{}
\caption{Displayed sectional headings} \label{fig:displaysehead}
\end{figure}
```

```
\begin{figure}
\centering
\setlayoutscale{1}
\drawparameterstrue
\runinheadtrue
\drawheading{}
```

```
\caption{Run-in sectional headings} \label{fig:runsehead}
\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 6.3).

## 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 `MakIndex` 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\endtightcenter{\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 [Wil07b]. I wanted the environment to be a tight list and started off by defining two helper items.

```
% \@zéroseps sets list before/after skips to minimum values
\newcommand*{\@zéroseps}{\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 `\@zéroseps` 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}{\@zéroseps
 \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.

The two versals in §2.3.4 were typeset with macros defined in `memsty`. The poorer of the two used the `\drop` macro which was written for LaTeX v2.09 by David Cantor and Dominik Wujastyk in 1998. The better used the `\versal` macro. Now, if you want to try your hand at this sort of thing there are some more packages on CTAN. I have found that the `lettrine` package [Fli98] serves my needs.

# B

---

## 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<sup>1</sup> 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 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.

---

<sup>1</sup> Probably also a masochist with plenty of time.

B.1 THE T<sub>E</sub>X PROCESS

As we are delving deeper than normal and because at the bottom it is the T<sub>E</sub>X language that does all the work, it is useful to have an idea of how T<sub>E</sub>X 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 ‘T<sub>E</sub>X’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 T<sub>E</sub>X primitives.

**Execution** The execution process is T<sub>E</sub>X’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 definitions and also builds horizontal, vertical and mathematical lists.

**Layout** The layout processor (T<sub>E</sub>X’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, T<sub>E</sub>X 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.

## B.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.

### B.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 of 4 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.

#### B.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 error 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.

## B.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.

### B.5.1 Numbers

In LaTeX a counter is used for storing an integer number.

Table B.1: Some internal macros for numbers

<code>\m@ne</code>	-1	<code>\z@</code>	0	<code>\@ne</code>	1
<code>\tw@</code>	2	<code>\thr@@</code>	3	<code>\sixt@@n</code>	16
<code>\@xxxii</code>	32	<code>\@cclv</code>	255	<code>\@cclvi</code>	256
<code>\@m</code>	1000	<code>\@Mi</code>	10001	<code>\@Mii</code>	10002
<code>\@Miii</code>	10003	<code>\@Miv</code>	10004	<code>\@MM</code>	20000

```
\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
```

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 B.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`.

### B.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.

$\langle \textit{dimen} \rangle [ = ] \langle \textit{dimension} \rangle$   
 $\langle \textit{skip} \rangle [ = ] \langle \textit{glue} \rangle$

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 infinitely 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 preceded 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
```

## B.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>`, if 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

$$L_n = F_{n-1} + F_{n+1} \quad (\text{B.1})$$

$$5F_n = L_{n-1} + L_{n+1} \quad (\text{B.2})$$

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=\@one
 \the\fibprevprev
\else
 \ifnum \fibtogo=\@two
 \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 `\ifnums` 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 programming 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 `ltxcntrl.dtx`.

# C

---

## The terror 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.

### C.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
1.102 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 `1.` 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\alpha#1{...}'` and `'\alpha'` 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{\alpha}[1]{...}'` and `'\alpha'`.

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  $x_{12}$ . If you were after say,  $x_{23}$  instead, type `x_{2_{3}}`.

```
! Double superscript.
```

*I treat `'x^1^2'` essentially like `'x^1{}^2'`.*

This would produce  $x^{12}$ . If you were after say,  $x^{23}$  instead, type `x^{2^{3}}`.

! (\end occurred inside a group at level ...).

This 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 perhaps 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 '\$!\$'.*

The braces or math mode delimiters 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 assume 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. Please fix it and try again.*

**! I can't write on file '...', please type another.**

TeX couldn't write on a file, you might have misspelled 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 '`\&`' 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 necessary 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 gracefully from this error, if you're careful; see exercise 27.2 in The TeXbook.*

**! Missing delimiter(. 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 delimiters are characters whose \delcode is nonnegative, or you can use '\delimiter <delimiter 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 allotted 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 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 `\makeatother` (see §B.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\al{...}'`, then you must always put `'I'` 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 §B.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 `\#`.

**C.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
1.5 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 §14.4) 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 `...`

## C.2 LATEX ERRORS

LaTeX errors introduce themselves differently from those that TeX finds. For example, if you ever happened 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.

...

```
1.624 \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 delimiter**

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 delimiters 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...`. 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 `amsmath` 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 `\marginpars` 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 occurred 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 are trying to increase the page size too much with the \enlargethispage 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 `\marginpars` 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 §14.4 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 misspelled 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.

### C.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 current page it has been moved to the top of 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 occurred.

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.

## C.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.

...

```
1.954 \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 delimited arguments.
```

```
Empty preamble: '1' used
```

```
 The argument to an array or tabular is empty. The specification {1} 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.

`\spinewidth` and/or `\textwidth` and/or `\foremargin` are too large for `\paperwidth` by ...

The sum of the `\spinewidth` 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.

### C.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 toplevel 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.

---

## Command summary

---

<code>*pt</code>	Class option for an author-defined size for the body font.	65
<code>\:</code>	A medium space (4/18 em).	377
<code>\@docsep{&lt;num&gt;}</code>	Distance, as <num> math units, between dots in the dotted lines in the ToC, etc.	191
<code>\@dottedtocline{&lt;level&gt;}{&lt;indent&gt;}{&lt;numwidth&gt;}</code>	For a ToC, (LoF, LoT) entry at <level> specifies the <indent> and <numwidth> and draws a dotted line between the title and page number.	192
<code>\@fnsymbol{&lt;num&gt;}</code>	Converts <num> to the footnote marker representation.	279
<code>\@makefnmark</code>	Typesets the footnote marker where <code>\footnote</code> is called.	277
<code>\@pnumwidth{&lt;length&gt;}</code>	Space for a page number in the ToC, etc.	191
<code>\@thefnmark</code>	Value of the footnote marker.	277
<code>\@tocrmarg{&lt;length&gt;}</code>	Right hand margin for titles in the ToC, etc.	191
<code>\&gt;[&lt;length&gt;]</code>	Ends a verse line, and adds <length> vertical space.	296
<code>\&gt;*[&lt;length&gt;]</code>	Ends a line while preventing a pagebreak, and adds <length> vertical space.	296
<code>\&gt;[&lt;length&gt;]</code>	Shorthand for <code>\verselinebreak</code> .	297
<code>10pt</code>	Class option for a 10pt body font.	64
<code>11pt</code>	Class option for a 11pt body font.	64
<code>12pt</code>	Class option for a 12pt body font.	64
<code>14pt</code>	Class option for a 14pt body font.	64
<code>17pt</code>	Class option for a 17pt body font.	64
<code>20pt</code>	Class option for a 20pt body font.	64
<code>25pt</code>	Class option for a 25pt body font.	64
<code>30pt</code>	Class option for a 30pt body font.	64
<code>36pt</code>	Class option for a 36pt body font.	65
<code>48pt</code>	Class option for a 48pt body font.	65
<code>60pt</code>	Class option for a 60pt body font.	65
<code>9pt</code>	Class option for a 9pt body font.	64
<code>a3paper</code>	Class option for A3 stock paper size.	63
<code>a4paper</code>	Class option for A4 stock paper size.	63
<code>a5paper</code>	Class option for A5 stock paper size.	63
<code>a6paper</code>	Class option for A6 stock paper size.	63
<code>\abnormalparskip{&lt;length&gt;}</code>	Sets the inter-paragraph spacing to <length>.	101
<code>\abovecaptionskip</code>	Vertical space above a caption.	245
<code>\aboverulesep</code>	Space above a tabular rule.	259

<code>\abslabeldelim{&lt;text&gt;}</code> .....	118
<code>&lt;text&gt;</code> is typeset immediately after <code>\abstractname</code> in a run-in heading.	
<code>\absleftindent</code> Indentation of the left of the abstract text. ....	118
<code>\absnamepos</code> .....	118
Position of a non run-in title for an abstract ( <code>flushleft</code> , <code>center</code> , or <code>flushright</code> ).	
<code>\absparindent</code> Paragraph indent in the abstract environment. ....	118
<code>\absparsep</code> Paragraph separation in the abstract environment. ....	118
<code>\absrightindent</code> Indentation of the right of the abstract text. ....	118
<code>\abstitleskip</code> Space around the title of an abstract. ....	119
<code>\begin{abstract}</code> Environment for typesetting an abstract. ....	117
<code>\abstractcol</code> .....	118
Declares that an abstract in a twocolumn document should be typeset like an unnumbered chapter.	
<code>\abstractintoc</code> .....	118
Specifies that the abstract's title is to be added to the ToC.	
<code>\abstractname</code> An abstract's title. ....	118
<code>\abstractnamefont</code> .....	118
Font for typesetting and abstract's title ( <code>\abstractname</code> ).	
<code>\abstractnum</code> .....	118
Specifies that an abstract is to be typeset like a numbered chapter.	
<code>\abstractrunin</code> .....	118
Specifies that the title of an abstract should be set as a run-in heading.	
<code>\abstracttextfont</code> Font for typesetting the body text of an abstract. ....	118
<code>\addappheadtotoc</code> Adds ToC entry with the title <code>\appendixtocname</code> . ....	124
<code>\addcontentsline{&lt;file&gt;}{&lt;kind&gt;}{&lt;text&gt;}</code> .....	190
Writes heading/caption data to the <code>&lt;file&gt;</code> in the form of the <code>\contentsline</code> macro.	
<code>\added{&lt;change-id&gt;}</code> .....	362
Change mark for something added; <code>&lt;change-id&gt;</code> is put in the margin.	
<code>\addlinespace{&lt;width&gt;}</code> .....	260
Puts extra space, default <code>\defaultaddspace</code> , between a pair of tabular rows.	
<code>\addtocontents{&lt;file&gt;}{&lt;text&gt;}</code> Inserts <code>&lt;text&gt;</code> into <code>&lt;file&gt;</code> (ToC, etc). ....	192
<code>\addtodef{&lt;macro&gt;}{&lt;prepend&gt;}{&lt;append&gt;}</code> .....	367
Inserts <code>&lt;prepend&gt;</code> at the start of the current definition of <code>&lt;macro&gt;</code> and <code>&lt;append&gt;</code> at the end, treating the result as if it had been defined by <code>\renewcommand</code> .	
<code>\addtodef*{&lt;macro&gt;}{&lt;prepend&gt;}{&lt;append&gt;}</code> .....	367
Inserts <code>&lt;prepend&gt;</code> at the start of the current definition of <code>&lt;macro&gt;</code> and <code>&lt;append&gt;</code> at the end, treating the result as if it had been defined by <code>\renewcommand*</code> .	
<code>\addtoiargdef{&lt;macro&gt;}{&lt;prepend&gt;}{&lt;append&gt;}</code> .....	367
Inserts <code>&lt;prepend&gt;</code> at the start of the current definition of <code>&lt;macro&gt;</code> (which takes a single argument) and <code>&lt;append&gt;</code> at the end, treating the result as if it had been defined by <code>\renewcommand</code> .	
<code>\addtoiargdef*{&lt;macro&gt;}{&lt;prepend&gt;}{&lt;append&gt;}</code> .....	367
Inserts <code>&lt;prepend&gt;</code> at the start of the current definition of <code>&lt;macro&gt;</code> (which takes a single argument) and <code>&lt;append&gt;</code> at the end, treating the result as if it had been defined by <code>\renewcommand*</code> .	
<code>\addtonotes{&lt;text&gt;}</code> Inserts <code>&lt;text&gt;</code> into the endnotes ent file. ....	359
<code>\addtopsmarks{&lt;pagestyle&gt;}{&lt;prepend&gt;}{&lt;append&gt;}</code> .....	173
Inserts <code>&lt;prepend&gt;</code> and <code>&lt;append&gt;</code> before and after the current definition of <code>\makepsmarks</code> for <code>&lt;pagestyle&gt;</code> .	
<code>\addtostream{&lt;stream&gt;}{&lt;text&gt;}</code> .....	322
Adds <code>&lt;text&gt;</code> to the file associated with the output <code>&lt;stream&gt;</code> .	
<code>\begin{adjustwidth}{&lt;left&gt;}{&lt;right&gt;}</code> .....	180
Temporarily adds the lengths <code>{&lt;left&gt;}</code> and <code>{&lt;right&gt;}</code> to the left and right margins.	

<code>\begin{adjustwidth*}{\langle left \rangle}{\langle right \rangle}</code> .....	180
A sophisticated form of <code>adjustwidth</code> . Temporarily adds the lengths <code>{\langle left \rangle}</code> and <code>{\langle right \rangle}</code> to the spine and outer margins on odd (recto) pages, and on even (verso) pages adds them to the outer and spine margins, respectively.	
<code>\afterbookskip</code> Spacing below a <code>\book</code> title. ....	125
<code>\afterchapskip</code> Space after chapter title. ....	129
<code>\afterchapternum</code> .....	129
Called after printing a chapter number; by default inserts <code>\midchapskip</code> space.	
<code>\afterchaptertitle</code> .....	129
Called after printing a chapter title; by default inserts <code>\afterchapskip</code> space.	
<code>\afterepigraphskip</code> Vertical space after an epigraph. ....	289
<code>\afterbookskip</code> Spacing below a <code>\part</code> title. ....	125
<code>\afterPoemTitle</code> Called after printing the title of a <code>\PoemTitle</code> . ....	301
<code>\afterPoemTitlenum</code> .....	301
Called after printing the number of a <code>\PoemTitle</code> .	
<code>\afterPoemTitleskip</code> Vertical space after a poem title ....	302
<code>\afterXtitle</code> .....	195
Generic macro called after typesetting the title of the ‘X List of’.	
<code>\aliaspagestyle{\langle alias \rangle}original</code> .....	167
Defines the <code>\langle alias \rangle</code> pagestyle to be the same as the <code>\langle original \rangle</code> pagestyle.	
<code>\alsoname</code> Wording for a <i>see also</i> index entry. ....	346
<code>\begin{altverse}</code> Alternate lines in the stanza are indented. ....	299
<code>\amname</code> .....	365
Abbreviation for ante meridiem used in <code>\printtime*</code> (default am)	
<code>\and</code> Use within the argument to <code>\author</code> to separate author’s names. ....	113
<code>\andnext</code> Use within the argument to <code>\author</code> to insert a newline.. ....	113
<code>\anonsubappendices</code> .....	124
Do not precede the sub-appendix number with any name (the default).	
<code>\anyptfilebase</code> .....	65
First part of the name of the <code>clo</code> file for the <code>*pt</code> class option (default mem).	
<code>\anyptsize</code> .....	65
Second part (the pointsize) of the name the <code>clo</code> file for the <code>*pt</code> class option (default 10).	
<code>\begin{appendices}</code> .....	124
An environment form of <code>\appendix</code> ; chapters are restored to their condition (including numbering) after the environment ends.	
<code>\appendix</code> .....	123
Sets chapter numbering to alphabetic and sets the chapter name to <code>\appendixname</code> .	
<code>\appendixname</code> .....	123
Name given to chapters in appendices (default Appendix).	
<code>\appendixpage</code> .....	123
Creates an unnumbered anonymous part-like page with the title <code>\appendixpagename</code> and adds it to the ToC.	
<code>\appendixpage*</code> Like <code>\appendixpage</code> but makes no ToC entry. ....	123
<code>\appendixpagename</code> .....	123
Title used for an <code>\appendixpage</code> (default Appendices).	
<code>\appendixrefname</code> Name for an appendix used by <code>\Aref</code> . ....	334
<code>\appendixtocname</code> .....	124
Title of ToC entry added by <code>\addappheadtotoc</code> (default Appendices).	
<code>\Aref{\langle labstr \rangle}</code> .....	334
Prints a named ( <code>\appendixrefname</code> ) reference to a <code>\labeled</code> appendix.	

<code>\begin{array}[\langle pos \rangle]{\langle preamble \rangle}</code> .....	249
Environment for setting math elements in an array form.	
<code>\arraybackslash</code> Use instead of <code>\\</code> in a tabular column. ....	265
<code>\arraycolsep</code> Half the space between columns in an array. ....	267
<code>\arrayrulewidth</code> .....	267
Width of lines (e.g., <code>\hline</code> , <code>\vline</code> , etc.) in an array or tabular.	
<code>\arraystretch</code> .....	267
Multiplication factor for the normal row spacing in tabulars and arrays.	
<code>\arraytostring{\langle arrayname \rangle}{\langle result \rangle}</code> .....	375
Defines the macro <code>\langle result \rangle</code> to be the sequence of characters in the array <code>\langle arrayname \rangle</code> .	
<code>article</code> Class option for simulating the article class. ....	66
<code>article</code> .....	132
Chapter style similar to a section head in the article class but with different sized fonts.	
<code>\AtBeginClass{\langle pack \rangle}{\langle code \rangle}</code> .....	381
Inserts <code>\langle code \rangle</code> just before the <code>\langle class \rangle</code> class is used.	
<code>\AtBeginFile{\langle file \rangle}{\langle code \rangle}</code> .....	380
Inserts <code>\langle code \rangle</code> just before the <code>\langle file \rangle</code> is input (or included, etc.).	
<code>\AtBeginPackage{\langle pack \rangle}{\langle code \rangle}</code> .....	380
Inserts <code>\langle code \rangle</code> just before the <code>\langle pack \rangle</code> package is used.	
<code>\AtEndClass{\langle class \rangle}{\langle code \rangle}</code> .....	381
Inserts <code>\langle code \rangle</code> just after the <code>\langle class \rangle</code> class is used.	
<code>\AtEndFile{\langle file \rangle}{\langle code \rangle}</code> .....	380
Inserts <code>\langle code \rangle</code> just after the <code>\langle file \rangle</code> is input (or included, etc.).	
<code>\AtEndPackage{\langle pack \rangle}{\langle code \rangle}</code> .....	380
Inserts <code>\langle code \rangle</code> just after the <code>\langle pack \rangle</code> package is used.	
<code>\atendtheglossaryhook</code> .....	353
Vacuous macro called as the first thing by <code>\end{theglossary}</code> .	
<code>\author{\langle text \rangle}</code> .....	113
Used by <code>\maketitle</code> to typeset <code>\langle text \rangle</code> as the document author.	
<code>\autocols[\langle width \rangle]{\langle pos \rangle}{\langle num \rangle}{\langle style \rangle}{\langle entries \rangle}</code> .....	271
Lists the <code>\langle entries \rangle</code> down <code>\langle num \rangle</code> columns in a tabular fashion.	
<code>\autorows[\langle width \rangle]{\langle pos \rangle}{\langle num \rangle}{\langle style \rangle}{\langle entries \rangle}</code> .....	270
Lists the <code>\langle entries \rangle</code> in rows across <code>\langle num \rangle</code> columns in a tabular fashion.	
<code>b3paper</code> Class option for B3 stock paper size. ....	63
<code>b4paper</code> Class option for B4 stock paper size. ....	63
<code>b5paper</code> Class option for B5 stock paper size. ....	63
<code>b6paper</code> Class option for B6 stock paper size. ....	63
<code>\backmatter</code> .....	121
Prohibits sectional numbering and floats, etc., will be numbered continuously.	
<code>\baselineskip</code> (length) .....	82
The normal vertical distance between the baselines of adjacent lines of text (depends on the size of the font).	
<code>\beforebookskip</code> Spacing above a <code>\book</code> title. ....	125
<code>\beforechapskip</code> Space above chapter name and number. ....	129
<code>\beforeepigraphskip</code> Vertical space before an epigraph. ....	289
<code>\beforepartskip</code> Spacing above a <code>\part</code> title. ....	125
<code>\beforePoemTitleskip</code> Vertical space before a poem title. ....	302
<code>\begintheglossaryhook</code> .....	353
Vacuous macro called as the last thing by <code>\begin{theglossary}</code> .	
<code>\belowcaptionskip</code> Vertical space below a caption. ....	245

---

<code>\belowrulesep</code>	Space below a tabular rule.	259
<code>\bfseries</code>	Declaration for using a bold font.	94
<code>bianchi</code>	A two line, centered chapterstyle with rules above and below.	136
<code>\bibintoc</code>	Title of the bibliography will be added to the ToC.	337
<code>\bibitem[⟨label⟩]{⟨labstr⟩}</code>	Introduces an entry in the bibliography. The <code>⟨labstr⟩</code> argument corresponds to a <code>\cite's</code> <code>⟨labstr⟩</code> argument. The optional <code>⟨label⟩</code> overrides the default numerical printed entry label.	339
<code>\bibitem</code>	Starts a new bibliographic entry in a <code>thebibliography</code> listing.	337
<code>\bibitemsep</code>	Vertical space between entries in a bibliography.	339
<code>\bibliography{⟨bibfile-list⟩}</code>	Print the bibliography having used BibTeX to extract entries from the <code>⟨bibfile-list⟩</code> of comma-separated names of bib files.	339
<code>\bibliographystyle{⟨style⟩}</code>	Typeset the bibliographic entries according to <code>⟨style⟩</code> .	339
<code>\biblistextra</code>	Called immediately after the <code>\bibitemlist</code> is set up.	338
<code>\bibmark</code>	Can be used in pagestyles for page headers in a bibliography.	338
<code>\bibname</code>	The title for a bibliography	337
<code>\bibsection</code>	Initialises the bibliography and typesets the title.	338
<code>\bicaption[⟨label⟩]{⟨short1⟩}{⟨long1⟩}{⟨NAME⟩}{⟨long2⟩}</code>	A bilingual caption in a float but only the first added to the 'List of...'. A continued bilingual caption.	235
<code>\bicontcaption{⟨long1⟩}{⟨NAME⟩}{⟨long2⟩}</code>		236
<code>\bionenumcaption[⟨label⟩]{⟨short1⟩}{⟨long1⟩}{⟨NAME⟩}{⟨short2⟩}{⟨long2⟩}</code>	A bilingual caption with both captions numbered in the float but only the first in the 'List of...'. A bilingual caption with both captions numbered in the float and in the 'List of...'. Typesets a numbered book <code>⟨title⟩</code> on a page by itself, adding <code>⟨title⟩</code> to the ToC.	234
<code>\bitwonumcaption[⟨label⟩]{⟨short1⟩}{⟨long1⟩}{⟨NAME⟩}{⟨short2⟩}{⟨long2⟩}</code>		234
<code>\book{⟨title⟩}</code>		122
<code>\booknamefont</code>	Font used by <code>\printbookname</code> for the book name.	126
<code>\booknamenum</code>	Called between printing a book name and number.	126
<code>\booknumfont</code>	Font used by <code>\printbooknum</code> for the book number.	126
<code>\bookpagemark{⟨title⟩}</code>	For setting any marks with the title from a <code>\book</code> for a running header.	127
<code>\bookrefname</code>	Name for a book used by <code>\Bref</code> .	334
<code>\booktitlefont</code>	Font used by <code>\printbooktitle</code> for the title.	126
<code>\bottomrule[⟨width⟩]</code>	Draws a rule across a tabular, default width <code>\heavyrulewidth</code> .	259
<code>\bottomsectionskip</code>	Amount of stretch on a <code>\raggedbottomsection</code> short page.	123
<code>\begin{boxedverbatim}</code>	May put a box around the verbatim material; lines may be numbered and page breaks are allowed.	317
<code>\begin{boxedverbatim*}</code>	May put a box around the verbatim* material; lines may be numbered and page breaks are allowed.	317
<code>\boxedverbatiminput{⟨file⟩}</code>	Acts like <code>boxedverbatim</code> except the contents is read from the <code>⟨file⟩</code> file.	323
<code>\boxedverbatiminput*{⟨file⟩}</code>	Acts like <code>boxedverbatim*</code> except the contents is read from the <code>⟨file⟩</code> file.	323

<code>\Bref{&lt;labstr&gt;}</code> .....	334
Prints a named ( <code>\bookrefname</code> ) reference to a <code>\labeled</code> book.	
<code>bringhurst</code> .....	136
A raggedright, unnumbered, small caps chapterstyle with a textwidth rule below.	
<code>broadsheetpaper</code> Class option for broadsheet stock paper size. ....	64
<code>brotherton</code> .....	136
A chapterstyle like the default but with the number spelled out.	
<code>\bs</code> prints <code>\</code> . ....	383
<code>\bvbox</code> .....	317
Rectangular boxes will be drawn for <code>boxedverbatim</code> environments.	
<code>\bvboxsep</code> .....	317
Separation between text and framing in <code>boxedverbatim</code> environments.	
<code>\bvendofpage{&lt;text&gt;}</code> .....	317
Use <code>&lt;text&gt;</code> as the <code>boxedverbatim</code> page break marker at the bottom of a page.	
<code>\bvendrulehook</code> .....	317
Called at the end of a <code>boxedverbatim</code> environment, and before a pagebreak.	
<code>\bvleftsidehook</code> .....	317
Called before each line in a <code>boxedverbatim</code> environment.	
<code>\bvnumbersinside</code> Line numbers typeset inside a <code>boxedverbatim</code> box. ....	319
<code>\bvnumbersoutside</code> .....	319
Line numbers typeset outside a <code>boxedverbatim</code> box.	
<code>\bvperpagefalse</code> Do not mark page breaks in a <code>boxedverbatim</code> . ....	317
<code>\bvperpagetrue</code> .....	317
Visibly break a <code>boxedverbatim</code> at a page break using <code>\bvtopofpage</code> and <code>\bvendofpage</code> .	
<code>\bvrightsidehook</code> .....	317
Called after each line in a <code>boxedverbatim</code> environment.	
<code>\bvsides</code> .....	317
Draw vertical rules on each side of <code>boxedverbatim</code> environments.	
<code>\bvtopandtail</code> .....	317
Draw horizontal rules before and after <code>boxedverbatim</code> environments.	
<code>\bvtopmidhook</code> .....	317
Called after <code>\bvtoprulehook</code> at the start of a <code>boxedverbatim</code> environment.	
<code>\bvtopofpage{&lt;text&gt;}</code> .....	317
Use <code>&lt;text&gt;</code> as the <code>boxedverbatim</code> page break marker at the top of a page.	
<code>\bvtoprulehook</code> .....	317
Called at the start of a <code>boxedverbatim</code> environment and after a pagebreak.	
 <code>\calccentering{&lt;length&gt;}</code> .....	 181
Sets the <code>&lt;length&gt;</code> command to the value to add/subtract from margins to center text on the physical page.	
<code>\cancelthanksrule</code> .....	116
Specifies that the <code>\footnoterule</code> is to be used from now on.	
<code>\caption[&lt;short&gt;]{&lt;long&gt;}</code> .....	243
Typeset a caption with title <code>&lt;long&gt;</code> , and add it, or <code>&lt;short&gt;</code> instead if given, to a 'List of...'. .	
<code>\captiondelim{&lt;delim&gt;}</code> .....	226
Specifies <code>&lt;delim&gt;</code> to be the delimiter between the number and title in a caption.	
<code>\captionnamefont{&lt;fontspec&gt;}</code> .....	226
Set the font for the first part (name and number) of a caption, upto and including the delimiter.	
<code>\captionstyle[&lt;short&gt;]{&lt;style&gt;}</code> .....	226
Set the paragraph style for the caption. The optional <code>&lt;short&gt;</code> is the style for captions shorter than a full line.	

---

<code>\captiontitlefinal{&lt;text&gt;}</code> .....	227
<i>&lt;text&gt;</i> will be put immediately at the end of a caption title, but will not appear in a ‘List of...’.	
<code>\captiontitlefont{&lt;fontspec&gt;}</code> Set the font for the caption title text. ....	226
<code>\captionwidth{&lt;length&gt;}</code> Set the caption width to <i>&lt;length&gt;</i> . ....	227
<code>\cardinal{&lt;number&gt;}</code> Typesets {<number>} as a cardinal number. ....	370
<code>\begin{center}</code> .....	179
Text to be set raggedleft and raggedright, with each line centered.	
<code>\centering</code> .....	179
Declaration for text to be set raggedleft and raggedright, with each line centered.	
<code>\cftaddnumtitleline{&lt;ext&gt;}{&lt;kind&gt;}{&lt;num&gt;}{&lt;title&gt;}{&lt;page&gt;}</code> .....	202
Writes a <code>\contentsline</code> to the ‘List of...’ <i>&lt;ext&gt;</i> file for a <i>&lt;kind&gt;</i> entry with number <i>&lt;number&gt;</i> and <i>&lt;title&gt;</i> and <i>&lt;page&gt;</i> number.	
<code>\cftaddtitleline{&lt;ext&gt;}{&lt;kind&gt;}{&lt;title&gt;}{&lt;page&gt;}</code> .....	202
Writes a <code>\contentsline</code> to the ‘List of...’ <i>&lt;ext&gt;</i> file for a <i>&lt;kind&gt;</i> entry with <i>&lt;title&gt;</i> and <i>&lt;page&gt;</i> number.	
<code>\cftbeforeKskip</code> .....	198
Generic vertical space before a ‘K’ entry in a ‘List of...’.	
<code>\cftbookbreak</code> Starts a <code>\book</code> entry in the ToC. ....	197
<code>\cftchapterbreak</code> Starts a <code>\chapter</code> entry in the ToC. ....	197
<code>\cftdot</code> The ‘dot’ in the dotted leaders in ‘List of...’. ....	196
<code>\cftdotsep</code> .....	196
The separation between dots in a dotted leader in a ‘List of...’.	
<code>\cftinsertcode{&lt;name&gt;}{&lt;code&gt;}</code> .....	202
Defines Toc (LoF, LoT) <i>&lt;name&gt;</i> insertion to be <i>&lt;code&gt;</i> .	
<code>\cftinserthook{&lt;file&gt;}{&lt;name&gt;}</code> .....	202
Inserts code <i>&lt;name&gt;</i> into <i>&lt;file&gt;</i> (e.g., <i>toc</i> , <i>lof</i> , etc).	
<code>\cftKafterpnum</code> .....	199
Called after typesetting the page number of a ‘K’ entry in a ‘List of...’.	
<code>\cftKaftersnum</code> .....	199
Called immediately after typesetting the number of a ‘K’ entry in a ‘List of...’.	
<code>\cftKaftersnumb</code> .....	199
Called immediately after typesetting the number box of a ‘K’ entry in a ‘List of...’.	
<code>\cftKdotsep</code> .....	199
Separation between dots in a leader between the title and page number of a ‘K’ entry in a ‘List of...’.	
<code>\cftKfont</code> .....	198
Controls the appearance of the number and title of a ‘K’ entry in a ‘List of...’.	
<code>\cftKformatpnum{&lt;pnum&gt;}</code> .....	199
Typesets the page number <i>&lt;pnum&gt;</i> of a ‘K’ entry in a ‘List of...’.	
<code>\cftKindent</code> .....	198
Generic indent of an ‘K’ entry from the left margin in a ‘List of...’.	
<code>\cftKleader</code> .....	199
Leader between the title and page number of a ‘K’ entry in a ‘List of...’.	
<code>\cftKname</code> .....	198
Called as the first element of the ‘K’ entry line in a ‘List of...’.	
<code>\cftKnumwidth</code> .....	198
Generic space for the number of a ‘K’ entry in a ‘List of...’.	
<code>\cftKpagefont</code> Font for the page number of a ‘K’ entry in a ‘List of...’. ....	199
<code>\cftKpresnum</code> .....	199
Called immediately before typesetting the number of a ‘K’ entry in a ‘List of...’.	

<code>\cftlocalchange{&lt;ext&gt;}{&lt;pnumwidth&gt;}{&lt;tocrmarg&gt;}</code> .....	202
Writes commands to the <ext> ‘List of...’ file resetting \@pnumwidth and \@tocrmarg to the specified values.	
<code>\cftnodots</code> .....	196
A separation between dots in a dotted leader in a ‘List of...’ that is too large for any dots to occur.	
<code>\cftpagenumbersoff{&lt;kind&gt;}</code> .....	201
Eliminates page numbers for the <kind> entries in a ‘List of...’.	
<code>\cftpagenumberon{&lt;kind&gt;}</code> .....	201
Reverses the effect of <code>\cftpagenumbersoff</code> .	
<code>\cftparskip</code> Fills the last line in a paragraph in a ‘List of...’ .....	201
<code>\cftparskip</code> The <code>\parskip</code> to be used in a ‘List of...’ .....	197
<code>\cftpартbreak</code> Starts a <code>\part</code> entry in the ToC. ....	197
<code>\cftsetindents{&lt;kind&gt;}{&lt;indent&gt;}{&lt;numwidth&gt;}</code> .....	199
Set the <kind> entry <i>indent</i> to <indent> and its <i>numwidth</i> to <numwidth>.	
<code>\changecaptionwidth</code> .....	227
Captions will be set within the width specified by <code>\captionwidth</code> .	
<code>\changed{&lt;change-id&gt;}</code> .....	362
Change mark for something changed; <change-id> is put in the margin.	
<code>\changeglossactual[&lt;file&gt;]{&lt;char&gt;}</code> .....	352
Specifies <char> as the <i>actual</i> character for glossary <file>.	
<code>\changeglossnum[&lt;file&gt;]{&lt;thecounter&gt;}</code> .....	352
Specifies <thecounter> as the <num> for glossary <file>.	
<code>\changeglossnumformat[&lt;file&gt;]{&lt;format&gt;}</code> .....	352
Specifies <format> as the format for <num> for glossary <file>.	
<code>\changeglossref[&lt;file&gt;]{&lt;thecounter&gt;}</code> .....	352
Specifies <thecounter> as the <ref> for glossary <file>.	
<code>\changemarks</code> Print change marks. ....	361
<code>\chapindent</code> .....	135
A length you can use in a defining chapter style (or anything else).	
<code>\chapnamefont</code> Font used by <code>\printchaptername</code> . ....	129
<code>\chapnumfont</code> Font used by <code>\printchapternum</code> . ....	129
<code>chappell</code> .....	136
A centered chapterstyle with a rule between the number line (in a roman font) and the title line in italics.	
<code>\chapter[&lt;toc-title&gt;][&lt;head-title&gt;]{&lt;title&gt;}</code> .....	122
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 <code>\section</code> of floats.	
<code>\chapter*[&lt;head-title&gt;]{&lt;title&gt;}</code> .....	122
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.	
<code>\chapterheadstart</code> .....	129
Called at start of printing a chapter header; by default inserts <code>\beforechapskip</code> space.	
<code>\chapternamenum</code> .....	129
Inserted between printing the chapter name and number. It is usually a space.	
<code>\chapterprecis{&lt;text&gt;}</code> Prints <text> and also adds it to the ToC. ....	150
<code>\chapterprecishere{&lt;text&gt;}</code> Typesets <text> for a chapter precis. ....	151
<code>\chapterprecistoc{&lt;text&gt;}</code> Adds <text> for a chapter precis to the ToC. ....	151
<code>\chapterrefname</code> Name for a chapter used by <code>\Cref</code> . ....	334

---

<code>\chapterstyle{&lt;style&gt;}</code>	Sets chapter heading layout to <i>&lt;style&gt;</i> . . . . .	130
<code>\chaptitelfont</code>	Font used by <code>\printchaptertitle</code> . . . . .	129
<code>\checkandfixthelayout[&lt;algorithm&gt;]</code>	. . . . .	82
	Command to check and implement the page layout specifications, adjusting the <code>\textheight</code> using <i>&lt;algorithm&gt;</i> (classic, fixed, lines, or nearest, the default being classic) for the calculation.	
<code>\checkarrayindex{&lt;arrayname&gt;}{&lt;index&gt;}</code>	. . . . .	375
	Checks if <i>&lt;index&gt;</i> is a valid index value for the array <i>&lt;arrayname&gt;</i> . Sets <code>\ifbounderror</code> true if there is a problemn otherwise false.	
<code>\checkifinteger{&lt;num&gt;}</code>	. . . . .	375
	If <i>&lt;num&gt;</i> is an integer and not less than zero, sets <code>\ifinteger</code> true, otherwise false.	
<code>\checkoddpag</code>	. . . . .	369
	Sets <code>\ifoddpag</code> true if called on an odd-numbered page, otherwise false.	
<code>\checkthelayout[&lt;algorithm&gt;]</code>	. . . . .	82
	Command to check the page layout specifications, adjusting the <code>\textheight</code> using <i>&lt;algorithm&gt;</i> (classic, fixed, lines, or nearest, the default being classic) for the calculation.	
<code>\cite[&lt;detail&gt;]{&lt;labstr-list&gt;}</code>	. . . . .	337
	Citation in the text to bibliographic items specified in the <i>&lt;labstr-list&gt;</i> of comma-separated bibliographic identifiers; optional information, e.g., page number, is supplied via <i>&lt;detail&gt;</i> .	
<code>\citeindexfile</code>	File name for the citation index. . . . .	349
<code>\clearforchapter</code>	Selects a chapter's opening page. . . . .	128
<code>\cleartoevenpage[&lt;text&gt;]</code>	. . . . .	291
	Clears the current page and moves to the next verso page; the optional <i>&lt;text&gt;</i> is put on the skipped page (if there is one).	
<code>\cleartooddpag[&lt;text&gt;]</code>	. . . . .	370
	Flushes any pending floats to then start typesetting on the next odd page. The optional <i>&lt;text&gt;</i> is put on the skipped page (if there is one).	
<code>\cleartorecto</code>	Simpler form of <code>\cleartooddpag</code> . . . . .	370
<code>\cleartoverso</code>	Simpler form of <code>\cleartoevenpag</code> . . . . .	370
<code>\closeinputstream{&lt;stream&gt;}</code>	. . . . .	323
	Detaches and closes the file associated with the input <i>&lt;stream&gt;</i> .	
<code>\closeoutputstream{&lt;stream&gt;}</code>	. . . . .	322
	Detaches and closes the file associated with the output <i>&lt;stream&gt;</i> .	
<code>\cmd{&lt;cmd&gt;}</code>	. . . . .	383
	where <i>&lt;cmd&gt;</i> is a macro name like <code>\cmd</code> , prints and indexes <code>\cmd</code> .	
<code>\cmdprint{&lt;cmd&gt;}</code>	where <i>&lt;cmd&gt;</i> is a macro name like <code>\cmd</code> , prints <code>\cmd</code> . . . . .	383
<code>\cmidrule[&lt;width&gt;](&lt;trim&gt;){&lt;m-n&gt;}</code>	. . . . .	260
	Draws a rule, default thickness <code>\cmidrulewidth</code> , across tabular columns <i>&lt;m&gt;</i> to <i>&lt;n&gt;</i> ; the ends may be <i>&lt;trim&gt;</i> ed by <code>\cmidrulekern</code> .	
<code>\cmidrulekern</code>	Trim amount for <code>\cmidrule</code> . . . . .	260
<code>\cmidrulewidth</code>	Default width for a <code>\cmidrule</code> tabular rule. . . . .	260
<code>\colorchapnum</code>	Color for the number in the <i>pedersen</i> chapterstyle. . . . .	148
<code>\colorchaptitle</code>	Color for the title in the <i>pedersen</i> chapterstyle. . . . .	148
<code>\columnsep</code> (length)	Width of the gutter between columns. . . . .	80
<code>\columnseprule</code> (length)	Thickness of the rule in the gutter. . . . .	80
<code>\begin{comment}</code>	Skip over the environment. . . . .	314
<code>\commentsoff{&lt;name&gt;}</code>	. . . . .	314
	Process contents of the <i>&lt;name&gt;</i> comment environment.	
<code>\commentson{&lt;name&gt;}</code>	. . . . .	314
	Skip contents of the <i>&lt;name&gt;</i> comment environment.	
<i>companion</i>	Chapterstyle like those in the <i>LaTeX companion</i> series. . . . .	132

<code>\contcaption{&lt;text&gt;}</code> .....	229
A continued caption, replacing the original title with <code>&lt;text&gt;</code> .	
<code>\contentsline{&lt;kind&gt;}{&lt;text&gt;}{&lt;page&gt;}</code> .....	190
An entry in a "List of..." file for a <code>&lt;kind&gt;</code> entry, with <code>&lt;text&gt;</code> being the title which was on <code>&lt;page&gt;</code> .	
<code>\contentsname</code> The title for the Table of Contents. ....	195
<code>\continuousmarks</code> .....	114
Specifies that the thanks/footnote marker is not zeroed after titling.	
<code>\continuousnotenums</code> .....	357
Declaration to make the numbering of endnotes continuous throughout the document.	
<code>\contsubbottom[&lt;list-entry&gt;][&lt;subtitle&gt;]{&lt;text&gt;}</code> .....	238
A continued <code>\subbottom</code> .	
<code>\contsubcaption[&lt;list-entry&gt;]{&lt;subtitle&gt;}</code> A continued <code>\subcaption</code> . ....	238
<code>\contsubtop[&lt;list-entry&gt;][&lt;subtitle&gt;]{&lt;text&gt;}</code> A continued <code>\subtop</code> . ....	238
<code>\coppagestyle{&lt;copy&gt;}original</code> .....	167
Creates a new pagestyle called <code>&lt;copy&gt;</code> using the <code>&lt;original&gt;</code> pagestyle specification.	
<code>\counterwithin{&lt;ctr&gt;}{&lt;within&gt;}</code> .....	366
Makes the counter <code>ctr</code> (created via <code>\newcounter</code> ) act as though it had been initially defined to be reset by counter <code>within</code> . It also redefines <code>\thectr</code> to include <code>\thewithin</code> .	
<code>\counterwithin*{&lt;ctr&gt;}{&lt;within&gt;}</code> .....	366
Makes the counter <code>ctr</code> (created via <code>\newcounter</code> ) act as though it had been initially defined to be reset by counter <code>within</code> , leaving the original definition of <code>\thectr</code> .	
<code>\counterwithout{&lt;ctr&gt;}{&lt;within&gt;}</code> .....	366
Makes the counter <code>ctr</code> (created via <code>\newcounter{&lt;ctr&gt;}[&lt;within&gt;]</code> ) act as though it had been initially defined via <code>\newcounter{&lt;ctr&gt;}</code> . It also redefines <code>\thectr</code> to typeset as arabic numerals.	
<code>\counterwithout*{&lt;ctr&gt;}{&lt;within&gt;}</code> .....	366
Makes the counter <code>ctr</code> (created via <code>\newcounter{&lt;ctr&gt;}[&lt;within&gt;]</code> ) act as though it had been initially defined via <code>\newcounter{&lt;ctr&gt;}</code> , leaving the original definition of <code>\thectr</code> .	
<code>\cplabel</code> .....	369
Prefix for labels used by <code>\checkoddpaper</code> odd/even page checking.	
<code>\createmark{&lt;sec&gt;}{&lt;marks&gt;}{&lt;show&gt;}{&lt;prefix&gt;}{&lt;postix&gt;}</code> .....	170
Defines the <code>\secmark</code> macro where <code>&lt;show&gt;</code> ( <code>shownumber</code> or <code>nonumber</code> ) controls whether the division number will be displayed within <code>\mainmatter</code> , <code>&lt;marks&gt;</code> is <code>left</code> , <code>both</code> or <code>right</code> , and <code>&lt;prefix&gt;</code> and <code>&lt;postfix&gt;</code> are affixed before and after the <code>&lt;sec&gt;</code> (division) number.	
<code>\createplainmark{&lt;type&gt;}{&lt;marks&gt;}{&lt;text&gt;}</code> .....	170
Defines the <code>\typemark</code> macro using <code>&lt;text&gt;</code> as the mark, where <code>&lt;marks&gt;</code> is <code>left</code> , <code>both</code> or <code>right</code> .	
<code>\Cref{&lt;labstr&gt;}</code> .....	334
Prints a named ( <code>\chapterrefname</code> ) reference to a <code>\labeled</code> chapter.	
<code>crosshead</code> A centered chapterstyle in a bold font. ....	136
<code>crownvopaper</code> Class option for crown octavo stock paper size. ....	64
<code>\cs{&lt;name&gt;}</code> prints <code>\name</code> . ....	383
<code>\begin{ctabular}[&lt;pos&gt;]{&lt;format&gt;}</code> .....	269
Like <code>tabular</code> except that it will continue over a page break.	
<code>culver</code> One line, centered, bold chapterstyle using Roman numerals. ....	138
 <code>dash</code> .....	 138
Two line, centered, regular font, chapterstyle. The number has a dash on either side.	
<code>\date{&lt;text&gt;}</code> .....	113
Used by <code>\maketitle</code> to typeset <code>&lt;text&gt;</code> as the document date.	
<code>dbillpaper</code> Class option for dollar bill stock paper size. ....	64
<code>\defaultaddspace</code> Default space for <code>\addlinespace</code> . ....	260

<i>default</i>	The default book class chapterstyle.	130
<code>\defaultlists</code>	Declaration specifying the default vertical spacing <code>list</code> -based environments.	183
<code>\defaultsecnum</code>	Declaration reversing the effect of <code>\hangsecnum</code> .	155
<code>\deleted{⟨change-id⟩}</code>	Change mark for something deleted; <code>⟨change-id⟩</code> is put in the margin.	362
<code>\DeleteShortVerb{⟨backslash-char⟩}</code>	Returns <code>⟨char⟩</code> to its normal meaning instead of being a shorthand for <code>\verb⟨char⟩</code> .	315
<i>demo2</i>	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.	139
<i>demo3</i>	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 <i>demo2</i> style.	139
<i>demyvopaper</i>	Class option for demy octavo stock paper size.	64
<code>\begin{description}</code>	A list of descriptions of <code>\item</code> s.	182
<code>\descriptionlabel{⟨label⟩}</code>	Specifies the format of the <code>⟨label⟩</code> of an <code>\item</code> in a <code>description</code> environment.	182
<code>\DisemulatePackage{⟨package⟩}</code>	Undo a previous <code>\EmulatedPackage</code> or <code>\EmulatedPackageWithOptions</code> for the <code>⟨package⟩</code> package.	380
<code>\doublerulesep</code>	Space between adjacent rules in a tabular, or an array.	259
<code>\begin{DoubleSpace}</code>	Environment form of <code>\DoubleSpacing</code>	103
<code>\DoubleSpacing</code>	Declaration doubling the baselineskip.	103
<i>dowding</i>	A centered two line chapterstyle in an italic font and the number is spelled out.	139
<code>\downbracefill</code>	Fills a tabular column with a down brace.	261
<i>draft</i>	Class option for draft document.	66
<code>\dropchapter{⟨length⟩}</code>	Lowers subsequent chapter heads by <code>⟨length⟩</code> .	290
<code>\droptitle</code>	Length controlling the position of <code>\maketitle</code> on the page (default 0pt).	111
<i>ebook</i>	Class option for electronic book stock size.	63
<i>ell</i>	A raggedleft large sanserif chapterstyle with the number in the margin. An ‘L’ shaped rule separates the number and title lines.	139
<code>\emindershape{⟨shape⟩}</code>	Font shape for emphasized text within emphasized text.	96
<code>\emph{⟨text⟩}</code>	Use a change in font to emphasise <code>⟨text⟩</code> .	95
<code>\emptythanks</code>	Discards any text from previous uses of <code>\thanks</code> .	114
<code>\EmulatedPackage{⟨package⟩}[⟨date⟩]</code>	Claim that the <code>⟨package⟩</code> package has been loaded.	379
<code>\EmulatedPackageWithOptions{⟨optionlist⟩}{⟨package⟩}[⟨date⟩]</code>	Claim that the <code>⟨package⟩</code> package has been loaded with options <code>⟨optionlist⟩</code> .	379
<code>\endMakeFramed</code>	Ends the <code>MakeFramed</code> environment.	310
<code>\enlargethispage{⟨length⟩}</code>	Increase (or decrease) the text height of the current page by <code>⟨length⟩</code> .	104
<code>\begin{enumerate}[⟨style⟩]</code>	An ordered list of <code>\item</code> s. If <code>⟨style⟩</code> is given it overrides the default scheme for indicating the item order.	183

<code>\epigraph{⟨text⟩}{⟨source⟩}</code> .....	287
Typesets the <code>⟨text⟩</code> and <code>⟨source⟩</code> of an epigraph.	
<code>\epigraphfontsize{⟨fontsize⟩}</code> Font size to be used for epigraphs. ....	289
<code>\epigraphforheader[⟨distance⟩]{⟨text⟩}</code> .....	290
Puts <code>⟨text⟩</code> into a zero-sized picture ( <code>\epigraphpicture</code> ) at the coordinate position (0, -metadistance).	
<code>\epigraphhead[⟨distance⟩]{⟨text⟩}</code> .....	290
Stores <code>⟨text⟩</code> for printing at <code>⟨distance⟩</code> below the page header.	
<code>\epigraphpicture</code> .....	290
A zero-sized picture holding the result of <code>\epigraphforheader</code> .	
<code>\epigraphposition{⟨flush⟩}</code> .....	288
Sets the horizontal placement of epigraphs.	
<code>\epigraphrule</code> .....	289
Thickness of the rule drawn between the text and source of an epigraph.	
<code>\begin{epigraphs}</code> Environment for several epigraphs. ....	287
<code>\epigraphsourceposition{⟨flush⟩}</code> .....	288
Sets the placement of the source within an epigraph.	
<code>\epigraphtextposition{⟨flush⟩}</code> .....	288
Sets the justification for epigraph text.	
<code>\epigraphwidth</code> Textwidth for epigraphs. ....	288
<code>executivepaper</code> Class option for executive-paper stock paper size. ....	64
<code>extrafontsizes</code> Class option for using scalable fonts that can exceed 25pt. ....	65
<code>\extrarowheight</code> .....	267
Length added to the normal row height in tabulars and arrays.	
<code>\extratabsurround</code> .....	268
Adds additional space for <code>\firstline</code> and <code>\lastline</code> .	
 <code>\fancybreak{⟨text⟩}</code> .....	157
An anonymous division with <code>⟨text⟩</code> centered and the following paragraph is not indented.	
<code>\fancybreak*{⟨text⟩}</code> .....	157
Like <code>\fancybreak</code> except that the following paragraph is indented.	
<code>\begin{fboxverbatim}</code> .....	317
Puts a frame around the verbatim material. Page breaks are not allowed.	
<code>\fcardinal{⟨number⟩}</code> .....	370
Typesets <code>{⟨number⟩}</code> as a cardinal number, with <code>\fnumbersep</code> between each triple of digits.	
<code>\feetabovefloat</code> Typeset footnotes above bottom floats (the default). ....	274
<code>\feetbelowfloat</code> Typeset footnotes below bottom floats. ....	274
<code>\figurerefname</code> Name for a figure used by <code>\fref</code> . ....	334
<code>final</code> Class option for final document. ....	66
<code>\firmlist</code> .....	183
In a standard list, sets the vertical spacing intermediate between the default and <code>\tightlist(s)</code> .	
<code>\firmlists</code> .....	183
Declaration for some vertical spacing in list-based environments. There may be some extra space before and after the environments.	
<code>\firmlists*</code> .....	183
The same as <code>\firmlists</code> except that there is no space before and after the environments.	
<code>\firstline</code> .....	268
An <code>\hline</code> (the first) that does not effect vertical alignment of an array or tabular.	
<code>\fixdvipslayout</code> Sets up page size information for dvi processors. ....	85
<code>\fixpdflayout</code> Sets up page size information for pdfLaTeX. ....	85

<code>\flagverse{⟨flag⟩}</code> .....	300
Used at the start of a verse line to put <code>⟨flag⟩</code> distance <code>\leftskip</code> before the start of the line.	
<code>\flegfigure</code> The name for a <code>\legend</code> in a figure. ....	232
<code>\flegfloat{⟨name⟩}</code> .....	232
Where <code>float</code> is a float type (e.g. <code>table</code> ), defines the <code>⟨name⟩</code> used by <code>\namedlegend</code> .	
<code>\flegtable</code> The name for a <code>\legend</code> in a table. ....	232
<code>\flegtocfloat{⟨title⟩}</code> .....	232
Where <code>float</code> is a float type (e.g., <code>figure</code> ), called by <code>\namedlegend</code> to add <code>⟨title⟩</code> to a 'List of...'. .	
<code>fleqn</code> Class option for fixed indentation of displayed math. ....	66
<code>\flushbottom</code> .....	67
Declaration for last line on a page to be at a constant height.	
<code>\begin{flushleft}</code> Text to be typeset flushleft and raggedright. ....	179
<code>\begin{flushright}</code> Text to be typeset flushright and raggedleft. ....	179
<code>\fnsymbol{⟨counter⟩}</code> .....	279
Typesets the representation of the footnote marker.	
<code>\fnumbersep</code> Separator between digit triples in numbers. ....	370
<code>foolscapvopaper</code> Class option for foolscap octavo stock paper size. ....	64
<code>\footfootmark</code> Typesets the footnote mark at the bottom of the page. ....	278
<code>\footfudgefiddle</code> .....	275
Integer number (default 64) to help when typesetting <code>\paragraphfootnotes</code> .	
<code>\footmarksep</code> Offset from the footnote mark box for lines after the first. ....	278
<code>\footmarkstyle{⟨style⟩}</code> .....	278
Style of the footnote mark at the bottom of the page.	
<code>\footmarkwidth</code> Width of footnote mark box. ....	278
<code>\footnote[⟨num⟩]{⟨text⟩}</code> Put <code>⟨text⟩</code> as a footnote. ....	273
<code>\footnoteA{⟨text⟩}</code> A series A footnote. ....	275
<code>\footnoteB{⟨text⟩}</code> A series B footnote. ....	275
<code>\footnoteC{⟨text⟩}</code> A series C footnote. ....	275
<code>\footnotemark[⟨num⟩]</code> Typesets a footnote mark. ....	273
<code>\footnoterule</code> Rule separating footnotes from the main text. ....	280
<code>\footnotetext[⟨num⟩]{⟨text⟩}</code> .....	273
Typesets <code>⟨text⟩</code> as a footnote at the bottom of the page but does not put a mark in the main text.	
<code>\footparindent</code> Paragraph indent for multiparagraph footnote text. ....	278
<code>\footreflabstr</code> Reference a labelled footnote. ....	273
<code>\footruleheight</code> .....	168
Macro specifying the height of a normal rule above a footer.	
<code>\footruleskip</code> .....	168
Macro specifying a distance sufficient to ensure that a rule above a footer will lie in the space between the footer and the typeblock.	
<code>\footskip (length)</code> .....	80
Distance between the bottom of the typeblock and the bottom of a footer.	
<code>\foottextfont</code> Font for footnote text. ....	278
<code>\foottopagenote</code> .....	360
Declaration which turns <code>\footnotes</code> into <code>\pagenotes</code> .	
<code>\fordinal{⟨number⟩}</code> .....	371
Typesets <code>{⟨number⟩}</code> as an ordinal number, with <code>\fnumbersep</code> between each triple of digits.	
<code>\FrameCommand</code> Draws a 'frame'. ....	310
<code>\begin{framed}</code> .....	310
Put a ruled box around the contents of the environment; the box can include pagebreaks.	

<code>\FrameHeightAdjust</code> .....	310
Height of the top of a frame in a <code>framed</code> environment above the baseline at the top of a page.	
<code>\FrameRestore</code> Restores settings after a ‘frame’. .....	310
<code>\FrameRule</code> Thickness of the rules around an <code>framed</code> environment. ....	310
<code>\FrameSep</code> .....	310
Separation between the surrounding box and text in a <code>framed</code> or <code>shaded</code> environment.	
<code>\fref{&lt;labstr&gt;}</code> .....	334
Prints a named ( <code>\figurerefname</code> ) reference to a <code>\labeled</code> figure.	
<code>\frontmatter</code> .....	121
Sets folios to be printed in lowercase roman and prohibits sectional number.	
<code>\frontmatter*</code> Same as <code>\frontmatter</code> except that folios are unaltered. ....	121
<code>\fussy</code> .....	104
Declaration for TeX to minimise interword space variations in justified text lines.	
 <code>ger</code> .....	139
A raggedright, large, bold, two line chapterstyle with rules above and below.	
<code>\getarrayelement{&lt;arrayname&gt;}{&lt;index&gt;}{&lt;result&gt;}</code> .....	375
Sets the parameterless macro <code>&lt;result&gt;</code> to the contents of the <code>{&lt;index&gt;}</code> location in array <code>&lt;arrayname&gt;</code> .	
<code>\glossary[&lt;file&gt;](&lt;key&gt;){&lt;term&gt;}{&lt;description&gt;}</code> .....	350
Adds <code>&lt;term&gt;</code> and its description, <code>&lt;desc&gt;</code> , to a glossary file — <code>\jobname.glo</code> by default or to <code>\file.glo</code> . The optional argument <code>&lt;key&gt;</code> can be used to provide a different sort key for <code>&lt;term&gt;</code> .	
<code>\glossarycolsep</code> Columns separation in a two column glossary. ....	353
<code>\glossaryintoc</code> Declaration to add glossary title to the ToC. ....	353
<code>\glossarymark</code> Redefine to specify marks for headers. ....	353
<code>\glossaryname</code> Name for a glossary. ....	353
<code>\glossaryrule</code> Width of inter-column rule in a two column glossary. ....	353
<code>\glossitem{&lt;term&gt;}{&lt;desc&gt;}{&lt;ref&gt;}{&lt;num&gt;}</code> .....	351
Glossary entry used in a <code>theglossary</code> environment	
 <code>\hangcaption</code> .....	226
Multiline captions will be typeset as a hanging paragraph.	
<code>\hangfrom{&lt;stuff&gt;}</code> Hangs a paragraph from <code>&lt;stuff&gt;</code> . ....	178
<code>\@hangfrom{&lt;code&gt;}</code> .....	154
Kernel macro aiding the setting hanging paragraphs.	
<code>hangnum</code> .....	130
Simple chapterstyle looking like a section head but with the number in the margin.	
<code>\hangpara{&lt;indent&gt;}{&lt;num&gt;}</code> .....	178
Apply <code>&lt;indent&gt;</code> for <code>&lt;num&gt;</code> lines to the immediately following paragraph.	
<code>\begin{hangparas}{&lt;indent&gt;}{&lt;num&gt;}</code> .....	178
Environment form of <code>\hangpara</code> , applying it to every paragraph in the environment.	
<code>\hangsecnum</code> .....	155
Declaration making sectioning numbers hang in the margin.	
<code>\hangsubcaption</code> The subcaption version of <code>\hangcaption</code> . ....	239
<code>\headdrop (length)</code> .....	80
Distance between the top of the trimmed page and the top of a header.	
<code>\headheight (length)</code> Height available for a header. ....	80
<code>\headnameref</code> Use header title for sectional title references. ....	335
<code>\headsep (length)</code> .....	80
Distance between the bottom of a header and the top of the typeblock.	

<code>\headstyles{&lt;name&gt;}</code>	159
Use the <code>&lt;name&gt;</code> set of headstyles for sectional division heads.	
<code>\headwith</code>	168
A (scratch) length normally used in the definition of headers and footers.	
<code>\heavyrulewidth</code>	259
Default width for a heavy tabular rule.	
<code>\hideindexmarks</code>	343
The <code>&lt;stuff&gt;</code> argument to <code>\index</code> and <code>\specialindex</code> will not be printed in the margin (the default).	
<code>\hmpunct</code>	365
Punctuation between hours and minutes in <code>\printtime</code> (default :)	
<code>\hrulefill</code>	261
Fills a tabular column with a rule.	
<code>\idtextinnotes{&lt;id&gt;}</code>	358
Prints an endnote's <code>&lt;id&gt;</code> text	
<code>\ifdraftdoc</code>	361
true if the draft class option has been used.	
<code>\ifetex</code>	376
true if etex is the underlying engine, otherwise false.	
<code>\ifoddpage</code>	369
Result of <code>\checkoddpaper</code> .	
<code>\ifonlyfloats{&lt;yes&gt;}{&lt;no&gt;}</code>	174
Processes <code>&lt;yes&gt;</code> on a page containing only floats, otherwise process <code>&lt;no&gt;</code> .	
<code>\ifreversesidepar</code>	281
If TRUE, reverses the margin for <code>\sidepars</code> .	
<code>\ifsamename</code>	368
Result from <code>\nametest</code> .	
<code>\ifsidebaroneside</code>	283
If true then sidebars are set in the right margin.	
<code>\ifsidecapleft</code>	240
true if sidecaptions will be set in the left margin, otherwise they will be set in the right margin.	
<code>\ifsideparswitch</code>	281
Depending on the class one/two side option specifies the margin for <code>\sidepars</code> .	
<code>\IfStreamOpen{&lt;stream&gt;}{&lt;yes&gt;}{&lt;no&gt;}</code>	322
If <code>&lt;stream&gt;</code> is open then the <code>&lt;yes&gt;</code> argument is processed otherwise the <code>&lt;no&gt;</code> argument is processed.	
<code>\ifxetex</code>	376
true if XeTeX is being used to process the document.	
<code>\ignorenoidxfile</code>	341
Do not report attempts to use an <code>idx</code> file that has not been declared by <code>\makeindex</code> .	
<code>\iiiirdstring</code>	371
Ordinal characters for 'rd', e.g., as in 3rd.	
<code>\iindstring</code>	371
Ordinal characters for 'nd', e.g., as in 2nd.	
<code>imperialvopaper</code>	64
Class option for imperial octavo stock paper size.	
<code>\indentcaption{&lt;length&gt;}</code>	226
Multiline captions will be typeset as a hanging paragraph hung by <code>&lt;length&gt;</code> .	
<code>\indentpattern{&lt;digits&gt;}</code>	299
Stanza lines in <code>patverse</code> environment are indented according to the <code>&lt;digits&gt;</code> pattern.	
<code>\index[&lt;file&gt;]{&lt;stuff&gt;}</code>	342
Add <code>&lt;stuff&gt;</code> and the current page number to the raw index data. By default this is written to file <code>\jobname.idx</code> . If the optional argument is given it will be written to file <code>&lt;file&gt;.idx</code> .	
<code>\indexcolsep (length)</code>	341
Width of the gutter in two column indexes.	
<code>\indexintoc</code>	341
Add the index title to the ToC (the default).	
<code>\indexmark</code>	341
Can be used in pagestyles for page headers in an index.	
<code>\indexmarkstyle</code>	343
Font style for printing index marks in the margin.	
<code>\indexname</code>	341
Name used for the the index title.	
<code>\indexrule (length)</code>	341
Width of the inter-column rule in two column indexes.	
<code>\insertchapterspace</code>	130
Called by <code>\chapter</code> to insert space into LoF and LoT.	

<code>\isopage[⟨spine⟩]</code> .....	89
Generates a page layout especially suited to ISO proportioned paper.	
<code>\iststring</code> Ordinal characters for ‘st’, e.g., as in 1st. ....	371
<code>\item[⟨label⟩]</code> .....	182
Introduces a new element in a list. The effect of <code>⟨label⟩</code> depends on the particular list form.	
<code>\item</code> Introduces a main index entry. ....	343
<code>\begin{itemize}[⟨marker⟩]</code> .....	182
An unordered list of <code>\items</code> . If given, the <code>⟨marker⟩</code> overrides the default marker for the elements.	
<code>\itshape</code> Declaration for using an italic font. ....	94
<code>\jobname</code> The name of the document’s main source file. ....	321
<code>\keepthetitle</code> .....	114
Makes most aspects of <code>\maketitle</code> unavailable but keeps <code>\thetitle</code> , <code>\theauthor</code> and <code>\thedate</code> .	
<code>\killtitle</code> Makes all aspects of <code>\maketitle</code> unavailable. ....	114
<code>komalike</code> A section-like chapterstyle in a sans serif font. ....	139
<code>\label{⟨labstr⟩}</code> .....	333
Associates the current (section, caption, ...) number, and page number, to <code>⟨labstr⟩</code> .	
<code>\label(⟨bookmark⟩){⟨labstr⟩}</code> .....	238
Associates <code>⟨labstr⟩</code> with the current (section, caption, etc.) number and page number. If used inside a subfloat and with the <code>hyperref</code> package the optional <code>⟨bookmark⟩</code> (note the parentheses not square brackets) is available to specify a <code>hyperref</code> bookmark.	
<code>landscape</code> .....	63
Class option to interchange height and width of stock paper size.	
<code>largecrownvopaper</code> Class option for large crown octavo stock paper size. ....	64
<code>largepostvopaper</code> Class option for large post octavo stock paper size. ....	64
<code>\lasthline</code> .....	268
An <code>\hline</code> (the last) that does not effect vertical alignment of an array or tabular.	
<code>lastpage</code> .....	364
Counter holding the number of the last page processed during the <i>previous</i> LaTeX run.	
<code>lastsheet</code> .....	364
Counter holding the number of sheets processed during the <i>previous</i> LaTeX run.	
<code>\lazypagecheck</code> .....	369
<code>\checkoddpaper</code> will use a possibly inaccurate but fast method for checking for an odd page number.	
<code>\lcmminusname</code> Lowercase ‘minus’ name, default ‘minus’. ....	373
<code>\leavespergathering{⟨num⟩}</code> .....	364
Ensure that the correct number of pages are output to make up gatherings of <code>⟨num⟩</code> leaves each.	
<code>ledgerpaper</code> Class option for ledger stock paper size. ....	64
<code>\begin{leftbar}</code> .....	311
Draws a thick vertical line in the left margin alongside the contents of the environment.	
<code>\leftmark</code> .....	166
Contains the value of the <code>⟨left⟩</code> argument of the last <code>\markboth</code> .	
<code>legalpaper</code> Class option for legal-paper stock paper size. ....	64
<code>\legend{⟨text⟩}</code> A legend (an anonymous caption). ....	230
<code>leqno</code> Class option for numbering equations at the left. ....	66
<code>letterpaper</code> Class option for letterpaper stock paper size. ....	64
<code>\lightrulewidth</code> Default width for a light tabular rule. ....	259

---

<code>\linenumberfont{&lt;fontspec&gt;}</code>	Specify the font for line numbers. ....	318
<code>\linenumberfrequency{&lt;nth&gt;}</code>	Number every <nth> line in a <code>boxedverbatim</code> or a <i>verse</i> . ....	318
<code>\begin{list}{&lt;default-label&gt;}{&lt;code&gt;}</code>	The general list environment. <default-label> is code that is used for an <code>\item</code> with no <label> and <code> is used to specify the list layout parameters. ....	183
<code>\listfigurename</code>	The title for the List of Figures. ....	195
<code>\listoffigures</code>	Typeset the LoF, adding its title to the ToC. ....	194
<code>\listoffigures*</code>	Typeset the LoF. ....	194
<code>\listoftables</code>	Typeset the LoT, adding its title to the ToC. ....	194
<code>\listoftables*</code>	Typeset the LoT. ....	194
<code>\listtablename</code>	The title for the List of Tables. ....	195
<code>\loosesubcaptions</code>	Specifies extra vertical space around subcaptions. ....	239
<code>\lxvchars (length)</code>	Approximate length of 65 characters. ....	75
<code>lyhne</code>	A raggedleft bold sanserif chapter title set between two rules, with the name and number above. It requires the <code>graphicx</code> package. ....	139
<code>madsen</code>	A raggedleft large bold sanserif chapterstyle with the number in the margin and a rule between the number and title lines. It requires the <code>graphicx</code> package. ....	142
<code>\mainmatter</code>	Sets folio numbers to arabic, starting with 1. Floats, etc., will be numbered per chapter and sectional divisions will be numbered. ....	121
<code>\mainmatter*</code>	Same as <code>\mainmatter</code> except that folios are unaltered. ....	121
<code>\makeevenfoot{&lt;style&gt;}{&lt;left&gt;}{&lt;center&gt;}{&lt;right&gt;}</code>	Defines the <left>, <center> and <right> parts of the even (verso) page footer of the <style> paget-style. ....	167
<code>\makeevenhead{&lt;style&gt;}{&lt;left&gt;}{&lt;center&gt;}{&lt;right&gt;}</code>	Defines the <left>, <center> and <right> parts of the even (verso) page header of the <style> paget-style. ....	167
<code>\makefootrule{&lt;style&gt;}{&lt;width&gt;}{&lt;thickness&gt;}{&lt;skip&gt;}</code>	Specifies the <width> and <thickness> of the rule drawn <skip> (see <code>\footskip</code> ) above the footers of the <style> pagestyle. ....	168
<code>\MakeFramed{&lt;settings&gt;}</code>	The <code>MakeFramed</code> environment is the workhorse for the <code>framed</code> , <code>shaded</code> , etc., environments. The <settings> argument controls the final appearance and should include a <code>\FrameRestore</code> to reset things back to normal. ....	310
<code>\makeglossary[&lt;file&gt;]</code>	Opens file <code>\jobname.glo</code> , or <code>\file.glo</code> , for glossary entries ....	350
<code>\makeheadposition{&lt;style&gt;}{&lt;ehpos&gt;}{&lt;ohpos&gt;}{&lt;efpos&gt;}{&lt;ofpos&gt;}</code>	Specifies the horizontal positioning of the even and odd headers and footers respectively for the <style> pagestyle. ....	168
<code>\makeheadrule{&lt;style&gt;}{&lt;width&gt;}{&lt;thickness&gt;}</code>	Specifies the <width> and <thickness> of the rule drawn below the headers of the <style> pagestyle. ....	168
<code>\makeheadstyles{&lt;name&gt;}{&lt;code&gt;}</code>	Create a new set of sectional division headstyles called <name> defined by <code>. ....	159
<code>\makeindex[&lt;file&gt;]</code>	Preamble command to collect raw index information. By default this is written to file <code>\jobname.idx</code> . If the optional argument is used it may be written to file <file>.idx. ....	340

<code>\makeoddfoot{&lt;style&gt;}{&lt;left&gt;}{&lt;center&gt;}{&lt;right&gt;}</code> .....	167
Defines the <code>&lt;left&gt;</code> , <code>&lt;center&gt;</code> and <code>&lt;right&gt;</code> parts of the odd (recto) page footer of the <code>&lt;style&gt;</code> pagestyle.	
<code>\makeoddhead{&lt;style&gt;}{&lt;left&gt;}{&lt;center&gt;}{&lt;right&gt;}</code> .....	167
Defines the <code>&lt;left&gt;</code> , <code>&lt;center&gt;</code> and <code>&lt;right&gt;</code> parts of the odd (recto) page header of the <code>&lt;style&gt;</code> pagestyle.	
<code>\makepagenote</code> Preamble command for enabling page/end notes. ....	356
<code>\makepagestyle{&lt;style&gt;}</code> Used to define a pagestyle <code>&lt;style&gt;</code> . ....	167
<code>\makepsmarks{&lt;style&gt;}{&lt;code&gt;}</code> .....	169
Hook into the <code>&lt;style&gt;</code> pagestyle, usually used for the <code>&lt;code&gt;</code> setting any marks.	
<code>\makerunningheadwidth{&lt;style&gt;}{&lt;length&gt;}</code> .....	168
Sets the width of the <code>&lt;style&gt;</code> pagestyle headers and footers to <code>&lt;length&gt;</code> .	
<code>\MakeShortVerb{&lt;backslash-char&gt;}</code> .....	315
Makes <code>&lt;char&gt;</code> a shorthand for <code>\verb&lt;char&gt;</code> .	
<code>\makethanksmark</code> Typesets the thanks marker and text at the foot. ....	116
<code>\makethanksmarkhook</code> Code hook into <code>\makethanksmark</code> . ....	116
<code>\maketitlehooka</code> Hook into <code>\maketitle</code> applied before the <code>\title</code> . ....	112
<code>\maketitlehookb</code> .....	112
Hook into <code>\maketitle</code> applied between the <code>\title</code> and <code>\author</code> .	
<code>\maketitlehookc</code> .....	112
Hook into <code>\maketitle</code> applied between the <code>\author</code> and <code>\date</code> .	
<code>\maketitlehookd</code> Hook into <code>\maketitle</code> applied after the <code>\date</code> . ....	112
<code>\marg{&lt;arg&gt;}</code> prints <code>{&lt;arg&gt;}</code> . ....	383
<code>\marginpar[&lt;left-text&gt;]{&lt;text&gt;}</code> .....	280
Puts <code>&lt;text&gt;</code> into the margin; if given, <code>&lt;left-text&gt;</code> will be used instead of <code>&lt;text&gt;</code> for the left margin.	
<code>\marginparpush (length)</code> .....	81
Minimum vertical distance between marginal notes.	
<code>\marginparsep (length)</code> .....	81
Horizontal distance between the edge of the typeblock and a marginal note.	
<code>\marginparwidth (length)</code> Maximum width of marginal note. ....	81
<code>\markboth{&lt;left&gt;}{&lt;right&gt;}</code> .....	165
Sets values of two markers to <code>&lt;left&gt;</code> and <code>&lt;right&gt;</code> respectively (see <code>\leftmark</code> and <code>\rightmark</code> ).	
<code>\markright{&lt;right&gt;}</code> .....	165
Sets value of one marker to <code>&lt;right&gt;</code> (see <code>\rightmark</code> ).	
<code>\maxsecnumdepth{&lt;secname&gt;}</code> .....	124
Sets division numbering level in the <code>\mainmatter</code> to <code>&lt;secname&gt;</code> .	
<code>\maxtocdepth{&lt;secname&gt;}</code> .....	194
Sets the maximum value of the <code>tocdepth</code> counter.	
<code>mcrownvopaper</code> Class option for metric crown octavo stock paper size. ....	63
<code>mdemyvopaper</code> Class option for metric demy octavo stock paper size. ....	63
<code>\mdseries</code> Declaration for using a medium font. ....	94
<code>\medievalpage[&lt;spine&gt;]</code> .....	89
Generates a page layout according to the principles of early printers.	
<code>mediumvopaper</code> Class option for medium octavo stock paper size. ....	64
<code>\medspace</code> A medium space (4/18 em). ....	377
<code>\memappchapinfo{&lt;thechapter&gt;}{&lt;fortoc&gt;}{&lt;forhead&gt;}{&lt;title&gt;}</code> .....	382
Code hook into an appendix <code>\chapter</code>	
<code>\memappchapstarinfo{&lt;fortoc&gt;}{&lt;title&gt;}</code> .....	382
Code hook into an appendix <code>\chapter*</code>	
<code>\memapppageinfo{&lt;title&gt;}</code> Code hook into <code>\appendixpage</code> . ....	383
<code>\memapppagestarinfo{&lt;title&gt;}</code> Code hook into <code>\appendixpage*</code> . ....	383

---

<code>\membcaptioninfo{&lt;type&gt;}{&lt;thetype&gt;}{&lt;fortoc1&gt;}{&lt;title1&gt;}{&lt;name2&gt;}{&lt;title2&gt;}</code>	383
Code hook into <code>\membcaption</code> .	
<code>\membionenumcaptioninfo{&lt;type&gt;}{&lt;thetype&gt;}{&lt;fortoc1&gt;}{&lt;title1&gt;}{&lt;name2&gt;}{&lt;fortoc2&gt;}{&lt;title2&gt;}</code>	383
Code hook into <code>\membionenumcaption</code> .	
<code>\membitwonumcaptioninfo{&lt;type&gt;}{&lt;thetype&gt;}{&lt;fortoc1&gt;}{&lt;title1&gt;}{&lt;name2&gt;}{&lt;fortoc2&gt;}{&lt;title2&gt;}</code>	383
Code hook into <code>\membitwonumcaption</code> .	
<code>\membookinfo{&lt;thebook&gt;}{&lt;fortoc&gt;}{&lt;title&gt;}</code>	Code hook into <code>\book</code> 382
<code>\membookstarinfo{&lt;title&gt;}</code>	Code hook into <code>\book*</code> 382
<code>\memcaptioninfo{&lt;type&gt;}{&lt;thetype&gt;}{&lt;fortoc&gt;}{&lt;title&gt;}</code>	383
Code hook into <code>\caption</code>	
<code>\memchapinfo{&lt;thechapter&gt;}{&lt;fortoc&gt;}{&lt;forhead&gt;}{&lt;title&gt;}</code>	382
Code hook into <code>\chapter</code>	
<code>\memchapstarinfo{&lt;fortoc&gt;}{&lt;title&gt;}</code>	Code hook into <code>\chapter*</code> 382
<code>\memdskip</code>	Adjusts the display skips according to <code>\memdskipstretch</code> . 103
<code>\memdskipstretch</code>	The current factor for increasing display skips. 103
<code>\memfontenc</code>	65
Font encoding for the <code>extrafont sizes</code> class option (default T1)	
<code>\memfontfamily</code>	65
Font family for the <code>extrafont sizes</code> class option (default lmr)	
<code>\memfontpack</code>	65
Font package for the <code>extrafont sizes</code> class option (default lmodern)	
<code>\memglodesc{&lt;desc&gt;}</code>	Wrapper round a glossary description. 352
<code>\memglonum{&lt;num&gt;}</code>	Wrapper round glossary numbers. 352
<code>\memgloref{&lt;ref&gt;}</code>	Wrapper round a glossary ref. 352
<code>\memgloterm{&lt;term&gt;}</code>	Wrapper round a glossary term. 352
<code>\memgobble{&lt;text&gt;}</code>	Gobbles its argument. Do not redefine it. 379
<code>\memjustarg{&lt;text&gt;}</code>	Definition is just <code>&lt;text&gt;</code> . Do not redefine it. 379
<code>\memleadpageinfo{&lt;pstyle&gt;}{&lt;cmdname&gt;}{&lt;title&gt;}</code>	383
Code hook into <code>\newleadpage</code> and <code>\renewleadpage</code> .	
<code>\memleadpageinfo{&lt;pstyle&gt;}{&lt;cmdname&gt;}{&lt;title&gt;}</code>	383
Code hook into <code>\newleadpage*</code> and <code>\renewleadpage*</code> .	
<code>\memlegendinfo{&lt;title&gt;}</code>	Code hook into <code>\legend</code> 383
<code>\memnamedlegendinfo{&lt;fortoc&gt;}{&lt;title&gt;}</code>	383
Code hook into <code>\namedlegend</code>	
<code>\mempartinfo{&lt;thepart&gt;}{&lt;fortoc&gt;}{&lt;title&gt;}</code>	Code hook into <code>\part</code> 382
<code>\mempartstarinfo{&lt;title&gt;}</code>	Code hook into <code>\part*</code> 382
<code>\mempoeminfo{&lt;title&gt;}</code>	Code hook into <code>\poemtitle</code> 383
<code>\mempoemstarinfo{&lt;title&gt;}</code>	Code hook into <code>\poemtitle*</code> 383
<code>\memPoemTitleinfo{&lt;thepoem&gt;}{&lt;fortoc&gt;}{&lt;forhead&gt;}{&lt;title&gt;}</code>	383
Code hook into <code>\PoemTitle</code>	
<code>\memPoemTitlestarinfo{&lt;fortoc&gt;}{&lt;title&gt;}</code>	383
Code hook into <code>\PoemTitle*</code>	
<code>\memsecinfo{&lt;name&gt;}{&lt;thename&gt;}{&lt;fortoc&gt;}{&lt;forhead&gt;}{&lt;title&gt;}</code>	382
Code hook into the <code>\name</code> section command	
<code>\memsecstarinfo{&lt;name&gt;}{&lt;title&gt;}</code>	382
Code hook into the <code>\name*</code> section command	
<code>\memUHead{&lt;text&gt;}</code>	170
May uppercase <code>&lt;text&gt;</code> , depending on <code>\uppercaseheads</code> and <code>\nouppercaseheads</code> .	
<code>\meta{&lt;arg&gt;}</code>	prints <code>&lt;arg&gt;</code> . 383

<code>\midbicaption{⟨text⟩}</code> .....	236
In bilingual captions, <code>⟨text⟩</code> is inserted after the first <code>\caption</code> and immediately before the second <code>\caption</code> .	
<code>\midbookskip</code> Spacing between a <code>\book's</code> number line and the title. ....	126
<code>\midchapskip</code> Space between chapter name and number and the title. ....	129
<code>\midpartskip</code> Spacing between a <code>\part's</code> number line and the title. ....	126
<code>\midPoemTitleskip</code> .....	302
Vertical space between the number and text of a poem title.	
<code>\midrule[⟨width⟩]</code> .....	259
Draws a rule across a tabular, default width <code>\lightrulewidth</code> .	
<code>\midsloppy</code> .....	104
Declaration for TeX to allow moderate interword space variations in justified text lines.	
<code>\begin{midsloppy}</code> .....	104
Typeset contents of the enclosed paragraph(s) using <code>\midsloppy</code> .	
<code>\minusname</code> Typeset for 'minus' before negative named numbers. ....	373
<code>\mlargecrownvopaper</code> .....	63
Class option for metric large crown octavo stock paper size.	
<code>\movetoevenpage[⟨text⟩]</code> .....	370
Stops the current page to start typesetting on the next even page. The optional <code>⟨text⟩</code> is put on the skipped page (if there is one).	
<code>\movetooddpage[⟨text⟩]</code> .....	370
Stops the current page to start typesetting on the next odd page. The optional <code>⟨text⟩</code> is put on the skipped page (if there is one).	
<code>ms</code> Class option for 'typewritten manuscript'. ....	66
<code>msmallroyalvopaper</code> .....	63
Class option for metric small royal octavo stock paper size.	
<code>\multfootsep</code> Separator between adjacent footnote marks. ....	274
<code>\namedlegend[⟨short⟩]{⟨long⟩}</code> .....	231
Like <code>\caption</code> but no number and no 'List of...' entry.	
<code>\namenumberand</code> The conjunction in named numbers, default 'and '. ....	372
<code>\namenumbercomma</code> The 'comma' in named numbers, default ', '. ....	372
<code>\namerefoff</code> Makes <code>\titleref</code> inoperative. ....	335
<code>\namerefon</code> Makes <code>\titleref</code> operative. ....	335
<code>\namesubappendices</code> .....	124
Precede sub-appendix numbers with the name <code>\appendixname</code> .	
<code>\nametest{⟨str1⟩}{⟨str2⟩}</code> .....	368
Sets <code>\ifsamename</code> true if <code>⟨str1⟩</code> is the same as <code>⟨str2⟩</code> , otherwise false.	
<code>\Needspace{⟨length⟩}</code> .....	369
Starts a new page, leaving the current page short, unless there is actually at least <code>⟨length⟩</code> vertical space left on the current page.	
<code>\needspace{⟨length⟩}</code> .....	369
Starts a new page, leaving the current page short, unless there is estimated <code>⟨length⟩</code> vertical space left on the current page.	
<code>\Needspace*{⟨length⟩}</code> .....	369
Starts a new page, leaving the current page short unless <code>\flushbottom</code> is in effect, unless there is actually at least <code>⟨length⟩</code> vertical space left on the current page.	
<code>\newarray{⟨arrayname⟩}{⟨low⟩}{⟨high⟩}</code> .....	374
Defines a new indexed array datastructure called <code>⟨arrayname⟩</code> with the (integer) index ranging from <code>⟨low⟩</code> to <code>⟨high⟩</code> .	

<code>\newblock</code> .....	339
Used in a bibliography to indicate a convenient place in an entry to have a pagebreak.	
<code>\newcolumntype{&lt;char&gt;}[&lt;nargs&gt;]{&lt;spec&gt;}</code> .....	254
Creates a new column type <i>&lt;char&gt;</i> according to <i>&lt;spec&gt;</i> (which has <i>&lt;nargs&gt;</i> number of arguments).	
<code>\newcomment{&lt;name&gt;}</code> .....	314
Define a new comment environment called <i>&lt;name&gt;</i> .	
<code>\newcounter{&lt;ctr&gt;}[&lt;within&gt;]</code> .....	365
Creates a new counter <i>ctr</i> , optionally reset when counter <i>within</i> changes.	
<code>\newfixedcaption[&lt;capcommand&gt;]{&lt;command&gt;}{&lt;float&gt;}</code> .....	233
Defines a captioning command <code>\command</code> that may be used outside the <i>&lt;float&gt;</i> float as though it was inside it. The <code>\capcommand</code> must have been previously defined as a captioning command for <i>&lt;float&gt;</i> .	
<code>\newfloat[&lt;within&gt;]{&lt;fenv&gt;}{&lt;ext&gt;}{&lt;capname&gt;}</code> .....	213
Creates new float environments, <i>fenv</i> and <i>fenv*</i> , using counter <i>fenv</i> , which may be restarted by the <i>&lt;within&gt;</i> counter, putting captioning information into the file with extension <i>&lt;ext&gt;</i> , and using <i>&lt;capname&gt;</i> as the name for a caption.	
<code>\newfootnoteseries{&lt;series&gt;}</code> Create a new footnote <i>&lt;series&gt;</i> . .....	276
<code>\newinputstream{&lt;stream&gt;}</code> Creates a new input stream called <i>&lt;stream&gt;</i> . .....	321
<code>\newlistentry[&lt;within&gt;]{&lt;cntr&gt;}{&lt;ext&gt;}{&lt;level-1&gt;}</code> .....	208
Creates the commands for typesetting an entry in a 'List of...'. <i>&lt;cntr&gt;</i> is the new counter for the entry, which may be reset by the <i>&lt;within&gt;</i> counter. <i>&lt;ext&gt;</i> is the file extension and <i>&lt;level-1&gt;</i> is one less than the entry's level.	
<code>\newlistof{&lt;listofcom&gt;}{&lt;ext&gt;}{&lt;listofname&gt;}</code> .....	207
Creates two new List of... commands, <code>\listofcom</code> and <code>\listofcom*</code> , which use a file with extension <i>&lt;ext&gt;</i> and <i>&lt;listofname&gt;</i> for the title.	
<code>\newloglike{&lt;cmd&gt;}{&lt;string&gt;}</code> .....	366
Creates a new log-like function command <i>&lt;cmd&gt;</i> typesetting <i>&lt;string&gt;</i> .	
<code>\newloglike{&lt;cmd&gt;}{&lt;string&gt;}</code> .....	366
Creates a new log-like function command <i>&lt;cmd&gt;</i> typesetting <i>&lt;string&gt;</i> , which can take limits.	
<code>\newoutputstream{&lt;stream&gt;}</code> .....	321
Creates a new output stream called <i>&lt;stream&gt;</i> .	
<code>\newsubfloat{&lt;float&gt;}</code> Creates subcaptions for use in the <i>&lt;float&gt;</i> float. ....	220
<code>\nobibintoc</code> Title of the bibliography is not added to the ToC. ....	337
<code>\nobvbox</code> boxedverbatim environments will not be framed in any way. ....	317
<code>\nochangemarks</code> Do not print change marks. ....	361
<code>\nocite{&lt;labstr&gt;}</code> .....	339
Add entry <i>&lt;labstr&gt;</i> to the bibliography, but with no in-text citation.	
<code>\noDisplayskipStretch</code> No increased display skips. ....	103
<code>\noglossaryintoc</code> .....	353
Declaration to prohibit adding glossary title to the ToC.	
<code>\noindexintoc</code> Do not add the index title to the ToC. ....	341
<code>\nonzeroparskip</code> .....	101
Sets the inter-paragraph spacing to a 'perhaps not too unreasonable' non-zero value.	
<code>\noprelistbreak</code> .....	365
Putting this immediately before an <i>itemize</i> (or <i>enumerate</i> , or ...) environment should prevent a pagebreak.	
<code>\normalbottomsection</code> .....	123
Cancels any previous <code>\raggedbottomsection</code> .	
<code>\normalcaption</code> .....	226
Multiline captions will be typeset as a block paragraph.	
<code>\normalcaptionwidth</code> Captions will be set to the full width. ....	227

<code>\normalfont</code>	93
Declaration setting the font to the normal body font (upright, Roman, and medium weight).	
<code>\normalmarginpar</code>	280
Sets the normal margins used by <code>\marginpar</code> (the default).	
<code>\normalrulethickness</code>	168
The normal thickness of a visible rule (default 0.4pt).	
<code>\normalsubcaption</code>	239
The subcaption version of <code>\normalcaption</code> .	
<code>\notedivision</code>	358
Heading printed by the <code>\printnotes</code> and <code>\printnotes*</code> macros.	
<code>\noteentry{&lt;notenum&gt;}{&lt;id&gt;}{&lt;text&gt;}{&lt;pagenum&gt;}</code>	358
Typesets a pagenote with number <code>&lt;notenum&gt;</code> , identifier <code>&lt;id&gt;</code> , contents <code>&lt;text&gt;</code> , created on page <code>&lt;pagenum&gt;</code> .	
<code>\noteidinnotes{&lt;notenum&gt;}{&lt;id&gt;}</code>	358
Prints an endnote's number or id in the endnote listing.	
<code>\noteinnotes{&lt;text&gt;}</code>	359
Prints the text of an endnote.	
<code>\notenuminnotes{&lt;num&gt;}</code>	358
Typesets the number <code>&lt;num&gt;</code> of a page note in the note listing.	
<code>\notenumintext{&lt;num&gt;}</code>	358
Typesets the number <code>&lt;num&gt;</code> of a page note in the main text.	
<code>\notepageref</code>	357
Declaration that page numbers are available to notes in the endnote listing.	
<code>\notesname</code>	358
Name for endnotes, default Notes.	
<code>\nouppercaseheads</code>	170
Defines <code>\memUChhead</code> as <code>\relax</code> (i.e., do nothing).	
<code>\nouppercaseheads</code>	165
Do not uppercase the titles in the headings.	
<code>ntglke</code>	142
A smaller version of the standard <code>chapterstyle</code> .	
<code>\nthstring</code>	371
Ordinal characters for 'th', e.g., as in 6th.	
<code>\NumberPoemTitle</code>	301
Declaration for <code>\PoemTitle</code> to be numbered.	
<code>\NumtoName{&lt;number&gt;}</code>	371
Typesets <code>&lt;number&gt;</code> as a cardinal using lowercase words, but uppercase for the initial letter of each word.	
<code>\numtoName{&lt;number&gt;}</code>	371
Typesets <code>&lt;number&gt;</code> as a cardinal using lowercase words, but uppercase for the initial letter of the first word.	
<code>\numtoname{&lt;number&gt;}</code>	371
Typesets <code>&lt;number&gt;</code> as a cardinal using lowercase words.	
<code>\oarg{&lt;arg&gt;}</code>	383
prints [ <code>&lt;arg&gt;</code> ].	
<code>oldfontcommands</code>	67
Class option for permitting obsolete, deprecated font commands.	
<code>oldpaper</code>	64
Class option for old stock paper size.	
<code>\begin{oncolabstract}</code>	119
Environment for typesetting a one column abstract in a two column document.	
<code>\oncolglossary</code>	353
Declaration for a single column glossary.	
<code>\twocolglossary</code>	353
Declaration for a two column glossary.	
<code>\oncolindex</code>	341
Typeset index in one column.	
<code>onecolumn</code>	66
Class option for a single column.	
<code>\begin{OnehalfSpace}</code>	103
Environment form of <code>\OnehalfSpacing</code>	
<code>\OnehalfSpacing</code>	103
Declaration increasing the baseline to create the illusion of double spacing.	
<code>\onelineskip (length)</code>	376
Distance between baselines of the document's main font and size.	

<code>oneside</code>	Class option for text on only one side of the paper.	66
<code>\openany</code>	Allow chapters to start on the next new page.	128
<code>openany</code>	Class option for chapters to start on a recto or a verso page.	66
<code>openbib</code>	Class option for indenting continuation lines in a bibliography.	66
<code>\openinputfile{&lt;filename&gt;}{&lt;stream&gt;}</code>	Attaches the file <i>&lt;filename&gt;</i> to the input <i>&lt;stream&gt;</i> .	323
<code>\openleft</code>	Force chapters to start on a new verso page.	128
<code>openleft</code>	Class option for chapters to start on verso pages.	66
<code>\openoutputfile{&lt;filename&gt;}{&lt;stream&gt;}</code>	Attaches the file <i>&lt;filename&gt;</i> to the output <i>&lt;stream&gt;</i> .	322
<code>\openright</code>	Force chapters to start on a new recto page.	128
<code>openright</code>	Class option for chapters to start on recto pages.	66
<code>\ordinal{&lt;number&gt;}</code>	Typesets <i>&lt;number&gt;</i> as an ordinal number.	371
<code>\OrdinaltoName{&lt;number&gt;}</code>	Typeset <i>&lt;number&gt;</i> as an ordinal using lowercase words, but uppercase the initial letter of each word.	372
<code>\ordinaltoName{&lt;number&gt;}</code>	Typeset <i>&lt;number&gt;</i> as an ordinal using lowercase words, but uppercase the initial letter of the first word.	372
<code>\ordinaltoname{&lt;number&gt;}</code>	Typeset <i>&lt;number&gt;</i> as an ordinal using lowercase words.	372
<code>\ordscript{&lt;chars&gt;}</code>	Typesets the ordinal characters <i>&lt;chars&gt;</i> .	371
<code>\overridescapmargin{&lt;margin&gt;}</code>	A one-time override of <code>\sidecapmargin</code> .	242
<code>\pageinnotes{&lt;pagenum&gt;}</code>	Controls the printing of an endnote's page reference number.	359
<code>\pagenotesubhead{&lt;chapapp&gt;}{&lt;num&gt;}{&lt;title&gt;}</code>	Typesets a subheading for notes from chapter or appendix <i>&lt;chapapp&gt;</i> <i>&lt;num&gt;</i> called <i>&lt;title&gt;</i> .	359
<code>\pagenumbering{&lt;rep&gt;}</code>	Resets the page number to 1, and causes the folios (page numbers) to be printed using the <i>&lt;rep&gt;</i> representation (e.g., <i>arabic</i> , <i>roman</i> , ...)	163
<code>\pagenumbering*{&lt;rep&gt;}</code>	Like <code>\pagenumbering</code> except that the page number is not reset.	163
<code>\pageref{&lt;labstr&gt;}</code>	Prints the page number associated with <i>&lt;labstr&gt;</i> from a <code>\label</code> .	333
<code>\pagerefname</code>	Name for a page used by <code>\pref</code> .	334
<code>\pagestyle{&lt;style&gt;}</code>	Sets the current pagestyle to <i>&lt;style&gt;</i> .	164
<code>\pagetofootnote</code>	Declaration which turns <code>\pagenotes</code> into <code>\footnotes</code> .	360
<code>\paperheight (length)</code>	The height of the trimmed stock paper.	71
<code>\paperwidth (length)</code>	The width of the trimmed stock paper.	71
<code>\par</code>	Ends a paragraph.	101
<code>\paragraphfootnotes</code>	Typeset footnotes as a single paragraph.	275
<code>\paragraphfootstyle{&lt;style&gt;}</code>	Set the <i>&lt;series&gt;</i> footnotes to be typeset in single paragraph style.	276
<code>\parahook</code>	Hook called immediately before typesetting the title of a paragraph head.	155
<code>\parg{&lt;arg&gt;}</code>	prints ( <i>&lt;arg&gt;</i> ).	383
<code>\parindent</code>	Indentation at the start of the first line of a paragraph.	102
<code>\parskip</code>	Space between paragraphs.	101

<code>\parnopar</code> .....	281
Forces a new paragraph, but with no apparent break in the text.	
<code>\parskip</code> (Extra) vertical space between paragraphs (default 0pt). ....	177
<code>\part{&lt;title&gt;}</code> .....	122
Typesets a numbered part <i>&lt;title&gt;</i> on a page by itself, adding <i>&lt;title&gt;</i> to the ToC.	
<code>\partmark{&lt;title&gt;}</code> .....	127
For setting any marks with the title from a <code>\part</code> for a running header.	
<code>\partnamefont</code> Font used by <code>\printpartname</code> for the part name. ....	126
<code>\partnamenum</code> Called between printing a part name and number. ....	126
<code>\partnumfont</code> Font used by <code>\printpartnum</code> for the part number. ....	126
<code>\partrefname</code> Name for a part used by <code>\Pref</code> . ....	334
<code>\parttitlefont</code> Font used by <code>\printparttitle</code> for the title. ....	126
<code>\patchcommand{&lt;macro&gt;}{&lt;start-code&gt;}{&lt;end-code&gt;}</code> .....	368
Inserts <i>&lt;start-code&gt;</i> before the current definition of the <i>&lt;macro&gt;</i> and <i>{&lt;end-code&gt;}</i> at the end of the current definition.	
<code>\begin{patverse}</code> .....	299
Stanza lines are indented according to the <code>\indentpattern</code> ; lines after the pattern is completed are not indented.	
<code>\begin{patverse*}</code> .....	299
Like <i>patverse</i> except that the pattern will keep repeating.	
<i>pedersen</i> .....	142
A single line chapterstyle in large italics with the number set in the righthand margin. The title and/or number may be colored. The <code>graphicx</code> package is required and the <code>color</code> (or <code>xcolor</code> ) package if you want to color.	
<code>\pfbreak</code> .....	157
An alternative for <code>\plainfancybreak</code> using <code>\pfbreakskip</code> and <code>\pfbreakdisplay</code> .	
<code>\pfbreak*</code> .....	157
An alternative for <code>\plainfancybreak*</code> using <code>\pfbreakskip</code> and <code>\pfbreakdisplay</code> .	
<code>\pfbreakdisplay{&lt;text&gt;}</code> <i>&lt;text&gt;</i> for a <code>\pfbreak</code> 's <code>\fancybreak</code> . ....	157
<code>\pfbreakskip</code> Space for a <code>\pfbreak</code> 's <code>\plainbreak</code> . ....	157
<code>\phantomsection</code> .....	194
A macro to be put before <code>\addcontentsline</code> when the <code>hyperref</code> package is used.	
<code>\plainbreak{&lt;num&gt;}</code> .....	157
An anonymous division of <i>&lt;num&gt;</i> blank lines and the following paragraph is not indented.	
<code>\plainbreak*{&lt;num&gt;}</code> .....	157
Like <code>\plainbreak</code> except that the following paragraph is indented.	
<code>\plainfancybreak{&lt;space&gt;}{&lt;num&gt;}{&lt;text&gt;}</code> .....	157
An anonymous division that acts like <code>\fancybreak</code> at the top of the page, or at the bottom if there is less than <i>&lt;space&gt;</i> left on the page, otherwise it acts like <code>\plainbreak</code> . The following paragraph is not indented.	
<code>\plainfancybreak*{&lt;space&gt;}{&lt;num&gt;}{&lt;text&gt;}</code> .....	157
Like <code>\plainfancybreak</code> except that the following paragraph is indented.	
<code>\plainfootnotes</code> .....	275
Typeset footnotes as separate marked paragraphs (the default).	
<code>\plainfootstyle{&lt;series&gt;}</code> .....	276
Set the <i>&lt;series&gt;</i> footnotes to be typeset plain style.	
<code>\PlainPoemTitle</code> Declaration for <code>\PoemTitle</code> to be unnumbered. ....	301
<code>\pmname</code> .....	365
Abbreviation for post meridiem used in <code>\printtime*</code> (default am)	
<code>\pnchap</code> .....	359
Chapter title for <code>\pagenotesubhead</code> . Defaults to the ToC entry.	

<code>\pnschap</code> .....	359
Starred chapter title for <code>\pagenotesubhead</code> . Defaults to the regular title.	
<code>\PoemTitle</code> [ <i>{fortoc}</i> ][ <i>{forhead}</i> ]{ <i>{title}</i> } .....	301
Typesets the title for a poem and puts it into the ToC.	
<code>\PoemTitle*</code> [ <i>{fortoc}</i> ][ <i>{forhead}</i> ]{ <i>{title}</i> } .....	301
Typesets an unnumbered title for a poem but does not add it to the ToC.	
<code>\PoemTitlefont</code> Font for the text of a poem title .....	302
<code>\PoemTitleheadstart</code> Called at the start of typesetting a <code>\PoemTitle</code> . .....	301
<code>\poemtitlemark</code> { <i>{forhead}</i> } Used to set marks for a <code>\PoemTitle</code> . .....	301
<code>\PoemTitlenumfont</code> Font for the number of a poem title .....	302
<code>\poemtitlepstyle</code> Page style for a <code>\PoemTitle</code> . .....	301
<code>\poemtitlestarmark</code> { <i>{forhead}</i> } Used to set marks for a <code>\PoemTitle*</code> . .....	301
<code>\poemtitlestarpstyle</code> Page style for a <code>\PoemTitle*</code> . .....	301
<code>\poemtoc</code> Kind of entry for a <code>\PoemTitle</code> in the ToC. ....	301
<code>\postdate</code> { <i>{text}</i> } .....	111
Command processed after the <code>\date</code> in <code>\maketitle</code> .	
<code>\postauthor</code> { <i>{text}</i> } .....	111
Command processed after the <code>\author</code> in <code>\maketitle</code> .	
<code>\postbibhook</code> .....	338
Called after typesetting the list of of bibliographic entries.	
<code>\postcaption</code> { <i>{posttext}</i> } .....	227
<i>{posttext}</i> will be processed at the end of a caption.	
<code>\postnoteinnotes</code> .....	358
Called by <code>\noteentry</code> to finish the printing of an endnote.	
<code>\postitle</code> { <i>{text}</i> } .....	111
Command processed after the <code>\title</code> in <code>\maketitle</code> .	
<code>postvopaper</code> Class option for post octavo stock paper size. ....	64
<code>pottvopaper</code> Class option for pott octavo stock paper size. ....	64
<code>\preauthor</code> { <i>{text}</i> } .....	111
Command processed before the <code>\author</code> in <code>\maketitle</code> .	
<code>\prebibhook</code> .....	338
Called between typesetting the title of a bibliography and starting the list of bibliographic entries.	
<code>\precaption</code> { <i>{pretext}</i> } .....	227
<i>{pretext}</i> will be processed at the start of a caption.	
<code>\prechapterprecisshift</code> .....	151
Spacing between a chapter head and a chapter precis.	
<code>\precistocfont</code> Font for typesetting <code>\precistoc</code> text. ....	203
<code>\precistoc</code> text{ <i>{text}</i> } ToC entry for chapter precis <i>{text}</i> .....	203
<code>\predate</code> { <i>{text}</i> } .....	111
Command processed before the <code>\date</code> in <code>\maketitle</code> .	
<code>\Pref</code> { <i>{labstr}</i> } .....	334
Prints a named ( <code>\partrefname</code> ) reference to a <code>\labeled</code> part.	
<code>\pref</code> { <i>{labstr}</i> } .....	334
Prints a named ( <code>\pagerefname</code> ) reference to a <code>\label</code> page reference.	
<code>\preglossaryhook</code> .....	353
Vacuous macro called after a glossary title is typeset.	
<code>\preindexhook</code> .....	341
Called between typesetting an index's title and the start of the list.	
<code>\prenoteinnotes</code> .....	358
Called by <code>\noteentry</code> to initialise the printing of an endnote.	

<code>\pretitle{&lt;text&gt;}</code> .....	111
Command processed before the <code>\title</code> in <code>\maketitle</code> .	
<code>\printbookname</code> .....	126
Prints the book name ( <code>\bookname</code> ) using the <code>\booknamefont</code> .	
<code>\printbooknum</code> Prints a book number using the <code>\booknumfont</code> . ....	126
<code>\printbooktitle</code> Prints the book title using the <code>\booktitlefont</code> . ....	126
<code>\printchaptername</code> .....	129
Prints the chapter name using the <code>\chapnamefont</code> .	
<code>\printchapternonum</code> .....	130
Replaces printing the chapter name and number in unnumbered chapters.	
<code>\printchapternum</code> .....	129
Prints the chapter number using the <code>\chapnumfont</code> .	
<code>\printchaptertitle{&lt;title&gt;}</code> .....	129
Prints the chapter <code>&lt;title&gt;</code> using the <code>\chapttitlefont</code> .	
<code>\printglossary[&lt;file&gt;]</code> .....	350
Prints the glossary from file <code>\jobname.gls</code> , or <code>\file.gls</code>	
<code>\printindex[&lt;file&gt;]</code> .....	340
Print the sorted index. By default this is read from file <code>\jobname.ind</code> . If the optional argument is given it will read the data from file <code>&lt;file&gt;.ind</code> .	
<code>\printpageinnotes{&lt;pagenum&gt;}</code> .....	359
Prints an endnote's page reference number.	
<code>\printpagenotes</code> .....	356
Input the pagenote ent file for printing, then close it to any more notes.	
<code>\printpagenotes*</code> .....	356
Input the pagenote ent file for printing, then empty it ready for further notes.	
<code>\printpartname</code> .....	126
Prints the part name ( <code>\partname</code> ) using the <code>\partnamefont</code> .	
<code>\printpartnum</code> Prints a part number using the <code>\partnumfont</code> . ....	126
<code>\printparttitle</code> Prints the part title using the <code>\parttitlefont</code> . ....	126
<code>\printPoemTitlenonum</code> .....	301
Used instead of <code>\printPoemTitlenum</code> for an unnumbered <code>\PoemTitle</code> .	
<code>\printPoemTitlenum</code> Typesets the number for a <code>\PoemTitle</code> . ....	301
<code>\printPoemTitletitle{&lt;title&gt;}</code> Typesets the title of a <code>\PoemTitle</code> . ....	301
<code>\printtime</code> Prints the time of day using a 24 hour clock. ....	365
<code>\printtime*</code> Prints the time of day using a 12 hour clock. ....	365
<code>\printXtitle{&lt;title&gt;}</code> .....	195
Generic macro printing <code>&lt;title&gt;</code> as the title for the 'X List of'.	
<code>\providecounter{&lt;ctr&gt;}[&lt;within&gt;]</code> .....	366
A 'provide' version of <code>\newcounter</code> .	
<code>\provideenvironment{&lt;name&gt;}[&lt;nummarks&gt;][&lt;optarg&gt;]{&lt;begindef&gt;}{&lt;enddef&gt;}</code> .....	366
A 'provide' version of <code>\(re)newenvironment</code> .	
<code>\providefixedcaption[&lt;capcommand&gt;]{&lt;command&gt;}{&lt;float&gt;}</code> .....	233
A 'provide' version of <code>\newfixedcaption</code> .	
<code>\providelength{&lt;cmd&gt;}</code> A 'provide' version of <code>\newlength</code> . ....	366
<code>\provideloglike{&lt;cmd&gt;}{&lt;string&gt;}</code> .....	366
A 'provide' version of <code>\newloglike</code> .	
<code>\provideloglike*{&lt;cmd&gt;}{&lt;string&gt;}</code> .....	366
A 'provide' version of <code>\newloglike*</code> .	
<code>\qitem{&lt;text&gt;}{&lt;source&gt;}</code> .....	287
Typesets the <code>&lt;text&gt;</code> and <code>&lt;source&gt;</code> of an epigraph in an <code>epigrpahs</code> environment.	

<code>\quarkmarks</code>	363
Trim marks in the style of Quark Xpress registration marks, typeset with <code>\registrationColour</code> .	
<code>\begin{quotation}</code>	180
Contents set justified in a narrower measure with normal <code>\parindent</code> .	
<code>\begin{quote}</code>	180
Contents set justified in a narrower measure, with zero <code>\parindent</code> .	
<code>\raggedbottom</code>	67
Declaration allowing the last line on a page to be at a variable height.	
<code>\raggedbottomsection</code>	123
Pages will be typeset short because of a moved subhead as if <code>\raggedbottom</code> was in effect.	
<code>\raggedleft</code>	179
Declaration for text to be set raggedleft and flushright.	
<code>\raggedright</code>	179
Declaration for text to be set flushleft and raggedright.	
<code>\raggedyright[<i>&lt;space&gt;</i>]</code>	179
Version of <code>\raggedright</code> with <i>&lt;space&gt;</i> providing control over the amount of raggedness.	
<code>\ragnarparindent</code>	179
The <code>\parindent</code> for <code>\raggedyright</code> paragraphs.	
<code>\readboxedverbatim{<i>&lt;stream&gt;</i>}</code>	323
Acts like <code>boxedverbatim</code> except the contents is read from the file associated with the input <i>&lt;stream&gt;</i> .	
<code>\readboxedverbatim*{<i>&lt;stream&gt;</i>}</code>	323
Acts like <code>boxedverbatim*</code> except the contents is read from the file associated with the input <i>&lt;stream&gt;</i> .	
<code>\readline{<i>&lt;stream&gt;</i>}</code>	323
Reads a single line from the file associated with the input <i>&lt;stream&gt;</i> .	
<code>\readstream{<i>&lt;stream&gt;</i>}</code>	323
Reads the entire contents of the file associated with the input <i>&lt;stream&gt;</i> .	
<code>\readverbatim{<i>&lt;stream&gt;</i>}</code>	323
Acts like <code>verbatim</code> except the contents is read from the file associated with the input <i>&lt;stream&gt;</i> .	
<code>\readverbatim*{<i>&lt;stream&gt;</i>}</code>	323
Acts like <code>verbatim*</code> except the contents is read from the file associated with the input <i>&lt;stream&gt;</i> .	
<code>\ref{<i>&lt;labstr&gt;</i>}</code>	333
Prints the (section, or other) number associated with <i>&lt;labstr&gt;</i> from a <code>\label</code> .	
<code>\registrationColour{<i>&lt;mark&gt;</i>}</code>	363
Typesets <code>\quarkmarks</code> .	
<code>\renewfixedcaption[<i>&lt;capcommand&gt;</i>]{<i>&lt;command&gt;</i>}{<i>&lt;float&gt;</i>}</code>	233
A ‘renew’ version of <code>\newfixedcaption</code> .	
<code>reparticle</code>	132
With the article class option it replicates a section head in the article class.	
<code>\reportnoidxfile</code>	341
Report attempts to use an <code>idx</code> file that has not been declared by <code>\makeindex</code> .	
<code>\RequireAtEndClass{<i>&lt;class&gt;</i>}{<i>&lt;code&gt;</i>}</code>	381
Inserts <i>&lt;code&gt;</i> just after the <i>&lt;class&gt;</i> class is used, or immediately if <i>&lt;class&gt;</i> has already been used.	
<code>\RequireAtEndPackage{<i>&lt;pack&gt;</i>}{<i>&lt;code&gt;</i>}</code>	380
Inserts <i>&lt;code&gt;</i> just after the <i>&lt;pack&gt;</i> package is used, or immediately if <i>&lt;pack&gt;</i> has already been used.	
<code>\RequireXeTeX</code>	376
Generates an error if the document is not being processed by XeTeX.	
<code>\resetbvlinenumber</code>	318
Resets the <code>boxedverbatim</code> line number to zero.	
<code>\restorepagenumber</code>	163
Sets the page number to that saved by the most recent <code>\savepagenumber</code> .	

<code>\restoretrivseps</code> .....	186
Sets the current <code>\topsep</code> and <code>\partopsep</code> to the values saved by <code>\savetrivseps</code> .	
<code>\reversemarginpar</code> .....	280
Reverses the normal margins used by <code>\marginpar</code> .	
<code>\rightmark</code> .....	166
Contains the value of the <code>&lt;right&gt;</code> argument of the first <code>\markboth</code> or <code>\markright</code> on the page; if there is none then the value of the most recent <code>&lt;right&gt;</code> argument.	
<code>\rmfamily</code> Declaration for using a Roman font. ....	94
<code>royalvopaper</code> Class option for royal octavo stock paper size. ....	64
<code>\savepagenumber</code> Saves the current page number. ....	163
<code>\savetrivseps</code> .....	186
Stores the current <code>\topsep</code> and <code>\partopsep</code> for <code>trivlists</code> .	
<code>\saythanks</code> .....	119
Following a <code>onecolabstract</code> it ensures that <code>\thanks</code> are printed.	
<code>\scshape</code> Declaration for using a small caps font. ....	94
<code>\@seccntformat{&lt;code&gt;}</code> .....	154
Kernel macro that formats the number in a sectional head.	
<code>\sechook</code> .....	155
Hook called immediately before typesetting the title of a section head.	
<code>\section[&lt;toc-title&gt;][&lt;head-title&gt;]{&lt;title&gt;}</code> .....	122
Typesets a section subhead <code>&lt;title&gt;</code> , adding <code>&lt;title&gt;</code> to the ToC and possibly the running headers. If given <code>&lt;toc-title&gt;</code> is used instead of <code>&lt;title&gt;</code> for the ToC and running header. If given <code>&lt;head-title&gt;</code> is used for a running header.	
<code>\section{&lt;title&gt;}</code> .....	122
Typesets an unnumbered section subhead <code>&lt;title&gt;</code> . There are no ToC or running header entries.	
<code>section</code> Simple chapterstyle looking like a section head. ....	130
<code>\sectionrefname</code> Name for a section used by <code>\Sref</code> . ....	334
<code>\see</code> <i>see</i> entry in an index using <code>\seename</code> for the wording. ....	346
<code>\seealso</code> <i>see also</i> entry in an index using <code>\alsoname</code> for the wording. ....	346
<code>\seename</code> Wording for a <i>see</i> index entry. ....	346
<code>\semiisopage[&lt;spine&gt;]</code> .....	89
Generates a page layout especially suited to ISO proportioned paper but with smaller margins than <code>\isopage</code> .	
<code>\setafterparaskip{&lt;skip&gt;}</code> .....	154
Sets the <code>\afterskip</code> for a paragraph head.	
<code>\setaftersecskip{&lt;skip&gt;}</code> Sets the <code>\afterskip</code> for a section head. ....	154
<code>\setafterSskip{&lt;skip&gt;}</code> Sets the <code>\afterskip</code> for an S head. ....	154
<code>\setaftersubparaskip{&lt;skip&gt;}</code> .....	154
Sets the <code>\afterskip</code> for a subparagraph head.	
<code>\setaftersubsecskip{&lt;skip&gt;}</code> .....	154
Sets the <code>\afterskip</code> for a subsection head.	
<code>\setaftersubsubsecskip{&lt;skip&gt;}</code> .....	154
Sets the <code>\afterskip</code> for a subsubsection head.	
<code>\setarrayelement{&lt;arrayname&gt;}{&lt;index&gt;}{&lt;text&gt;}</code> .....	375
Makes <code>&lt;text&gt;</code> the contents of the <code>{&lt;index&gt;}</code> location in array <code>&lt;arrayname&gt;</code> .	
<code>\setbeforeparaskip{&lt;skip&gt;}</code> .....	153
Sets the <code>\beforeskip</code> for a <code>\paragraph</code> head.	
<code>\setbeforesecskip{&lt;skip&gt;}</code> .....	153
Sets the <code>\beforeskip</code> for a <code>\section</code> head.	
<code>\setbeforeSskip{&lt;skip&gt;}</code> Sets the <code>\beforeskip</code> for an S head. ....	153

---

<code>\setbeforesubparaskip{&lt;skip&gt;}</code> .....	153
Sets the <code>\beforeskip</code> for a <code>\subparagraph</code> head.	
<code>\setbeforesubsecskip{&lt;skip&gt;}</code> .....	153
Sets the <code>\beforeskip</code> for a <code>\subsection</code> head.	
<code>\setbeforesubsubsecskip{&lt;skip&gt;}</code> .....	153
Sets the <code>\beforeskip</code> for a <code>\subsubsection</code> head.	
<code>\setbiblabel{&lt;style&gt;}</code> .....	338
Define the look of the bibliographic entry identifiers.	
<code>\setbvlinenums{&lt;first&gt;}{&lt;startat&gt;}</code> .....	318
The first line of the following <code>boxedverbatim</code> is number <code>{&lt;first&gt;}</code> and the first printed line number should be <code>&lt;startat&gt;</code> .	
<code>\setcolsepandrule{&lt;colsep&gt;}{&lt;thickness&gt;}</code> .....	80
Sets the width of the gutter and the thickness of the rule in the gutter.	
<code>\setDisplayskipStretch{&lt;factor&gt;}</code> .....	103
Increase the display skips by <code>gmetafactor</code> .	
<code>\setfloatlocations{&lt;float&gt;}{&lt;locs&gt;}</code> .....	220
Sets the default location for the <code>&lt;float&gt;</code> (e.g., <code>table</code> ) to <code>&lt;locs&gt;</code> (default <code>tbp</code> ).	
<code>\sethangfrom{&lt;code&gt;}</code> User macro redefining <code>\@hangfrom</code> to <code>&lt;code&gt;</code> . .....	154
<code>\setheaderspaces{&lt;headdrop&gt;}{&lt;headsep&gt;}{&lt;ratio&gt;}</code> .....	80
Sets the spacing above and below a header.	
<code>\setheadfoot{&lt;headheight&gt;}{&lt;footskip&gt;}</code> .....	80
Sets the <code>headheight</code> and <code>footskip</code> .	
<code>\setlrmargins{&lt;spine&gt;}{&lt;edge&gt;}{&lt;ratio&gt;}</code> .....	77
Sets the spine and fore-edge margins for the current typeblock width.	
<code>\setlrmarginsandblock{&lt;spine&gt;}{&lt;edge&gt;}{&lt;ratio&gt;}</code> .....	78
Sets the spine and fore-edge margins, modifying the typeblock width to match.	
<code>\setlrvchars[&lt;fontspec&gt;]</code> .....	75
Sets <code>cslxvchars</code> to the length of 65 characters in the <code>&lt;fontspec&gt;</code> font (default <code>\normalfont</code> ).	
<code>\setmarginnotes{&lt;separation&gt;}{&lt;width&gt;}{&lt;push&gt;}</code> .....	81
Sets the length parameters for marginal notes.	
<code>\setpagebl{&lt;height&gt;}{&lt;width&gt;}{&lt;ratio&gt;}</code> .....	89
Specifies a page of the given dimensions positioned at the bottom left of the stock; see <code>\settrimmedsize</code> .	
<code>\setpageml{&lt;height&gt;}{&lt;width&gt;}{&lt;ratio&gt;}</code> .....	89
Specifies a page of the given dimensions positioned at the middle left of the stock; see <code>\settrimmedsize</code> .	
<code>\setpagetl{&lt;height&gt;}{&lt;width&gt;}{&lt;ratio&gt;}</code> .....	89
Specifies a page of the given dimensions positioned at the top left of the stock; see <code>\settrimmedsize</code> .	
<code>\setparaheadstyle{&lt;font&gt;}</code> Sets the style (font) for a paragraph head. ....	153
<code>\setparahook{&lt;text&gt;}</code> Redefines <code>\parahook</code> to be <code>&lt;text&gt;</code> . ....	155
<code>\setparaindent{&lt;length&gt;}</code> Sets the <code>\indent</code> for a <code>\paragraph</code> head. ....	153
<code>\setpnumwidth{&lt;length&gt;}</code> .....	196
Sets the width of the page number box ( <code>\@pnumwidth</code> ) in a ‘List of...’ to <code>&lt;length&gt;</code> .	
<code>\setrmarg{&lt;length&gt;}</code> .....	196
Sets the right hand title margin ( <code>\@tocrmarg</code> ) in a ‘List of...’ to <code>&lt;length&gt;</code> .	
<code>\setseheadstyle{&lt;font&gt;}</code> Sets the style (font) for a section head. ....	153
<code>\setsechook{&lt;text&gt;}</code> Redefines <code>\sechook</code> to be <code>&lt;text&gt;</code> . ....	155
<code>\setsecindent{&lt;length&gt;}</code> Sets the <code>\indent</code> for a <code>\section</code> head. ....	153
<code>\setsecnumdepth{&lt;secname&gt;}</code> .....	124
Sets division numbering level to <code>&lt;secname&gt;</code> .	

<code>\setsecnumformat{&lt;code&gt;}</code>	Redefines <code>\@secntformat</code> to <code>&lt;code&gt;</code> .	154
<code>\setshheadstyle{&lt;font&gt;}</code>	Sets the style (font) for an S head.	153
<code>\setshook{&lt;text&gt;}</code>	Redefines <code>\Shook</code> to be <code>&lt;text&gt;</code> .	155
<code>\setsidcappos{&lt;pos&gt;}</code>	Declaration of the vertical position of a sidecaption with respect to the float.	241
<code>\setsidebarheight{&lt;height&gt;}</code>	Sets the height of sidebars. The default is <code>\textheight</code> .	283
<code>\setsidebars{&lt;hsep&gt;}{&lt;width&gt;}{&lt;vsep&gt;}{&lt;topsep&gt;}{&lt;font&gt;}{&lt;height&gt;}</code>	Sets the several sidebar parameters.	284
<code>\setsidcaps{&lt;sep&gt;}{&lt;width&gt;}</code>	Sets the lengths <code>\sidecapsep</code> and <code>\sidcapwidth</code> to the given values.	240
<code>\setSindent{&lt;length&gt;}</code>	Sets the <code>\indent</code> for an S head.	153
<code>\SetSingleSpace{&lt;factor&gt;}</code>	Change the baselineskip by <code>&lt;factor&gt;</code> .	103
<code>\setstocksize{&lt;height&gt;}{&lt;width&gt;}</code>	Set the stock paper size to be <code>&lt;height&gt;</code> by <code>&lt;width&gt;</code> .	71
<code>\setsubparaheadstyle{&lt;font&gt;}</code>	Sets the style (font) for a subparagraph head.	153
<code>\setsubparahook{&lt;text&gt;}</code>	Redefines <code>\subparahook</code> to be <code>&lt;text&gt;</code> .	155
<code>\setsubparaindent{&lt;length&gt;}</code>	Sets the <code>\indent</code> for a <code>\subparagraph</code> head.	153
<code>\setsubsechheadstyle{&lt;font&gt;}</code>	Sets the style (font) for a subsection head.	153
<code>\setsubsechook{&lt;text&gt;}</code>	Redefines <code>\subsechook</code> to be <code>&lt;text&gt;</code> .	155
<code>\setsubsecindent{&lt;length&gt;}</code>	Sets the <code>\indent</code> for a <code>\subsection</code> head.	153
<code>\setsubsubsechheadstyle{&lt;font&gt;}</code>	Sets the style (font) for a subsubsection head.	153
<code>\setsubsubsechook{&lt;text&gt;}</code>	Redefines <code>\subsubsechook</code> to be <code>&lt;text&gt;</code> .	155
<code>\setsubsubsecindent{&lt;length&gt;}</code>	Sets the <code>\indent</code> for a <code>\subsubsection</code> head.	153
<code>\settocdepth{&lt;secname&gt;}</code>	Sets the value of the <code>tocdepth</code> counter in the <code>toc</code> file.	194
<code>\settrimmedsize{&lt;height&gt;}{&lt;width&gt;}{&lt;ratio&gt;}</code>	Sets the size of the trimmed stock paper.	71
<code>\settrims{&lt;top&gt;}{&lt;foreedge&gt;}</code>	Sets the amount to be trimmed from the top and fore-edge of the stock paper.	74
<code>\settypeblocksize{&lt;height&gt;}{&lt;width&gt;}{&lt;ratio&gt;}</code>	Sets the size of the typeblock.	76
<code>\setulmargins{&lt;upper&gt;}{&lt;lower&gt;}{&lt;ratio&gt;}</code>	Sets the upper and lower margins with the current typeblock height.	79
<code>\setulmarginsandblock{&lt;upper&gt;}{&lt;lower&gt;}{&lt;ratio&gt;}</code>	Sets the upper and lower margins, modifying the typeblock height to match.	79
<code>\setverbatimfont{&lt;fontspec&gt;}</code>	Sets the font to be used for verbatim text.	315
<code>\setverselinenums{&lt;first&gt;}{&lt;startat&gt;}</code>	The first line of the following verse is number <code>{&lt;first&gt;}</code> and the first printed line number should be <code>&lt;startat&gt;</code> .	300
<code>\setxlvchars[&lt;fontspec&gt;]</code>	Sets <code>\xlvchars</code> to the length of 45 characters in the <code>&lt;fontspec&gt;</code> font (default <code>\normalfont</code> ).	75
<code>\sffamily</code>	Declaration for using a Sans serif font.	94

---

<code>\begin{shaded}</code>	310
Put a colored background behind the contents of the environment, which can include page-breaks. The color extends into the margins a little.	
<code>sheetsequence</code>	364
Counter for sheets (similar to <code>page</code> for pages).	
<code>\Shook</code>	155
Hook called immediately before typesetting the title of an S head.	
<code>\shortsubcaption</code>	239
The subcaption version of <code>\shortcaption</code> .	
<code>\showcols</code>	254
Writes a list of all <code>\newcolumnntypes</code> to the terminal and log file.	
<code>\showindexmarks</code>	343
The <code>&lt;stuff&gt;</code> argument to <code>\index</code> and <code>\specialindex</code> will be printed in the margin (for use in noting what has been indexed where).	
<code>showtrims</code>	66
Class option for printing trimming marks.	
<code>\showtrimsoff</code>	362
Switch off any trim marks.	
<code>\showtrimson</code>	362
If the <code>showtrims</code> option has been used, switch on any trim marks (this is the default).	
<code>\sidebar{&lt;text&gt;}</code>	283
Typesets <code>&lt;text&gt;</code> in a sidebar.	
<code>\sidebarfont</code>	283
Font used for sidebars.	
<code>\sidebarform</code>	283
Form (e.g., <code>\raggedright</code> ) used for sidebars.	
<code>\sidebarhsep</code>	283
Space between the edge of the main text and sidebars.	
<code>\sidebarmargin{&lt;margin&gt;}</code>	283
Specifies the <code>&lt;margin&gt;(s)</code> for sidebars.	
<code>\sidebartopsep</code>	283
Controls the vertical position of a sidebar. The default is 0pt which aligns the tops of the type-block and the sidebar.	
<code>\sidebarvsep</code>	283
Vertical space between sidebars that fall on the same page.	
<code>\sidebarwidth</code>	283
Width of sidebars.	
<code>\sidecapfloatwidth{&lt;length&gt;}</code>	242
Macro holding the width of a float with a sidecaption.	
<code>\sidecapmargin{&lt;margin&gt;}</code>	240
Sets the the margin for sidecaptions.	
<code>\sidecapraise</code>	242
Vertical distance added to the default vertical placement of a sidecaption.	
<code>\sidecapsep</code>	240
Length specifying the horizontal separation between a sidecaption and the float.	
<code>\sidecapstyle</code>	241
Style settings for a sidecaption.	
<code>\begin{sidecaption}[&lt;fortoc&gt;]{&lt;title&gt;}[&lt;label&gt;]</code>	240
Environment for setting a sidecaption.	
<code>\sidecapwidth</code>	240
Length specifying the maximum width of a sidecaption.	
<code>\begin{sidecontcaption}{&lt;title&gt;}[&lt;label&gt;]</code>	241
Environment for setting a continued sidecaption.	
<code>\begin{sidelegend}{&lt;title&gt;}[&lt;label&gt;]</code>	241
Environment for setting a legend kind of sidecaption.	
<code>\begin{sidenamedlegend}{&lt;title&gt;}[&lt;label&gt;]</code>	241
Environment for setting a named legend kind of sidecaption.	
<code>\sidepar[&lt;left&gt;]{&lt;right&gt;}</code>	281
Like <code>\marginpar</code> except that the note does not move vertically.	
<code>\sideparvshift</code>	281
Move a <code>\sidepar</code> up/down by this amount.	
<code>\begin{SingleSpace}</code>	103
Environment form of <code>\SingleSpacing</code>	
<code>\SingleSpacing</code>	103
Declaration restoring normal single spacing (or that set by <code>\SetSingleSpace</code> ).	

<code>\slashfrac{⟨top⟩}{⟨bottom⟩}</code> .....	373
Typesets like $\frac{3}{4}$ , using the <code>\slashfracstyle</code> font.	
<code>\slashfracstyle{⟨num⟩}</code> .....	373
Typesets $\langle num \rangle$ in a particular (font, size) style.	
<code>\sloppy</code> .....	104
Declaration for TeX to allow large interword space variations in justified text lines.	
<code>\sloppybottom</code> .....	105
Declaration for TeX to allow an extra line at the bottom of a page. The <code>\topskip</code> must have been increased beforehand.	
<code>\begin{sloppypar}</code> .....	104
Typeset contents of the enclosed paragraph(s) using <code>\sloppy</code> .	
<code>\slshape</code> Declaration for using a slanted font. ....	94
<code>smalldemyvopaper</code> Class option for small demy octavo stock paper size. ....	64
<code>smallroyalvopaper</code> Class option for small royal octavo stock paper size. ....	64
<code>\begin{snugshade}</code> Like shaded but does not bleed into the margins. ....	310
<code>southall</code> .....	142
A raggedright chapterstyle with the number and title on the same line and a rule below.	
<code>\begin{SingleSpace}{⟨factor⟩}</code> .....	103
Environment form of <code>\SetSingleSpace</code>	
<code>\specialindex{⟨file⟩}{⟨counter⟩}{⟨stuff⟩}</code> .....	343
Add $\langle stuff \rangle$ and the current value of $\langle counter \rangle$ to the raw index data file $\langle file \rangle$ .idx.	
<code>\specialrule{⟨width⟩}{⟨abovespace⟩}{⟨belowspace⟩}</code> .....	260
Draws a rule with the given parameters across a tabular.	
<code>\Sref{⟨labstr⟩}</code> .....	334
Prints a named ( <code>\sectionrefname</code> ) reference to a <code>\labeled</code> section.	
<code>\stanzaskip</code> Vertical space between verse stanzas. ....	296
<code>statementpaper</code> Class option for statement stock paper size. ....	64
<code>\stockheight (length)</code> Height of the stock paper. ....	71
<code>\stockwidth (length)</code> Width of the stock paper. ....	71
<code>\strictpagecheck</code> .....	369
<code>\checkoddpaper</code> will use an accurate but time and space consuming method for checking for an odd page number.	
<code>\stringtoarray{⟨arrayname⟩}{⟨string⟩}</code> .....	375
Puts each character from $\langle string \rangle$ sequentially into array $\langle arrayname \rangle$ , starting at index 1.	
<code>\begin{subappendices}</code> .....	124
Like the <code>appendices</code> environment but used at the end of a chapter for per-chapter sub-appendices.	
<code>\subbottom[⟨list-entry⟩][⟨subtitle⟩]{⟨text⟩}</code> .....	237
Puts a subcaption identifier, and optionally $\langle subtitle \rangle$ , below $\langle text \rangle$ .	
<code>\subcaption[⟨list-entry⟩]{⟨subtitle⟩}</code> .....	237
Analogous to <code>\caption</code> but for an identified subcaption within a float.	
<code>\subcaptionfont{⟨fontspec⟩}</code> Font for subcaption titles. ....	239
<code>\subcaptionlabelfont{⟨fontspec⟩}</code> Font for subcaption identifiers. ....	239
<code>\subcaptionref{⟨labstr⟩}</code> .....	238
Print the subcaption identifier for a $\langle labstr \rangle$ labelled subcaption.	
<code>\subcaptionsize{⟨size⟩}</code> Font size for subcaptions. ....	239
<code>\subcaptionstyle{⟨style⟩}</code> Paragraph $\langle style \rangle$ for subcaptions. ....	239
<code>\subconcluded</code> .....	238
Indicates (to LaTeX) that a continued subfloat is finished.	
<code>\subitem</code> Introduces a subsidiary index entry. ....	343

<code>\subparahook</code> .....	155
Hook called immediately before typesetting the title of a subparagraph head.	
<code>\subsechook</code> .....	155
Hook called immediately before typesetting the title of a subsection head.	
<code>\subsubitem</code> Introduces a third level index entry. ....	343
<code>\subsubsechook</code> .....	155
Hook called immediately before typesetting the title of a subsubsection head.	
<code>\subtop</code> [ <i>(list-entry)</i> ] [ <i>(subtitle)</i> ] { <i>(text)</i> } .....	237
Puts a subcaption identifier, and optionally <i>(subtitle)</i> , on top of <i>(text)</i> .	
<code>superroyalvopaper</code> Class option for super royal octavo stock paper size. ....	64
<code>\suppressfloats</code> [ <i>(pos)</i> ] .....	220
Suppresses any floats on the current page at the given <i>(pos)</i> placement.	
<code>\symbolthanksmark</code> .....	114
Set the thanks marks to be printed using the footnote series of symbols.	
<code>\tabcolsep</code> Half the space between columns in a tabular. ....	267
<code>\tableofcontents</code> Typeset the ToC, adding its title to the ToC itself. ....	194
<code>\tableofcontents*</code> Typeset the ToC. ....	194
<code>\tablerefname</code> Name for a table used by <code>\tref</code> . ....	334
<code>\tabsoff</code> Ignore extra TAB spaces in a verbatim. ....	316
<code>\tabson</code> [ <i>(number)</i> ] .....	316
Set <i>(number)</i> of spaces in a verbatim for a TAB character; default 4.	
<code>\begin{tabular}</code> [ <i>(pos)</i> ] { <i>(preamble)</i> } .....	249
Environment for setting text elements in a tabular form.	
<code>\begin{tabular*}</code> { <i>(width)</i> } [ <i>(pos)</i> ] { <i>(preamble)</i> } .....	249
Environment for setting text elements in a tabular form within an overall <i>(width)</i> ; intercolumn spacing is adjusted to suit.	
<code>\begin{tabularx}</code> { <i>(width)</i> } [ <i>(pos)</i> ] { <i>(preamble)</i> } .....	249
Environment for setting text elements in a tabular form within an overall <i>(width)</i> ; column widths are adjusted to suit.	
<code>\tabularxcolumn</code> Column type for an X column in a <code>tabularx</code> . ....	265
<code>tandh</code> A simple section-like chapterstyle in a bold font. ....	144
<code>\tensunitsep</code> .....	372
The separator/conjoiner between tens and units in named numbers, default ‘-’.	
<code>\textbf</code> { <i>(text)</i> } Typeset <i>(text)</i> with a bold font. ....	94
<code>\textheight</code> (length) The height of the typeblock. ....	76
<code>\textit</code> { <i>(text)</i> } Typeset <i>(text)</i> with an italic font. ....	94
<code>\textmd</code> { <i>(text)</i> } Typeset <i>(text)</i> with a medium font. ....	94
<code>\textrm</code> { <i>(text)</i> } Typeset <i>(text)</i> with a Roman font. ....	94
<code>\textsc</code> { <i>(text)</i> } Typeset <i>(text)</i> with a small caps font. ....	94
<code>\textsf</code> { <i>(text)</i> } Typeset <i>(text)</i> with a Sans serif font. ....	94
<code>\textsl</code> { <i>(text)</i> } Typeset <i>(text)</i> with a slanted (oblique) font. ....	94
<code>\textsubscript</code> { <i>(sub)</i> } Typesets <i>(sub)</i> as a subscript. ....	374
<code>\textsuperscript</code> { <i>(super)</i> } Typesets <i>(super)</i> as a superscript. ....	374
<code>\texttt</code> { <i>(text)</i> } Typeset <i>(text)</i> with a Typewriter (monospaced) font. ....	94
<code>\textup</code> { <i>(text)</i> } Typeset <i>(text)</i> with an upright font. ....	94
<code>\textwidth</code> (length) The width of the typeblock. ....	76
<code>\thanksfootmark</code> Typesets a thanks mark at the foot. ....	115
<code>\thankheadextra</code> { <i>(pre)</i> }{ <i>(post)</i> } .....	114
Inserts <i>(pre)</i> and <i>(post)</i> before and after thanks markers in the titling code.	

<code>\thanksmark{&lt;n&gt;}</code> .....	115
Prints a thanks mark identical to the n'th (previously) printed mark.	
<code>\thanksmarksep</code> .....	115
Indentation of second and subsequent thanks text lines at the foot.	
<code>\thanksmarkseries{&lt;format&gt;}</code> .....	114
Thanks marks will be printed using <i>&lt;format&gt;</i> series of symbols.	
<code>\thanksmarkstyle{&lt;defn&gt;}</code> .....	115
Sets the style for the thanks marks at the foot.	
<code>\thanksmarkwidth</code> Width of box for the thanks marks at the foot. ....	115
<code>\thanksrule</code> The rule to be typeset before the thanks in the foot. ....	116
<code>thatcher</code> .....	144
A centered small caps chapterstyle with the number line separated from the title by a short rule.	
<code>\theauthor</code> Copy of <i>&lt;text&gt;</i> from <code>\author</code> . ....	113
<code>\begin{thebibliography}{&lt;exlabel&gt;}</code> .....	337
Environment for typesetting a bibliography. <i>&lt;exlabel&gt;</i> is an arbitrary piece of text as wide as the widest label for the bibliographic items.	
<code>\thectr</code> Typesets the value of the counter <code>ctr</code> . ....	365
<code>\thedata</code> Copy of <i>&lt;text&gt;</i> from <code>\date</code> . ....	113
<code>\begin{theglossary}</code> Environment for typesetting a glossary. ....	350
<code>\begin{theindex}</code> Environment for typesetting an index ....	341
<code>\thepoem</code> Typeset the current Poem Title number ....	301
<code>\thepoemline</code> .....	300
The numeric representation of verse line numbers (default arabic).	
<code>\thesheetsequence</code> Typesets the current sheet sequence number. ....	364
<code>\thetitle</code> Copy of <i>&lt;text&gt;</i> from <code>\title</code> . ....	113
<code>\thinspace</code> A thin space (3/18 em). ....	377
<code>\thispagestyle{&lt;style&gt;}</code> .....	164
Sets the pagestyle to <i>&lt;style&gt;</i> for the current page only.	
<code>\threecolumnfootnotes</code> Typeset footnotes in three columns. ....	275
<code>\threecolumnfootstyle{&lt;series&gt;}</code> .....	276
Set the <i>&lt;series&gt;</i> footnotes to be typeset in three column style.	
<code>\tightlist</code> In a standard list, removes extra vertical spacing. ....	183
<code>\tightlists</code> .....	183
Declaration removing extra vertical space from <code>list</code> -based environments.	
<code>\tightsubcaptions</code> .....	239
Specifies the default vertical space around subcaptions.	
<code>\title{&lt;text&gt;}</code> .....	113
Used by <code>\maketitle</code> to typeset <i>&lt;text&gt;</i> as the document title.	
<code>\titleref{&lt;labstr&gt;}</code> .....	335
Prints the (section, or other) title of the number associated with <i>&lt;labstr&gt;</i> from a <code>\label</code> .	
<code>\begin{titlingpage}</code> .....	112
Environment for a title page, but I don't recommend using it.	
<code>\tmarkbl</code> Trim mark for bottom left of trimmed page. ....	362
<code>\tmarkbm</code> Trim mark for bottom middle of trimmed page. ....	362
<code>\tmarkbr</code> Trim mark for bottom right of trimmed page. ....	362
<code>\tmarkml</code> Trim mark for middle left of trimmed page. ....	362
<code>\tmarkmr</code> Trim mark for middle right of trimmed page. ....	362
<code>\tmarktl</code> Trim mark for top left of trimmed page. ....	362
<code>\tmarktm</code> Trim mark for top middle of trimmed page. ....	362
<code>\tmarktr</code> Trim mark for top right of trimmed page. ....	362

<code>\tocnameref</code>	Use ToC title for sectional title references.	335
<code>\toprule</code>	<code>[\width]</code>	259
	Draws a rule across a tabular, default width <code>\heavyrulewidth</code> .	
<code>\topskip</code>	(length)	82
	The height of the first line of text on a page. This is usually less than the <code>\baselineskip</code> .	
<code>\tracingtabularx</code>		264
	Writes information about the changing column widths while setting a <code>tabularx</code> .	
<code>\traditionalparskip</code>		101
	Sets the interparagraph spacing to its traditional value.	
<code>\tref</code>	<code>\labstr</code>	334
	Prints a named ( <code>\tablerefname</code> ) reference to a <code>\labeled</code> table.	
<code>\trimedge</code>	(length)	74
	Amount to be trimmed from the fore-edge of the stock paper.	
<code>\trimFrame</code>		362
	Trim mark of a frame drawn round the trimmed page boundary.	
<code>\trimLmarks</code>		362
	Trim marks of ‘L’ shapes at the four corners of the trimmed page.	
<code>\trimmark</code>	Cross mark used by <code>\trimXmarks</code> .	363
<code>\trimmarks</code>		362
	Displays 8 (in)visible trim marks round the boundary of the trimmed page.	
<code>\trimNone</code>	No trim marks.	362
<code>\trimtop</code>	(length) Amount to be trimmed from the top of the stock paper.	74
<code>\trimXmarks</code>		362
	Trim marks of crosses at the four corners of the trimmed page.	
<code>\ttfamily</code>	Declaration for using a Typewriter (monospaced) font.	94
<code>\twocolindex</code>	Typeset index in two columns (the default).	341
<code>twocolumn</code>	Class option for two columns.	66
<code>\twocolumnfootnotes</code>	Typeset footnotes in two columns.	275
<code>\twocolumnfootstyle</code>	<code>\series</code>	276
	Set the <code>\series</code> footnotes to be typeset in two column style.	
<code>twoside</code>	Class option for text on both sides of the paper.	66
<code>\TXoverb</code>	A poor man’s <code>\verb</code> for use in a <code>tabularx</code> .	266
<code>\typeoutlayout</code>		84
	Outputs the current values of the class layout parameters to the terminal and log file.	
<code>\typeoutstandardlayout</code>		84
	Outputs the current values of the standard LaTeX layout parameters to the terminal and log file.	
<code>\ucminusname</code>		373
	Lowercase ‘minus’ name with initial uppercase letter, default ‘Minus’.	
<code>\undodrop</code>		290
	Following a <code>\dropchapter</code> restores subsequent chapter heads to their normal position.	
<code>\upbracefill</code>	Fills a tabular column with an up brace.	261
<code>\uppercaseheads</code>		170
	Defines <code>\memUChed</code> as equivalent to <code>\MakeUppercase</code> .	
<code>\uppercaseheads</code>	Set the titles in the headings pagestyle in Uppercase.	165
<code>\upshape</code>	Declaration for using an upright font.	94
<code>\usethanksrule</code>		116
	Specifies that the <code>\thanksrule</code> is to be typeset in the <code>titlingpage</code> environment.	

<i>veelo</i> .....	144
A raggedleft large bold chapterstyle with a large black square in the margin by the number line.	
It requires the graphicx package.	
<code>\begin{verbatim}</code> Typeset the contents verbatim. ....	316
<code>\verbatimbreakchar{&lt;char&gt;}</code> .....	316
Character indicating a verbatim line is being wrapped.	
<code>\verbatimindent</code> Indent for wrapped overlong verbatim lines. ....	316
<code>\verbatiminput{&lt;file&gt;}</code> .....	323
Acts like <code>verbatim</code> except the contents is read from the <i>&lt;file&gt;</i> file.	
<code>\verbatiminput*{&lt;file&gt;}</code> .....	323
Acts like <code>verbatim*</code> except the contents is read from the <i>&lt;file&gt;</i> file.	
<code>\begin{verbatimoutput}{&lt;file&gt;}</code> .....	322
The contents of the environment are written verbatim to the <i>&lt;file&gt;</i> file, overwriting anything previously in the file.	
<code>\verbfootnote[&lt;num&gt;]{&lt;text&gt;}</code> .....	274
Like <code>\footnote</code> except that <i>&lt;text&gt;</i> can contain verbatim material.	
<code>\begin{verse}[&lt;length&gt;]</code> .....	295
Environment for typesetting verse; if given the midpoint of <i>&lt;length&gt;</i> is placed at the center of the typeblock measure.	
<code>\verselinebreak[&lt;length&gt;]</code> .....	297
Makes a break in a verse line, indenting the next part by twice <code>\vgap</code> , or by <i>&lt;length&gt;</i> if it is given.	
<code>\verselinenumbersleft</code> .....	300
Following this declaration verse line numbers are set at the left of the verse lines.	
<code>\verselinenumbersright</code> .....	300
Following this declaration verse line numbers are set at the right of the verse lines.	
<code>\versewidth</code> Scratch length, typically for use in verse typesetting. ....	295
<i>verville</i> .....	144
A single line, large, centered, chapterstyle with rules above and below.	
<code>\vinphantom{&lt;text&gt;}</code> Leaves a space as wide as <i>&lt;text&gt;</i> . ....	297
<code>\leftmargin</code> General verse indent from the left of the typeblock. ....	296
<code>\leftofline{&lt;text&gt;}</code> ‘Hanging left’ <i>&lt;text&gt;</i> at the start of a verse line. ....	298
<code>\leftskip</code> .....	300
Space between the argument to <code>\flageverse</code> and <code>\flagverse</code> .	
<code>\begin{vplace}[&lt;num&gt;]</code> .....	379
The contents of this environment are centered vertically. The optional <i>&lt;num&gt;</i> argument can be used to specify the ratio of the upper space to the lower space.	
<code>\rightskip</code> .....	300
Verse line numbers are set distance <code>\rightskip</code> into the margin.	
 <i>wilsondob</i> A one line flushright chapterstyle in a large italic font. ....	144
<code>\wrappingoff</code> .....	316
The normal behaviour of not wrapping overlong verbatim lines.	
<code>\wrappingon</code> Wrap overlong verbatim lines. ....	316
<code>\begin{writeverbatim}{&lt;stream&gt;}</code> .....	322
The contents of the environment are written verbatim to the <i>&lt;stream&gt;</i> stream.	
 <code>\Xheadstart</code> Generic macro called before printing a ‘X List of’ title. ....	195
<code>\xlvchars (length)</code> Approximate length of 45 characters. ....	75
<code>\Xmark</code> Generic macro setting the marks for the ‘X List of’. ....	195

<code>\zerotrivseps</code> Eliminate space before and after a <code>trivlist</code> . . . . .	186
-----------------------------------------------------------------------------------------------	-----



---

## 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>)
- [Ado01] *How to Create Adobe PDF eBooks*. Adobe Systems Inc., 2001. (Available from <http://www.adobe.com/epaper/tips/acr5ebook/pdfs/eBook.pdf>)
- [Ars99] Donald Arseneau. *The url package*. February, 1999. (Available from CTAN as `/macros/latex/contrib/misc/url.sty`)
- [Ars01a] Donald Arseneau. *The titleref package*. April, 2001. (Available from CTAN as `/macros/latex/contrib/misc/titleref.sty`)
- [Ars01b] Donald Arseneau. *The chapterbib package*. September, 2001. (Available from CTAN as `/macros/latex/contrib/cite/chapterbib.sty`)
- [Ars07] Donald Arseneau. *The framed package* v0.95. October, 2007. (Available from CTAN as `/macros/latex/contrib/misc/framed.sty`)
- [Bar92] Helen Barolini. *Aldus and his Dream Book*. Italica Press, 1992. ISBN 0-934977-22-4.
- [BDG89] Charles Bigelow, Paul Hayden Duensing and Linnea Gentry (Eds). *Fine Print on Type*. 1989. Fine Print, CA (ISBN 0-9607290-X) or Bedford Arts, CA (ISBN 0-938491-17-2).
- [Boh90] Robert Bohle. *Publication Design for Editors*. Prentice-Hall, 1990.
- [Ber02] Jens Berger. *The titlesec and titletoc packages*. September, 2002. (Available from CTAN in `/macros/latex/contrib/titlesec`)
- [Bez99] Javier Bezos. *The titlesec and titletoc packages*. February, 1999. (Available from CTAN in `/macros/latex/contrib/titlesec`)
- [Bra94] Johannes Braams *et al.* *Standard LaTeX2e packages makeidx and showidx*. November, 1994. (Available from CTAN as `/macros/latex/base/makeidx.dtx(ins)`)
- [Bra97] Johannes Braams. *The alltt environment*. June, 1997. (Available from CTAN as `/macros/latex/base/alltt.dtx(ins)`)
- [Bri99] Robert Bringhurst. *The Elements of Typographic Style*. Hartley & Marks, second edition, 1999. ISBN 0-88179-033-8.
- [Bur59] C. L. Burt. *A Psychological Study of Typography*. Cambridge University Press, 1959.
- [Car94] David Carlisle. *The delarray package*. March, 1994. (Available from CTAN in `/macros/latex/required/tools`)
- [Car95] David Carlisle. *The afterpage package*. October, 1995. (Available from CTAN in `/macros/latex/required/tools`)

## BIBLIOGRAPHY

---

- [Car98b] David Carlisle. *The longtable package*. May, 1998. (Available from CTAN in `/macros/latex/required/tools`)
- [Car98c] David Carlisle. *The enumerate package*. August, 1998. (Available from CTAN in `/macros/latex/required/tools`)
- [Car98d] David Carlisle. *The remreset package*. August, 1998. (Available from CTAN in `/macros/latex/contrib/carlisle`)
- [Car99] David Carlisle. *The tabularx package*. January, 1999. (Available from CTAN in `/macros/latex/required/tools`)
- [CR99] David Carlisle and Sebastian Rahtz. *The graphicx package*. February, 1999. (Available from CTAN in `/macros/latex/required/graphics`)
- [Car01] David Carlisle. *The dcolumn package*. May, 2001. (Available from CTAN in `/macros/latex/required/tools`)
- [Car05] David Carlisle. *Packages in the graphics bundle* (includes the color package). November, 2005. (Available from CTAN in `/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 in `/indexing/makeindex/paper`)
- [Chi93] *The Chicago Manual of Style*, Fourteenth Edition. The University of Chicago, 1993. ISBN 0-226-10389-7.
- [Coc02] Steven Douglas Cochran. *The subfigure package*. March, 2002. (Available from CTAN in `/macros/latex/contrib/subfigure`)
- [CG96] John H. Conway and Richard K. Guy. *The Book of Numbers*. Copernicus, Springer-Verlag, 1996. ISBN 0-387-97993-X.
- [Cra92] James Craig. *Designing with Type: A Basic Course in Typography*. Watson-Guptill, NY, 1992.
- [Dal99a] Patrick W. Daly. *Natural Sciences Citations and References*. May, 1999. (Available from CTAN in `/macros/latex/contrib/natbib`)
- [Dal99b] Patrick W. Daly. *Customizing Bibliographic Style Files*. August, 1999. (Available from CTAN in `/macros/latex/contrib/custom-bib`)
- [Deg92] Asaf Degani. *On the Typography of Flight-Deck Documentation*. NASA Contractor Report # 177605. December, 1992. (Available from <http://members.aol.com/willadams/typgrphy.htm#NASA>)
- [Dow96] Geoffrey Dowding. *Finer Points in the Spacing & Arrangement of Type*. Hartley & Marks, 1996. ISBN 0-88179-119-9.
- [Dow98] Geoffrey Dowding. *An Introduction to the History of Printing Types*. The British Library and Oak Knoll Press, 1998. ISBN 0-7123-4563-9 UK, 1-884718-44-2 USA.
- [Dow00] Michael J. Downes. *The patchcmd package*. July, 2000. (Available from CTAN in `/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 in `/macros/??????????`)

- 
- [Fai98] Robin Fairbairns. *The moreverb package*. December, 1998. (Available from CTAN in `/macros/latex/contrib/moreverb`)
  - [Fai00] Robin Fairbairns. *footmisc — a portmanteau package for customising footnotes in LaTeX2e*. March, 2000. (Available from CTAN in `/macros/latex/contrib/footmisc`)
  - [FAQ] Robin Fairbairns. *The UK TeX FAQ*. (Available from CTAN in `/help/uk-tex-faq`)
  - [Fea03] Simon Fear. *Publication quality tables in LaTeX*. March, 2003. (Available from CTAN in `/macros/latex/contrib/booktabs`)
  - [Flu98] Daniel Flipo. *Typesetting ‘lettrines’ in LaTeX2e documents*. March, 1998. (Available from CTAN in `/macros/latex/contrib/lettrine`)
  - [Fly98] Peter Flynn. *Formatting Information: A Beginner’s Introduction to Typesetting with LaTeX2*. 2002. (Available from CTAN in `/info/beginlatex`)
  - [Fra00] Melchior Franz. *The crop package*. February, 2000. (Available from CTAN in `/macros/latex/contrib/crop`)
  - [FOS98] Friedrich Friedl, Nicolaus Ott and Bernard Stein. *Typography: An Encyclopedic Survey of Type Designs and Techniques throughout History*. Black Dog & Leventhal Publishers Inc., 1998. ISBN 1-57912-023-7.
  - [Gar66] Martin Gardner. *More Mathematical Puzzles and Diversions*. Penguin Books, 1996. ISBN 0-14-020748-1.
  - [GM<sup>+</sup>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.
  - [Gou87] J. D. Gould *et al.* ‘Reading from CRT displays can be as fast as reading from paper’. *Human Factors*, pp 497–517, 29:5, 1987.
  - [HR83] J. Hartley and D. Rooum. ‘Sir Cyril Burt and typography’. *British Journal of Psychology*, pp 203–212, 74:2, 1983.
  - [HM01] Steven Heller and Philip B. Meggs (Eds). *Texts on Type: Critical Writings on Typography*. Allworth Press, 2001. ISBN 1-58115-082-2.
  - [Hoe98] Alan Hoenig. *TeX Unbound: LaTeX and TeX strategies for fonts, graphics, and more*. Oxford University Press, 1998. ISBN 0-19-509686-X.
  - [HK75] J. K. Hvistendahl and M. R. Kahl. ‘Roman vs. sans serif body type: Readability and reader preference’. *AANPA News Research Bulletin*, pp 3–11, 17 Jan., 1975.
  - [Jon95] David M. Jones. *A new implementation of LaTeX’s indexing commands*. September, 1995. (Available from CTAN in `/macros/latex/contrib/camel`)
  - [Keh98] Roger Kehr. *xindy: A flexible indexing system*. February, 1998. (Available from CTAN in `/indexing/xindy`)
  - [Ker07] Uwe Kern. *Extending LaTeX’s color facilities: the xcolor package*. January, 2007. (Available from CTAN in `/macros/latex/contrib/xcolor`)
  - [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.

- 476

- 
- [Oet] Tobias Oetiker. *The Not So Short Introduction to LaTeX2e*. (Available from CTAN in `/info/lshort/english`)
  - [Oos96] Piet van Oostrum. *Page Layout in LaTeX*. June, 1996. (Available from CTAN in `/macros/latex/contrib/fancyhdr`)
  - [Pak01] Scott Pakin. *The Comprehensive LaTeX Symbol List*. July, 2001. (Available from CTAN in `/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/distrib/doc/btxdoc.tex`)
  - [Pat88b] Oren Patashnik. *Designing BibTeX Styles*. February, 1988. (Available from CTAN as `/bibliography/bibtex/distrib/doc/btxhak.tex`)
  - [Pug02] Diego Puga. *The Pazo Math fonts for mathematical typesetting with the Palatino fonts*. May, 2002. (Available from CTAN in `/fonts/mathpazo`)
  - [Rahtz01] Sebastian Rahtz. *Section name references in LaTeX*. January, 2001. (Available from CTAN in `/macros/latex/contrib/hyperref`)
  - [Rahtz02] Sebastian Rahtz. *Hypertext marks in LaTeX*. May, 2002. (Available from CTAN in `/macros/latex/contrib/hyperref`)
  - [Rec97] Keith Reckdahl. *Using Imported Graphics in LaTeX2e*. December, 1997. (Available from CTAN as `/info/epspatex.ps` or `/info/epslatex.pdf`)
  - [Reh72] Rolf Rehe. 'Type and how to make it most legible'. *Design Research International*, 1972.
  - [Rei07] Edward M. Reingold. 'Writing numbers in words in TeX'. *TUGboat*, 28, 2 pp 256–259, 2007.
  - [RAE71] D. O. Robinson, M. Abbamonte and S. H. Evans. 'Why serifs are important: The perception of small print'. *Visible Language*, pp 353–359, 4, 1971.
  - [Rog43] Bruce Rogers. *Paragraphs on Printing*. William E. Rudge's Sons, Inc., 1943. (Reissued by Dover, 1979, ISBN 0–486–23817–2)
  - [Rog49] Bruce Rogers. *Centaur Types*. October House, 1949.
  - [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 in `/macros/latex/required/tools`)
  - [Sch97] Karen A. Schriver. *Dynamics in Document Design*. Wiley & Sons, 1997.
  - [Tal06] Nicola L. C. Talbot. *datetime.sty: Formatting Current Date and Time*. December, 2006. (Available from CTAN in `/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 in `/macros/latex/required`)
- [Tin63] Miles A. Tinker. *Legibility of Print*. Books on Demand (University Microfilms International), 1963.
- [Tob00] Geoffrey Tobin. *setspace.sty*. December, 2000. (Available from CTAN in `/macros/latex/contrib/setspace`)
- [Tsc91] Jan Tschichold. *The Form of the Book*. Lund Humphries, 1991. ISBN 0-85331-623-6.
- [Tuf83] Edward R. Tufte. *The Visual Display of Quantative Information*. Graphics Press, 1983.
- [Ume99] Hideo Umeki. *The geometry package*. November, 1999. (Available from CTAN in `/macros/latex/contrib/geometry`)
- [Whe95] Colin Wheildon. *Type & Layout*. Strathmore Press, 1995. ISBN 0-9624891-5-8.
- [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 in `/macros/latex/contrib/tocvsec2`)
- [Wil00a] Peter Wilson. *The epigraph package*. February, 2000. (Available from CTAN in `/macros/latex/contrib/epigraph`)
- [Wil00b] Peter Wilson. *LaTeX files for typesetting ISO standards*. February, 2000. (Available from CTAN in `/macros/latex/contrib/isostds/iso`)
- [Wil00c] Peter Wilson. *The nextpage package*. February, 2000. (Available from CTAN as `/macros/latex/contrib/misc/nextpage.sty`)
- [Wil00d] Peter Wilson. *The needspace package*. March, 2000. (Available from CTAN as `/macros/latex/contrib/misc/needspace.sty`)
- [Wil00e] Peter Wilson. *The xtab package*. April 2000. (Available from CTAN in `macros/latex/contrib/xtab`)
- [Wil01a] Peter Wilson. *The abstract package*. February, 2001. (Available from CTAN in `/macros/latex/contrib/abstract`)
- [Wil01b] Peter Wilson. *The chngpage package*. February, 2001. (Available from CTAN as `/macros/latex/contrib/misc/chngpage.sty`)
- [Wil01c] Peter Wilson. *The appendix package*. March, 2001. (Available from CTAN in `/macros/latex/contrib/appendix`)
- [Wil01d] Peter Wilson. *The ccaption package*. March, 2001. (Available from CTAN in `/macros/latex/contrib/ccaption`)
- [Wil01e] Peter Wilson. *The chngcntr package*. April, 2001. (Available from CTAN as `/macros/latex/contrib/misc/chngcntr.sty`)
- [Wil01f] Peter Wilson. *The hanging package*. March, 2001. (Available from CTAN in `/macros/latex/contrib/hanging`)
- [Wil01g] Peter Wilson. *The titling package*. March, 2001. (Available from CTAN in `/macros/latex/contrib/titling`)
- [Wil01h] Peter Wilson. *The tocbibind package*. April, 2001. (Available from CTAN in `/macros/latex/contrib/tocbibind`)

- [Wil01i] Peter Wilson. *The tocloft package*. April, 2001. (Available from CTAN in `/macros/latex/contrib/tocloft`)
- [Wil01j] Peter Wilson. *The LaTeX memoir class for configurable book typesetting: Source code*. July, 2001. (Available from CTAN in `/macros/latex/contrib/memoir`)
- [Wil01k] Peter Wilson. *Typesetting simple verse with LaTeX*. July, 2001. (Available from CTAN in `/macros/latex/contrib/verse`)
- [Wil01l] Peter Wilson. *Printing booklets with LaTeX*. August, 2001. (Available from CTAN in `/macros/latex/contrib/booklet`)
- [Wil03a] Peter Wilson. *The layouts package*. November, 2003. (Available from CTAN in `/macros/latex/contrib/layouts`)
- [Wil03b] Peter Wilson. *ledmac: A presumptuous attempt to port EDMAC and TABMAC to LaTeX*. November, 2003. (Available from CTAN in `/macros/latex/contrib/ledmac`)
- [Wil04a] Peter Wilson. ‘Glisterings’. TUGboat, 25, 2 pp 201–202, 2004.
- [Wil04b] Peter Wilson. *The pagenote package*. September, 2004. (Available from CTAN in `/macros/latex/contrib/pagenote`)
- [Wil07a] Peter Wilson. *Some Examples of Title Pages*. Herries Press, 2007. (Available from CTAN as `info/latex-samples/titlepages.pdf`)
- [Wil07b] Peter Wilson. *The LaTeX memoir class for configurable book typesetting: source code*. November, 2007. (Available from CTAN in `/macros/latex/contrib/memoir`)
- [Wil07c] Peter Wilson. *The Memoir Class for Configurable Typesetting — User Guide*. November, 2007. (Available from CTAN in `/macros/latex/contrib/memoir`)
- [Wil07d] Peter Wilson. *ADDENDUM: The Memoir Class for Configurable Typesetting — User Guide*. November, 2007. (Available from CTAN in `/macros/latex/contrib/memoir`)
- [Wil08] Peter Wilson. *The changepage package*. March, 2008. (Available from CTAN as `/macros/latex/contrib/misc/changepage.sty`)
- [Wil??] Peter Wilson. *A Rumour of Humour: A scientist’s commonplace book*. To be published?
- [Wul53] Emerson G. Wulling. *A Comp’s-Eye View of Footnotes*. Sumac Press, 1953.
- [Zac69] B. Zachrisson. *Studies in the Legibility of Printed Text*. Almqvist & Wiksell, Stockholm, 1969.
- [Zan98] Timothy Van Zandt. *Documentation for fancybox.sty: Box tips and tricks for LaTeX*, November, 1998. (Available from CTAN in `/macros/latex/contrib/fancybox`)
- [Zap00] Hermann Zapf. *The Fine Art of Letters*. The Grolier Club, 2000. ISBN 0–910672–35–0.



---

# Index

---

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

## Alphabets

- \!, 377
- ! (level specifier), 344
- !h float specifier ..., 428
- \#, 418, 422
- \&, 419
- \(, 417, 424
- \), 424
- \*pt (option), 65, 432
- \+, 427
- \,, 377
- \-, 423, 427
- ... allowed only in math mode, 424
- ... at index ... in pattern ..., 432
- ... is negative, 430
- ... is not a counter, 430
- ... is not a macro, 432
- ... is not an input stream, 432
- ... is not an output stream, 432
- ... is zero or negative, 430
- \:, 377
- \;, 377
- \<, 424, 427
- <{...}at wrong position ..., 430
- \=, 426, 427
- =, 404
- \>, 427
- >{...}at wrong position ..., 430
- @,
  - in macro code, 193, 210, 229, 245, 391, 403, 422
- @ (actual specifier), 344
- @-expression, 426
- \@@end, 421
- \@@wrglom@m, 351, 352
- \@MM, 406
- \@Mi, 406
- \@Mii, 406
- \@Miii, 406
- \@Miv, 406
- \@bsphack, 329
- \@caption, 244, 245
- \@capttype, 244
- \@ccclv, 406
- \@ccclvi, 406
- \@cftasnum, 204
- \@cftasnumb, 204
- \@cftbsnum, 204
- \@chapap, 134
- \@dblpfbot (length), 223, 224
- \@dblpfsep (length), 223
- \@dblpftop (length), 223, 224
- \@dblftop (length), 224
- \@dotsep, 191, 193, 199
- \@dottedtocline, 192
- \@esphack, 329
- \@fibnext, 413
- \@fibseries, 412–414
- \@fnsymbol, 279
- \@fpbot (length), 223, 224
- \@fpsep (length), 223, 224
- \@fptop (length), 223, 224
- \@hangfrom, 154, 178, 389
- \@m, 406
- \@makecaption, 244–246

\@makefnmark, 277  
\@minus, 407  
\@namedef, 404  
\@nameuse, 404  
\@ne, 406  
\@plus, 407  
\@pnumwidth, 191, 196, 202  
\@pnumwidth (length), 199  
\@secntformat, 154, 155  
\@setref, 330  
\@startsection, 210  
\@tempdima, 411  
\@tempdimb, 411  
\@thefnmark, 277, 278  
\@tocrmarg, 191, 192, 196, 202  
\@whiledim, 414  
\@whilenum, 414  
\@xfloat, 244  
\@xxxii, 406  
\@zerosecs, 320, 397  
\[, 249, 255, 417, 424  
\\, 123, 227, 228, 249, 259, 260, 265, 293,  
295–297, 397, 417, 419, 420, 422,  
427  
\ (escape specifier), 347  
\\\*, 227, 295, 296  
\\>, 297, 304  
\\!, 296  
\< in mid line, 424  
\LoadClass in package file, 425  
\include cannot be nested, 425  
\verb illegal in command argument, 309  
\], 249, 255, 424  
| (encap specifier), 346  
10pt (option), 64, 65, 67, 98  
11pt (option), 64, 98  
12pt (option), 64, 67, 98  
14pt (option), 64, 98, 168  
16mo, 7  
17pt (option), 64, 65, 98, 431  
20pt (option), 64, 65, 98  
25pt (option), 64, 98, 431  
2em-dash, 53  
30pt (option), 64, 98  
32mo, 7  
36pt (option), 65, 98  
3em-dash, 54

48pt (option), 65, 98  
4to, 7  
60pt (option), 65, 98  
64mo, 7  
8vo, 7  
9pt (option), 64, 98

## A

A box was supposed to be here, 416  
A pattern has not been specified, 430  
a3paper (option), 63  
A4 paper, 69  
a4paper (option), 63, 86, 88  
a5paper (option), 63  
a6paper (option), 63  
abbreviation, 6, 51  
\abnormalparskip, 101, 102  
\abovecaptionskip (length), 245  
\aboverulesep (length), 259–261  
\abslabeldelim, 118  
\absleftindent (length), 118  
\absnamepos, 118  
\absparindent (length), 118, 119  
\absparsep (length), 118  
\absrightindent (length), 118  
\abstitlekip (length), 118, 119  
abstract, 112, 117  
heading, 117  
one column, 119  
styling, 117–119  
abstract (environment), xiii, 117–119,  
378  
abstract (package), xix, 117, 380  
\abstractcol, 117, 118  
\abstractintoc, 117, 118  
\abstractname, 117, 118, 378  
\abstractnamefont, 118  
\abstractnum, 117, 118  
\abstractrunin, 117–119  
\abstracttextfont, 118  
acknowledgements, 44  
acronym, 52  
act  
play, 49  
Adams, William, 136, 363  
add to a macro, 367

- add to contents, 192
- \addappheadtotoc, 123, 124
- \addcontentsline, 189, 190, 192–194, 211, 231
- \added, 362
- \addlinespace, 260
- \addpenalty, 426
- \addperiod, 377
- \addtocontents, 192, 193, 202, 211
- \addtocounter, 163, 424, 426
- \addtodef, 86, 208, 367, 432
- \addtodef\*, 367
- \addtoiargdef, 367, 432
- \addtoiargdef\*, 367
- \addtolength, 81, 118, 129, 183, 296, 409
- \addtonotes, 359
- \addtopsmarks, 173
- \addtostream, 322
- \addvspace, 130, 426
- adjustwidth (environment), 180, 181, 183, 313
- adjustwidth\* (environment), 113, 180
- adjuwidth (environment), 135
- Adobe Garamond, 23
- \advance, 407–409
- \afterbookskip, 125
- \afterchapskip (length), 128, 129
- \afterchapternum, 128, 129, 134
- \afterchaptertitle, 128, 129, 195, 388
- \afterepigraphskip (length), 289
- \afternextrecto, 210
- \afternextverso, 210
- \afterpage, 210
- afterpage (package), 210, 233
- \afterpartskip, 125, 292
- \afterPoemTitle, 301, 302
- \afterPoemTitlenum, 301, 302
- \afterPoemTitleskip (length), 302
- afterword, 6
- \afterXtitle, 195
- pagenote, 359
- Alcuin, 35
- Aldus,
  - Linotype, 29
- \aliaspagestyle, 167
- alltt (environment), 293, 294
- alltt (package), 293, 395
- Alph, 163
- alph, 163
- alphabet length, 75
- alphabetic numbering, 163
- \alsoname, 347, 378
- altverse (environment), 297, 299, 303
- \amname, 365, 378
- amsfonts (package), 425
- \and, 113
- \andnext, 113
- \anonsubappendices, 124
- anonymous division, 157
  - styling, 157–159
- ans (file), 329
- answer (environment), 329
- answer (package), 331
- \anyptfilebase, 65, 432
- \anyptsize, 65, 432
- appendices (environment), 124
- appendix, 6, 47, 123, 124, 127, 134, 193, 201
  - subappendix, 124
- \appendix, 123, 124
- appendix (package), xix, 380
- \appendixname, 123, 124, 359, 378
- \appendixpage, 123, 127, 378
- \appendixpage\*, 123
- \appendixpagename, 123, 378
- \appendixrefname, 334
- \appendixtocname, 124, 378
- arabic, 163
- \Aref, 334
- argument,
  - optional, 331
- Argument of ... has an extra }, 416
- Argument to \overridesidecapmargin
  - neither ..., 430
- Argument to \setsidecappos is not ..., 430
- Arithmetic overflow, 416
- array, 417
  - intercolumn space, 267
  - row fill, 261–263
  - row spacing, 267

array (environment), 249, 251, 254, 255,  
     262, 265, 267, 269, 417, 419, 420,  
     423, 425, 426, 430, 431  
 array (package), xvii, xix, 249, 380  
 \arraybackslash, 265  
 \arraycolsep (length), 267  
 \arrayrulewidth (length), 267  
 \arraystretch, 267  
 \arraytostring, 375  
 Arrighi, Ludovico degli, 24  
 Arseneau, Donald, 105, 278, 283, 310, 311,  
     379  
 article (chapterstyle), 132  
 article (class), xv, 66, 117, 132, 192  
 article (option), 66, 122, 123, 128, 132, 151  
 \AtBeginClass, 381  
 \AtBeginDocument, 380  
 \AtBeginFile, 380  
 \AtBeginPackage, 380  
 patchcmd, 430  
 \AtEndClass, 381  
 \AtEndDocument, 380  
 \AtEndFile, 380  
 \AtEndOfClass, 380, 381  
 \AtEndOfPackage, 380–382  
 \AtEndPackage, 380–382  
 \atendtheglossaryhook, 353, 354  
 \author, 107, 111, 113, 428, 433  
 \autocols, 271  
 \autorows, 270, 271  
 aux (file), 321, 333, 401, 402  
 Avantgard, 76

## B

b (position argument), 214, 220, 224  
 b3paper (option), 63  
 b4paper (option), 63  
 b5paper (option), 63  
 b6paper (option), 63  
 babel (package), xv  
 back matter, 3, 6, 22, 40, 43, 44, 47, 48, 121,  
     337  
 \backmatter, 121–123  
 Bad \line or \vector argument, 424  
 Bad \sidebarmargin argument, 432  
 Bad math environment delimiter, 424

Barbon,  
     Monotype, 23  
 \baselineskip, 102  
 \baselineskip (length), 80, 82, 84, 102,  
     103, 178, 245, 289, 376, 397, 398  
 Baskerville, 75  
 bastard title, *see also* half-title  
 bastard title page, 3  
 \beforebookskip, 125  
 \beforechapskip (length), 128, 129,  
     150  
 \beforeepigraphskip (length), 289  
 \beforepartskip, 125, 292  
 \beforePoemTitleskip (length), 302  
 \begin, 204, 319, 424, 425  
 \begintheglossaryhook, 353, 354  
 \begin{...} ended by \end{...}, 424  
 \belowcaptionskip (length), 245  
 \belowrulesep (length), 259–261  
 \bfseries, 94, 126, 127, 129, 153, 182  
 bianchi (chapterstyle), 136  
 Bianchi, Stefano, 136  
 bib (file), 339, 340  
 \bibindent, 66  
 \bibintoc, 337  
 \bibitem, 337, 339, 428, 433  
 bibitemlist (environment), 337–339  
 \bibitemsep (length), 339  
 \biblabel, 429  
 bibliographic database, 339  
 bibliography, 4, 6, 48, 66, 201, 210, 292,  
     337–340  
     explanatory text, 338  
     flushleft entries, 338  
     heading, 338  
     label styling, 339  
     list styling, 338–339  
     title in ToC, 337  
 \bibliography, 339  
 \bibliographystyle, 339  
 \biblistextra, 338  
 \bibmark, 338  
 \bibname, 337, 378  
 \bibsection, 337, 338  
 BibTeX database, 340  
 BibTeX (program), xv, 339, 340  
 BibTeX style, 339, 340

abbrev, 340  
 alpha, 340  
 changing, 340  
 plain, 340  
 unsrt, 340  
 \bicaption, 235  
 \bicontcaption, 236  
 big point, xxiv  
 bilingual captions, 234–236  
 \bionenumcaption, 234  
 bitmap, 93  
     font, 93, 101  
 \bitwonumcaption, 234  
 blank space, 297  
 body font, 93, 97  
 boek (class), 142, 162  
 \boldmath, 428  
 book, 5, 121  
     number, 6, 122, 124  
 \book, 121, 122, 124, 125, 127, 128, 165,  
     197, 199, 292, 378, 382  
 book (class), xv, 117, 130, 134, 159, 162,  
     169, 192  
 book (pagestyle), 125, 165  
 \book\*, 382  
 Bookman, 75, 76  
 \bookname, 126, 378  
 \booknamefont, 126, 160  
 \booknamenum, 126  
 \booknumberline, 199  
 \booknumfont, 126, 160  
 \bookpagemark, 127  
 \bookrefname, 334, 378  
 booktabs (package), xvii, xix, 249, 259,  
     380  
 \booktitlefont, 126, 160  
 \bottomfraction, 223–225  
 bottomnumber (counter), 223  
 \bottomrule, 259, 260  
 \bottomsectionskip (length), 123  
 box, 213, 268, 309  
     framed, 310–313  
     include pagebreak, 310  
     shaded background, 310  
 \box, 416  
 \box255, 420  
 boxed verbatim, 317

boxedverbatim (environment), 317–  
     319, 323, 326  
 boxedverbatim\* (environment), 317  
 \boxedverbatiminput, 323  
 \boxedverbatiminput\*, 323  
 bp, xxiv  
 \Bref, 334, 378  
 bringhurst (chapterstyle), 136, 162  
 bringhurst (headstyles), 161, 162  
 bringhurst (package), 391, 392  
 Bringhurst, Robert, 20, 32  
 \bringpicl, 390  
 \bringpicr, 390  
 broadsheetpaper (option), 64  
 broadside, 6, 7  
 brotherton (chapterstyle), 136  
 \bs, 383  
 bst (file), 340  
 buffer size, 423  
 \bvbox, 317, 318  
 \bvboxsep (length), 317  
 \bvendofpage, 317  
 \bvendrulehook, 317  
 \bvleftsidehook, 317  
 \bvnumberinside, 319  
 \bvnumbersinside, 319  
 \bvnumbersoutside, 319  
 \bvperpagefalse, 317  
 \bvperpagetrue, 317  
 \bvrightsidehook, 317  
 \bvsides, 317, 318  
 \bvtopandtail, 317, 318  
 \bvtopmidhook, 317  
 \bvtopofpage, 317  
 \bvtoprulhook, 317  
 by (keyword), 407

## C

c (position argument), 269, 270  
 calc (package), 84, 86, 284  
 \calccentering, 113, 181, 182  
 Cambridge University Press, 23  
     Christmas Book, 23, 27, 29  
     Crutchley, Brooke, 23  
     Morison, Stanley, 23  
 Can only be used in the preamble, 425

- \cancelthanksrule, 116
- Cannot change a macro that has delimited arguments, 430
- caption, 16, 28, 57, 121, 202, 213, 225–247
  - LaTeX methods, 243–246
  - anonymous, 225, 230–232
  - bilingual, 234–236
    - in list of, 234
    - styling, 236
  - continuation, 225
  - continued, 229–230
    - outside a float, 233
  - delimiter, 226, 246
  - fixed, 217, 232–234, 269
  - font, 226, 228, 245, 246
  - footnote, 246–247
  - multiline, 226, 229
  - multiple, 216
  - new subcaption, 220
  - on opposite page, 233
  - outside a float, 232–234
  - paragraph style, 226
  - ruled, 227
  - short style, 226
  - side, 240–243
  - side caption, 227
  - space above, 245
  - style, 213, 225–229
  - subcaption, 236–240
    - continued, 238
    - in list of, 237
    - referencing, 238
    - styling, 239
  - width, 227
- \caption, 190, 197, 202, 211, 213, 229, 230, 232–234, 236–238, 243, 244, 246, 269, 333, 335, 354, 378, 383, 391, 424, 425
- \caption outside float, 425
- \captiondelim, 226
- \captionnamefont, 226, 242
- \captionstyle, 226
- \captiontitlefinal, 227
- \captiontitlefont, 226
- \captionwidth, 227
- cardinal,
  - number, 58
- \cardinal, 370, 371
- \cases, 256
- casting off, 9
- cc, xxiv
- ccaption (package), xvii, xix, 225, 380
- \cdot, 251
- Centaur, 20, 22, 29
- center (environment), 118, 179, 181, 186, 187, 245, 397
- centering,
  - vertical, 379
- \centering, 178–180, 226, 228, 239, 243, 245, 265, 302, 397
- \centerlastline, 226, 229, 239
- centimetre, xxiv
- Century Old Style, 108
- \cftaddnumtitleline, 202
- \cftaddtitleline, 202
- \cftbeforeKskip (length), 198
- \cftbookbreak, 197, 198
- \cftchapterbreak, 197, 198
- \cftdot, 196
- \cftdotsep, 196, 199
- \cftinsertcode, 202, 203
- \cftinserthook, 202, 203
- \cftKafterpnum, 199, 200, 203
- \cftKaftersnum, 198–200, 204
- \cftKaftersnumb, 198–200, 204
- \cftKdotsep, 199
- \cftKfont, 198, 200, 203
- \cftKformatpnum, 199
- \cftKindent (length), 198
- \cftKleader, 199–201
- \cftKname, 198
- \cftKnumwidth (length), 198, 199
- \cftKpagefont, 199
- \cftKpresnum, 198, 199, 204
- \cftlocalchange, 202
- \cftnodots, 196, 199
- \cftpagenumbersoff, 201, 210
- \cftpagenumberon, 201
- \cftparfillskip, 201
- \cftparskip (length), 197
- \cftpartbreak, 197, 198
- \cftsetindents, 199, 209
- \changecaptionwidth, 227
- \changed, 362

- \changeGLOSSactual, 352
- \changeGLOSSnum, 352, 353
- \changeGLOSSnumformat, 352
- \changeGLOSSref, 352
- \changemarks, 361
- changePAGE (package), 368
- \chapindent (length), 135
- \chapnamefont, 129, 160
- \chapnumfont, 129, 160
- chappell* (chapterstyle), 136, 145
- chapter, 4–6, 9, 18, 38, 40, 41, 66, 121, 124, 191
  - design, 20–24, 28, 29
  - number, 6, 40, 122, 123
  - precis, 150–151, 211–212
- \chapter, 66, 121–125, 128, 130, 132, 151, 165, 166, 172, 192, 194, 197–199, 208, 212, 289, 290, 292, 301, 333, 378, 382
- chapter (counter), 327
- chapter* (pagestyle), 123, 128, 165, 196
- \chapter\*, 122, 130, 290, 382
- chapterbib (package), 338
- \chapterheadstart, 128, 129
- \chaptermark, 166, 171
- \chaptername, 123, 359, 378
- \chapternamenum, 129
- \chapternumberline, 199
- \chapterprecis, 150, 151, 211, 212, 361
- \chapterprecishere, 151, 211, 212
- \chapterprecistoc, 151, 203, 211, 212
- \chapterrefname, 334, 378
- chapterstyle, xxi, 130–150
  - article*, 132
  - bianchi*, 136
  - bringhurst*, 136, 162
  - brotherton*, 136
  - chappell*, 136, 145
  - companion*, 132, 135, 396
  - crosshead*, 136, 162
  - culver*, 138
  - dash*, 138
  - default*, 130, 134–136, 290
  - demo*, 147, 148
  - demo2*, 139, 147, 148
  - demo3*, 139, 148, 161, 162
  - dowding*, 139, 162
  - ell*, 139
  - fred*, 130
  - ger*, 139
  - hangnum*, 130, 135
  - komalike*, 139, 162
  - lyhne*, 142
  - madsen*, 142
  - ntglike*, 142, 162
  - pedersen*, 121, 142, 148
  - pederson*, 161
  - reparticle*, 132
  - section*, 130, 132, 134, 135
  - southall*, 144, 149
  - tandh*, 144, 162
  - thatcher*, 144
  - veelo*, 144, 149, 150
  - verville*, 144
  - wilsondob*, 144, 162
- \chapterstyle, 130
- \chaptitelfont, 129, 160
- characters
  - list, 49
- Characters dropped after \end{...}, 432
- Charlemagne, 35
- Charter, 76
- \checkandfixthelayout, 82, 84, 88, 92, 105, 240, 284, 386
- \checkarrayindex, 375
- \checkifinteger, 375
- \checkoddpAGE, 369
- \checkthelayout, 82, 83, 431
- chngcntr (package), xix, 365, 380
- chnGPAGE (package), xix, 180, 368
- cicero, xxiv
- CIP, 5
- \circle, 429
- Citation ... on page ..., 428
- Citation ... undefined, 428
- \cite, 337, 339, 428, 429
- cite bibliographic item, 337
- \citeindexfile, 349
- \citeindextrue, 349
- class, xv, xxi
  - article, xv, 66, 117, 132, 192
  - boek, 142, 162

- book, xv, 117, 130, 134, 159, 162, 169, 192
- iso, 361
- memoir, xv, xxi, 63, 159, 192, 194, 209, 295
- report, xv, 111, 117, 134, 159, 192
- scrbook, 139, 162
- class option, 97
- class options, 63
  - article, 66
  - bibliography, 66
  - fonts, 67
  - math, 66
  - printing, 66
  - stock size, 63
  - type size, 64
- \cleaders, 158
- \cleardoublepage, 164, 165, 220, 233, 291, 369
- cleared (pagestyle), 165
- \clearforchapter, 128
- \clearpage, 128, 220, 233, 357, 369, 423, 424
- \cleartoevenpage, 233, 291, 369, 370
- \cleartooddpage, 370
- \cleartorecto, 128, 165, 370
- \cleartoverso, 128, 165, 370
- \cline, 267
- clo (file), 432
- \closeinputstream, 323
- \closeoutputstream, 322
- \club, 159
- cm, xxiv
- \cmd, 383
- \cmdprint, 383
- \cmidrule, 260, 261
- \cmidrulekern (length), 260, 261
- \cmidrulesep (length), 260
- \cmidrulewidth (length), 260, 261
- colophon, 6, 181
- color, 42, 51
  - blind, 42
- color (package), 142, 310, 395
- \colorchapnum, 148
- \colorchaptitle, 148
- column, 18, 21, 32
  - double, 27, 29, 32, 66, 80, 117, 119, 173, 395
  - multiple, 14, 31, 32, 56, 58, 220
  - narrow, 37, 56–57
  - single, 32, 66, 117, 119
- Column ... is already defined, 433
- \columnsep (length), 80, 83
- \columnseprule (length), 80, 83
- Command ... invalid ..., 425, 428
- Command ... not provided ..., 425
- Command \. . . already defined ..., 425
- comment (environment), 314, 315
- comment out text, 314
- comment (package), 315
- \commentsoff, 314
- \commentson, 314
- companion (chapterstyle), 132, 135, 396
- companion (pagestyle), 165, 171–174, 396
- Computer Modern, 65, 75, 76
- Computer Modern Roman, 24, 75
- Computer Modern Sans, 75
- Computer Modern Typewriter, 75
- Computer Sans, 76
- concordance, 342
- Concrete Roman, 75, 76
- conditional, 410
- configuration file, 85
  - MakeIndex, 344
- Connes, Frederic, 5, 344
- \contcaption, 229
- contents
  - list, 44
- \contentsline, 190, 192, 202
- \contentsname, 194, 195, 378
- \continuousmarks, 114
- \continuousnotenums, 357
- contributor, 6
- control sequence, 400
- \contsubbottom, 238
- \contsubcaption, 238
- \contsubtop, 238
- \contsupptop, 238
- \copy, 416
- copyfitting, 32
- \copypagestyle, 167
- copyright, 4, 5, 41
  - page, 43, 44

copyright page, 3–5  
 count, 406  
 \count, 426  
 counter, 213, 405  
     bottomnumber, 223  
     chapter, 327  
     dbltopnumber, 223  
     footnote, 114, 273  
     increment, 329  
     lastpage, 364  
     lastsheet, 364  
     maxsecnumdepth, 122  
     page, 163  
     pagenote, 357, 358  
     poemline, 300  
     pseudo, 327  
     question, 330  
     secnumdepth, 122, 124, 125  
     sheetsequence, 364  
     tocdepth, 189, 194, 205, 208, 214  
     topnumber, 223  
     totalnumber, 223  
     Xdepth, 214  
 Counter ... already defined, 433  
 counter representation, 163  
     Alph, 163  
     alph, 163  
     arabic, 163  
     Roman, 163  
     roman, 163  
 Counter too large, 425  
 \counterwithin, 366  
 \counterwithin\*, 366  
 \counterwithout, 366  
 \counterwithout\*, 366  
 Courier, 75, 76  
 \cplabel, 369  
 \cr, 417–419  
 \createmark, 170, 171, 432  
 \createplainmark, 170, 432  
 \Cref, 334, 378  
 crop (package), xx, 380  
 cross reference, 333  
     automatic, 333  
     specified, 333  
 crosshead (chapterstyle), 136, 162  
 crosshead (headstyles), 161, 162

crownvopaper (option), 64  
 Crutchley, Brooke, 23  
 \cs, 383  
 \csname, 404, 417, 419  
 \cstyle, 396  
 ctabular (environment), 269  
 CTAN, xx, xxi, 93  
 CTT, xx, xxi, 37, 149  
 culver (chapterstyle), 138  
 Culver, Christopher, 138

## D

dash, 53–54  
     2em, 53  
     3em, 54  
     em, 53  
     en, 53  
     hyphen, 53  
 dash (chapterstyle), 138  
 \date, 107, 111, 113, 433  
 datetime (package), 365  
 dbillpaper (option), 64  
 \dblfloatpagefraction, 223  
 \dblfloatsep (length), 223  
 \dbltextfloatsep (length), 223  
 \dbltopfraction, 223  
 dbltopnumber (counter), 223  
 dcolumn (package), xvii, xix, 249, 380  
 dd, xxiv  
 de Tourmes, Jean, 21  
 dedication, 44  
 \def, 324, 403, 404, 422  
 default,  
     page layout, 69  
     printing options, 66  
     stock, 63  
     type size, 65  
 default (chapterstyle), 130, 134–136, 290  
 default (headstyles), 159–162  
 \defaultaddspace, 260  
 \defaultaddspace (length), 260, 261  
 \defaultlists, 183  
 \defaultsecnum, 155  
 delarray (package), xvii, xix, 249, 380  
 \delcode, 420  
 \deleted, 362

\DeleteShortVerb, 315  
\delimeter, 420  
demo (chapterstyle), 147, 148  
demo2 (chapterstyle), 139, 147, 148  
demo3 (chapterstyle), 139, 148, 161, 162  
demyvopaper (option), 64  
description (environment), 182, 183, 185  
\descriptionlabel, 182  
\diamond, 158  
didot point, xxiv  
\dimen, 407, 409  
dimension, 407  
Dimension too large, 416  
\ding, 158  
\DisemulatePackage, 380  
Display math should end with \$\$, 416  
\divide, 407–409  
divide by zero, 416  
division,  
    anonymous, 157  
    sectional, *see also* subhead, 124  
\do, 414  
Do not use \footnote ..., 433  
document (environment), 67, 380, 401, 402  
\documentclass, 63, 65, 67, 76, 93, 425, 427  
double column,  
    index, 340, 341  
double spacing, 102  
Double subscript, 416  
Double superscript, 416  
\doublerulesep (length), 259, 260, 267  
\DoubleSpacing, 103  
dowding (chapterstyle), 139, 162  
dowding (headstyles), 161, 162  
Dowding, Geoffrey, 21  
\downbracefill, 261  
Downes, Michael, 105, 368  
\dp, 405  
draft document, 169  
draft (option), 66, 169, 329, 361  
*Dramatis Personae*, 49  
\drop, 398  
\dropchapter, 290  
\droptitle (length), 111, 112

dvi (file), 321, 400, 401  
dvips (program), 85  
Dye, Thomas, 144, 149

## E

headstyles, 159  
Ebook, *see* electronic books  
ebook (option), 63, 67  
\edef, 404  
seename, 346  
leftmargini (length), 296  
Ehrhardt, 21  
electronic books, 41–42  
ell (chapterstyle), 139  
ellipses, 53  
Els, Danie, 127, 198  
\else, 410, 414  
em, xxv  
em-dash, 53  
\emminershape, 96  
\emph, 95, 96  
emphasis, 36, 55, 57, 95  
Empty ‘thebibliography’ environment, 433  
empty (pagestyle), 112, 127, 164, 165, 167, 169, 196  
Empty preamble: ‘I’ used, 430  
\emptythanks, 114  
\EmulatedPackage, 379, 380  
\EmulatedPackageWithOptions, 379, 380  
en, xxv  
en-dash, 53  
\end, 204, 319, 417, 421, 424  
end floats, 325–328  
\end occurred inside a group ..., 417  
\end occurred when ..., 417  
\endcsname, 404, 417, 419  
\endgroup, 417, 419  
\endinput, 391  
\endlist, 185  
\endMakeFramed, 310  
endnote, 46, 330  
    font size, 47  
    mark, *see also* reference mark, 46, 47  
\endnote, 355

endnotes, 48, 323–325  
 endnotes (package), 355  
`\endwriteverbatim`, 328  
`\enlargethispage`, 104, 426  
`\enlargthispage`, 426  
 ent (file), 324, 325, 356–359  
 enumerate (environment), 178, 182, 183, 333, 433  
 enumerate (package), xix, 182, 380  
 environment,  
     abstract, xiii, 117–119, 378  
     adjustwidth, 180, 181, 183, 313  
     adjustwidth\*, 113, 180  
     aduswidth, 135  
     alltt, 293, 294  
     altverse, 297, 299, 303  
     answer, 329  
     appendices, 124  
     array, 249, 251, 254, 255, 262, 265, 267, 269, 417, 419, 420, 423, 425, 426, 430, 431  
     bibitemlist, 337–339  
     boxedverbatim, 317–319, 323, 326  
     boxedverbatim\*, 317  
     center, 118, 179, 181, 186, 187, 245, 397  
     comment, 314, 315  
     ctabular, 269  
     description, 182, 183, 185  
     document, 67, 380, 401, 402  
     enumerate, 178, 182, 183, 333, 433  
     epigraph, xvi  
     epigraphs, 288–290  
     eqnarray, 427  
     equation, 333  
     fboxverbatim, 317, 328  
     figure, 213, 231, 233, 244, 326, 327, 425, 426  
     flushleft, 118, 179, 187  
     flushright, 118, 179, 187  
     framed, 214, 310–313  
     framedminipage, 309, 310  
     framewithtitle, 312, 313  
     hangparas, 178  
     itemize, 182, 183, 283, 365, 426  
     lcode, 320, 397  
     leftbar, 311

list, 118, 182, 183, 186, 296, 397  
 lrbox, 309  
 MakeFramed, 310  
 midsloppypar, 104  
 minipage, 103, 217, 232, 237, 242, 243, 273, 309, 390  
 multicol, 204  
 new, 328, 329  
 onecolabstract, 119  
 patverse, 297, 299, 300, 305, 374, 430  
 patverse\*, 300, 430  
 picture, 402, 423, 424, 429  
 qfame, 313  
 qframe, 313  
 qshade, 313  
 quotation, 117, 180, 182, 183, 313, 427  
 quote, 151, 180, 182, 183, 212  
 shaded, 310, 311, 313  
 sidecaption, 240, 241, 243  
 sidecontcaption, 241  
 sidelegend, 241  
 sidenamedlegend, 241  
 sloppypar, 104  
 snugshade, 310, 311  
 snugshaded, 310  
 subappendices, 124  
 symbols, 185  
 syntax, 397  
 tabbing, 424, 426  
 table, 213, 231, 232, 269, 425  
 tabular, 216, 243, 249, 251, 254, 263–265, 267, 269, 273, 397, 417, 419, 420, 423, 425, 426, 430, 431  
 tabular\*, 249, 263, 264, 267  
 tabularx, 249, 263–266, 434  
 thebibliography, 165, 337, 338, 353, 426, 433  
 theglossary, 350, 351, 353  
 theindex, 165, 341, 343, 353  
 tightcenter, 397  
 titledframe, 313  
 titlepage, 112  
 titlingpage, 112, 113, 116, 164, 165  
 trivlist, 186, 187, 320, 397

- verbatim, 187, 294, 315–317, 319, 320, 323, 397, 423, 432
  - verbatim\*, 315–317
  - verbatimoutput, 322
  - verse, xviii, 293–296, 298–300
  - vminipage, 103
  - vplace, 379
  - writefigure, 327
  - writeverbatim, 322, 323, 327
  - Environment ... already defined, 433
  - Environment ... undefined, 425
  - epigraph, 3, 24, 128, 287–292
  - \epigraph, 287–290
  - epigraph (environment), xvi
  - epigraph (package), xix, 287, 380
  - epigraph* (pagestyle), 165, 292
  - \epigraphfontsize, 289
  - \epigraphforheader, 290
  - \epigraphhead, 165, 289, 290
  - \epigraphpicture, 290
  - \epigraphposition, 288
  - \epigraphrule (length), 289
  - epigraphs (environment), 288–290
  - \epigraphsourceposition, 288, 289
  - \epigraphtextposition, 288
  - \epigraphwidth (length), 288, 290
  - epilogue, 6
  - eqnarray (environment), 427
  - equation (environment), 333
  - error, 415
    - LaTeX, 424–427
    - memoir class, 429–432
    - TeX, 415–424
  - error message,
    - response, 415
      - continue, 415
      - edit, 415
      - exit, 415
      - help, 415
      - insert, 415
      - quiet, 415
      - run, 415
      - scroll, 415
  - descriptionlabel, 182
  - etex, 376
  - Euclid, 16
  - \evensidemargin (length), 83
  - \everydisplay, 103
  - ex, xxv
  - exception dictionary, 423
  - executivepaper (option), 64
  - \ext@type, 244
  - extend a macro, 367
  - Extra ..., 417
  - Extra ... or forgotten ..., 417
  - Extra alignment tab ..., 417
  - Extra \else, 417
  - Extra \endcsname, 417
  - Extra \fi, 417
  - Extra \or, 417
  - Extra \right, 417
  - \extracolsep, 254
  - extract, 45
  - extrafont sizes (option), 65, 431
  - \extrarowheight (length), 267
  - \extratabsurround, 268
  - \extratabsurround (length), 268
- F**
- fancybox (package), 311
  - \fancybreak, 157, 158
  - \fancybreak\*, 157
  - fancyhdr (package), xv, xvi, xx, 164, 380
  - \fbox, 309, 310, 317
  - \fboxrule (length), 331
  - \fboxsep (length), 331
  - fboxverbatim (environment), 317, 328
  - \fcardinal, 370, 371
  - \feetabovelfloat, 274
  - \feetbelowfloat, 274
  - \fi, 410, 413, 414
  - Fibonacci series, 17–18
  - \fibseries, 412, 413
  - \fibtogo, 413
  - figure, 3, 66, 174, 180, 192, 200–202, 209, 213, 214, 216, 225, 233, 234, 236, 237, 245, 247, 291, 385, 390, 391
    - float definition, 213
    - reference, 334
    - sub-, 209
    - subfigure, 236–240
  - figure (environment), 213, 231, 233, 244, 326, 327, 425, 426

- \figurename, 213, 246, 378
- \figurerefname, 334, 378
- figures,
  - hanging, 58
  - lining, 58
  - lowercase, 58
  - old-style, 58
  - ranging, 58
  - text, 58
  - titling, 58
- file, 321–330
  - ans, 329
  - aux, 321, 333, 401, 402
  - bib, 339, 340
  - bst, 340
  - clo, 432
  - close, 322, 323
  - dvi, 321, 400, 401
  - ent, 324, 325, 356–359
  - idx, 340–342, 344, 349
  - ind, 340, 341, 344, 349
  - ist, 344, 347
  - jobname.idx, 342
  - jobname.ind, 341
  - lof, 190
  - log, 321, 380, 401, 415
  - lot, 190
  - mem10.clo, 432
  - open, 322, 323
  - pdf, 321
  - read, 309, 321, 323–324, 328
    - single line, 323
    - verbatim, 323
  - source2e.tex, 403
  - tex, 321
  - toc, 151, 189, 190, 203, 212, 321, 356, 401, 402
  - write, 309, 321–324, 327, 329
    - verbatim, 322
- File ended while scanning ..., 417
- File not found ..., 425
- filll, 409
- fill, 409
- fil, 409, 410
- final (option), 66, 329, 361
- \finishwritefigure, 327, 328
- \firmlist, 183
- \firmlists, 183
- \firmlists\*, 183
- \firsthline, 268
- \fixdvipslayout, 85, 86
- fixltx2e (package), 173, 220, 395
- \fixpdflayout, 85
- \fixthelayout, 82, 84
- flafter (package), 221
- \flagverse, 300, 306
- \flegfigure, 232
- \flegfloat, 232
- \flegtable, 232
- \flegtocfigure, 232
- \flegtocfloat, 232
- \flegtocfloat, 232
- fleqn (option), 66
- fleuron, xvii, 22, 158
- float, 67, 121, 213, 225, 227, 229–233, 236, 238, 244, 247, 273, 370, 391, 395
  - bottom, 220, 223, 224
  - double column, 213, 220, 223, 225
  - flush, 220
  - framed, 214
  - here, 220, 223, 224
  - multiple, 216–220
    - table and figure, 217
  - new, 213–214
  - new diagram, 214
  - new subfloat, 220
  - page, 174–176, 220, 224
  - parameters, 221
  - placement, 220–225
  - position, 214
  - ruled, 215
  - set off, 214–215
  - single column, 213, 220
  - subfloat, 236–240
  - suppress, 220
  - suppress bottom, 220
  - suppress top, 220
  - top, 220, 221, 223
- Float too large ..., 428
- Float(s) lost, 425
- floatcomp (pagestyle), 174
- \floatpagefraction, 223–225
- \floatsep (length), 223, 225
- flush-and-hang, 47, 48

- \flushbottom, 67, 82, 83, 123, 369
- flushleft, 179
- flushleft (environment), 118, 179, 187
- flushright, 179
- flushright (environment), 118, 179, 187
- \fnsymbol, 279
- \fnum@figure, 245, 246
- \fnum@type, 244, 245
- \fnumbersep, 370, 371
- folio, xxiii, 3, 5–7, 18, 20–24, 29, 31, 40–42, 80, 121, 163, 164, 173, 210, 211, 385, 389
  - changing representation, 163
- Folio Society, 20, 21
- font, xxiii, xxv, 6, 13
  - Adobe Garamond, *see* Garamond
  - Avantgard, *see* Avantgard
  - Baskerville, *see* Baskerville
  - Bell Gothic, *see* Bell Gothic
  - bitmap, 93, 101
  - Bookman, *see* Bookman
  - Centaur, *see* Centaur
  - Century Old Style, *see* Century Old Style
  - change, 3, 9, 55–56
  - Charter, *see* Charter
  - Computer Modern, 93
  - Computer Modern Roman, *see* Computer Modern Roman
  - Computer Modern Sans, *see* Computer Modern Sans
  - Computer Modern Typewriter, *see* Computer Modern Typewriter
  - Concrete Roman, *see* Concrete Roman
  - Courier, *see* Courier
  - Ehrhardt, *see* Ehrhardt
  - family, xxiii
  - Galliard, *see* Galliard
  - Garamond Condensed, *see* Garamond Condensed
  - Helvetica, *see* Helvetica
  - Knuthian, 75
  - Linotype Aldus, *see* Aldus
  - measuring, 32
  - Metafont, 93
  - Minion, *see* Minion
  - Monophoto Garamond, *see* Garamond
  - monospaced, 75
  - Monotype Barbon, *see* Barbon
  - New Century, *see* New Century Schoolbook
  - New Century Schoolbook, *see* New Century Schoolbook
  - outline, 93, 97, 101
  - Palatino, *see* Palatino
  - PostScript, 75, 93
  - Sabon, *see* Sabon
  - sans, 35–38, 38
  - serified, 35–38, 38
  - Syntax, *see* Syntax
  - The Sans, *see* The Sans
  - Times, *see* Times
  - Times New Roman, *see* Times New Roman
  - Times New Roman Sans, *see* Times New Roman Sans
  - TrueType, 93
  - Typewriter, *see* Typewriter
  - Univers Condensed, *see* Univers Condensed
  - University of California Old Style, *see* University of California Old Style
  - Utopia, *see* Utopia
- Font ... not loadable ..., 417
- font characteristic,
  - family, 93, 94
  - series, 93, 94
  - shape, 93, 94
- Font command ... is not supported, 430
- font commands, 93
- font declarations, 93
- Font shape ..., 428
- font size, 97
- FontSite, 108
- foolscapvopaper (option), 64
- footer, 18, 20, 40–42, 69, 80–81, 164–166, 168, 169, 171–174, 361, 385
  - design, 20–24, 29, 31
- \footfootmark, 277, 278

\footfudgefiddle, 275  
 \footmarksep (length), 276, 278, 279  
 \footmarkstyle, 277, 278  
 \footmarkwidth (length), 276, 278, 279  
 footmisc (package), 274, 276  
 footnote, 18, 40, 46, 55, 97, 114–116, 159, 280  
     bottom float, 274  
     font size, 47  
     fragile, 247  
     in caption, 246–247, 416  
     in float, 247  
     in heading, 159, 416  
     mark, *see also* reference mark, 46, 55, 114, 159, 247  
     marker, 276, 277  
         multiple, 278  
         styling, 277  
     marker separator, 274  
     new series, 276  
     reference, 273  
     series, 276  
     style, 276  
     styles, 275  
     styling, 277–280  
     symbol, 279  
         order, 279  
     text, 247, 277, 278  
         font, 278  
     too long, 275  
     verbatim text, 274  
 \footnote, 159, 247, 273, 274, 276, 277, 324, 325, 356, 360, 433  
 footnote (counter), 114, 273  
 \footnoteA, 275  
 \footnoteB, 275, 276  
 \footnoteC, 275  
 \footnotemark, 115, 247, 273, 277  
 \footnotemarkB, 276  
 \footnoterule, 116, 276, 280  
 footnotes,  
     as a paragraph, 275  
     as paragraphs, 275  
     as three columns, 275  
     as two columns, 275  
 \footnotesize, 97, 98, 116, 278, 289  
 \footnotetext, 247, 273

\footnotetextB, 276  
 \footparindent (length), 278  
 \footref, 273  
 \footruleheight, 168  
 \footruleskip, 168  
 \footskip is too large ..., 431  
 \footskip (length), 80, 82, 83, 86, 386, 431  
 \foottextfont, 278  
 \foottopagenote, 359, 360  
 Forbidden control sequence found ..., 417  
 Ford, Matthew, 298  
 \fordinal, 371  
 fore-edge, 80  
 \foremargin (length), 82, 83, 282, 431  
 foreword, 4, 44  
 \frac, 373  
 fragile, 204, 331, 335, 416  
 fragile command, 159  
 frame,  
     box, 310–313  
         styling, 310  
     minipage, 309  
         verbatim, 309  
     narrow box, 311  
     rounded corners, 311  
     title, 311–313  
     verbatim, 317, 323  
         styling, 317  
 \FrameCommand, 310, 311  
 framed,  
     float, 214  
     verbatim, 328  
 framed (environment), 214, 310–313  
 framed (package), xix, 310, 313, 380  
 framedminipage (environment), 309, 310  
 \FrameHeightAdjust (length), 310  
 \FrameRestore, 310  
 \FrameRule (length), 310  
 \FrameSep (length), 310  
 framewithtitle (environment), 312, 313  
 fred (chapterstyle), 130  
 \freetabcaption, 269  
 \fref, 334, 378

front matter, 3, 4, 6, 22, 40, 42, 43, 48, 55, 121  
 frontispiece, 43, 202  
`\frontmatter`, 118, 121, 125, 130, 382  
`\frontmatter*`, 121  
 full stop, *see also* period  
`\fussy`, 104  
`\futurelet`, 368

## G

Galliard, 22  
 Garamond,  
   Adobe, 23  
   Monophoto, 24  
 Garcia, Gerardo, 139  
 gathering, 6, 364  
`\gdef`, 404  
 geometry (package), xvi, xx, 69, 380  
`ger` (chapterstyle), 139  
`\getarrayelement`, 375  
`\gfibseries`, 412, 413  
 ghostview (program), 85  
`\global`, 404  
 glossary, 6, 48  
`\glossary`, 350–352, 355, 423  
`\glossarycolsep` (length), 353  
`\glossaryentry`, 352  
`\glossaryintoc`, 353  
`\glossarymark`, 353  
`\glossaryname`, 353, 378  
`\glossaryrule` (length), 353  
`\glossitem`, 351, 353, 354  
 glue, 407  
 golden section, 16–18, 27, 86  
 Goudy, Frederick, 27  
`\gparindent` (length), 320, 321, 397  
 graphicx (package), 142, 144, 150, 395  
 group, 403  
 gsview32 (program), 85  
 Gutenberg, Johannes, 26, 31  
 gutter, 80  
   width, 80

## H

h (position argument), 220, 224

h float specifier ..., 428  
 half-title, 20–22, 24, 31  
   page, 44  
 half-title page, 3, 43, 107  
`\halign`, 417, 419  
`\hangcaption`, 226, 227  
`\hangfrom`, 178  
 hanging,  
   figures, 58  
 hanging (package), 178  
`hangnum` (chapterstyle), 130, 135  
`\hangpara`, 178  
`hangparas` (environment), 178  
`\hangsecnum`, 155  
`\hangsubcaption`, 239, 240  
 chapter, 125  
 hash size, 423  
`\hb@xt@`, 405  
`\hbox`, 158, 405, 416, 420, 422  
`\headdrop` (length), 81  
 header, 18, 20, 40–42, 69, 80–81, 86, 122, 159, 164–166, 168, 169, 171–174, 290, 361, 385, 389  
   design, 20, 22–24  
   specifying size, 80–82  
`\headheight` and/or `\headsep` are too large ..., 431  
`\headheight` (length), 80, 82, 83, 86, 431  
 heading, 3, 13, 32, 34, 39, 41, 67, 194, 221, 288  
   abstract, 117  
   book, 6, 125–127  
   chapter, 66, 127–130, 150, 201, 211, 289, 291  
   part, 6, 66, 125–127  
   sections, 66, 152–156  
 headings (pagestyle), 69, 164, 165, 169  
`\headmargin` (length), 82, 83, 86  
`\headnameref`, 335  
`\headsep` (length), 80, 81, 83, 86, 389, 431  
`\headskip`, 82  
 headstyles,  
   **brighthurst**, 161, 162  
   **crosshead**, 161, 162  
   **default**, 159–162  
   **dowding**, 161, 162  
   **komalike**, 161, 162

*memman*, 160–162  
*ntglke*, 161, 162  
*tandh*, 161, 162  
*wilsondob*, 161, 162  
\headstyles, 159  
\headwidth (length), 168  
\heavyrulewidth (length), 259–261  
Helvetica, 75, 76  
\hfil, 158, 199, 201, 410  
\hfill, 201, 216, 229, 410  
\hfilneg, 410  
\hhrrule, 217  
\hideindexmarks, 343  
\hline, 267  
\hmpunct, 365  
horizontal mode, 405  
\hrule, 217, 318  
\hrulefill, 261  
\hsize (length), 310  
\hskip, 410  
\hss, 410  
\ht, 405  
\HUGE, 98, 101  
\Huge, 97, 98, 126, 127, 129  
\huge, 97, 98, 126, 129  
Huge page cannot be shipped out, 417  
Hurenkinder, *see* widow, 38  
hyperindex, 344  
hyperref (package), xv, xix, 194, 238, 334, 343, 344, 381, 382  
hyphen, 53  
hyphenation, 56, 57, 283  
\hyphenation, 418, 420, 423  
Høgholm, Morten, 32, 75

## I

I can't find file ..., 417  
I can't go on meeting you like this, 417  
I can't write on file ..., 418  
sidecapfloatwidth, 242  
\idtextinnotes, 358, 359  
idx (file), 340–342, 344, 349  
\idxmark, 173  
if-then-else, 410  
\ifbounderror, 375  
\ifcase, 414

\ifdim, 411  
\ifdraft, 329  
\ifdraftdoc, 169, 361  
\ifetex, 376  
\ifinteger, 375  
ifmtarg (package), xix, 380  
\ifnum, 411  
\ifodd, 411  
\ifoddpage, 369  
\ifonlyfloats, 174  
\ifpdf, 375, 376  
ifpdf (package), xix, 380  
\ifsamenam, 368  
\ifscapmargleft, 240, 241  
\ifsidebaroneside, 283  
\IfStreamOpen, 322, 329  
\ifxetex, 376  
\ignorenoidxfile, 341  
\iiirdstring, 371  
\iindstring, 371  
Illegal character ..., 425  
Illegal parameter number ..., 418  
Illegal pream-token ..., 431  
Illegal unit of measure ..., 418  
illustration, 3, 9, 14, 16, 20, 24, 26, 28, 29, 32, 34, 42, 57, 128, 163, 202, 210, 216, 217, 232, 245  
list, 44  
multiple, 216  
imperialvopaper (option), 64  
Improper \hyphenation ..., 418  
in, xxiv  
inch, xxiv  
\include, 343, 380, 425, 426  
\includegraphics, 216  
\includeonly, 343  
Incomplete ..., 418  
incunabula, 26  
ind (file), 340, 341, 344, 349  
\indentcaption, 226, 227  
vindent (length), 296  
\indentpattern, 299, 300, 305, 307  
index, 4, 6, 48, 173, 201, 210, 292, 295, 396  
double column, 340, 341  
gutter, 341  
entry levels, 343  
explanatory text, 341

- main entry, 343
  - multiple, 295, 340, 342
  - multiple column, 56
  - name, 341
  - preparation, 342
  - print, 341
  - printing, 340–341
  - raggedright, 57
  - see also reference, 342, 346
  - see reference, 342, 346
  - show indexed items, 343
  - single column, 340, 341
  - subentry, 343
    - subsubentry, 343
  - title in ToC, 341
  - \index, 342–344, 346, 423
  - Index ... for pattern ..., 433
  - Index ... outside limits ..., 431
  - index (package), xix, 295, 380
  - index (pagestyle), 173
  - \indexcolsep (length), 341
  - indexing, 173
  - \indexintoc, 341
  - \indexmark, 341
  - \indexmarkstyle, 343
  - \indexname, 341, 378
  - \indexrule (length), 341
  - Infinite glue shrinkage ..., 418
  - \input, 323, 380, 417, 425
  - input stack size, 423
  - Input stream ... is already defined, 433
  - Input stream ... is not open, 433
  - Input stream ... is open, 433
  - \InputIfFileExists, 380, 381
  - insert in contents, 192
  - \insertchapterspace, 130, 208
  - interlinear space, *see* leading
  - \intextsep (length), 223
  - introduction, 44
  - ISBN, 5
  - iso (class), 361
  - \isopage, 88, 89
  - ist (file), 344, 347
  - \iststring, 371
  - italic, 35
  - \item, 182, 183, 288, 333, 339, 343, 425–427
  - itemize (environment), 182, 183, 283, 365, 426
  - \itemsep (length), 339
  - TitleFrame, 313
  - \itshape, 94, 151, 203, 212
- J**
- Jenson, Nicolas, 22
  - \jobname, 321, 324
  - jobname.idx (file), 342
  - jobname.ind (file), 341
  - jurabib (package), 340
- K**
- \keepthetitle, 114
  - keyword,
    - by, 407
    - minus, 407
    - plus, 407
  - \killtitle, 114
  - komalike* (chapterstyle), 139, 162
  - komalike* (headstyles), 161, 162
  - kyus, xxiv
- L**
- l (position argument), 269, 270
  - \l@book, 197
  - \l@kind, 190, 191
  - \l@section, 192
  - label, 333
  - \label, 211, 217, 233, 238, 273, 328–330, 333, 334, 406, 423, 428, 429
  - Label ... multiply defined, 428
  - Label(s) may have changed ..., 428
  - landscape, 45
    - orientation, 45
  - landscape (option), 63, 85
  - \LARGE, 97, 98
  - \Large, 97, 98, 388
  - \large, 97, 98, 154, 178, 302
  - largecrownvopaper (option), 64
  - largepostvopaper (option), 64
  - \lasthline, 268
  - lastpage (counter), 364

lastsheet (counter), 364	\arraycolsep, 267
LaTeX,	\arrayrulewidth, 267
error, 424–427	\baselineskip, 80, 82, 84, 102, 103, 178, 245, 289, 376, 397, 398
warning, 427–429	\beforechapskip, 128, 129, 150
latexsym (package), 425	\beforeepigraphskip, 289
layouts (package), 88, 89, 196, 395, 396	\beforePoemTitleskip, 302
\lazypagecheck, 180, 369	\belowcaptionskip, 245
\lccode, 420	\belowrulesep, 259–261
\lcminusname, 373, 378	\bibitemsep, 339
lcode (environment), 320, 397	\bottomsectionskip, 123
leading, xxiii, 34, 47, 102, 376	\bvboxsep, 317
\leadpagetoclevel, 127	\cftbeforeKskip, 198
leaf, xxiii, 7	\cftKindent, 198
League, Christopher, 108	\cftKnumwidth, 198, 199
\leavespergathering, 364	\cftparskip, 197
\leavevmode, 158, 405	\chapindent, 135
ledgerpaper (option), 64	\cmidrulekern, 260, 261
ledmac (package), 274, 276	\cmidrulesep, 260
\left, 255, 256, 417	\cmidrulewidth, 260, 261
leftbar (environment), 311	\columnsep, 80, 83
\leftmargini (length), 296	\columnseprule, 80, 83
\leftmark, 166, 169, 171–173	\dblfloatsep, 223
legalpaper (option), 64	\dbltextfloatsep, 223
legend, 57–58, 230	\defaultaddspace, 260, 261
in list of, 231	\doublerulesep, 259, 260, 267
named, 232	\droptitle, 111, 112
\legend, 202, 230, 232, 335, 336, 383, 391	leftmargini, 296
legibility, 35, 36, 51	\epigraphrule, 289
length, 407	\epigraphwidth, 288, 290
\@dblfpbot, 223, 224	\evensidemargin, 83
\@dblfpsep, 223	\extrarowheight, 267
\@dblfpstop, 223, 224	\extratabsurround, 268
\@dblftop, 224	\fboxrule, 331
\@fpbot, 223, 224	\fboxsep, 331
\@fpsep, 223, 224	\floatsep, 223, 225
\@fptop, 223, 224	\footmarksep, 276, 278, 279
\@pnumwidth, 199	\footmarkwidth, 276, 278, 279
\abovecaptionskip, 245	\footparindent, 278
\aboverulesep, 259–261	\footskip, 80, 82, 83, 86, 386, 431
\absleftindent, 118	\foremargin, 82, 83, 282, 431
\absparindent, 118, 119	\FrameHeightAdjust, 310
\absparsep, 118	\FrameRule, 310
\absrightindent, 118	\FrameSep, 310
\abstitlekip, 118, 119	\glossarycolsep, 353
\afterchapskip, 128, 129	\glossaryrule, 353
\afterepigraphskip, 289	\gparindent, 320, 321, 397
\afterPoemTitleskip, 302	

<code>\headdrop</code> , 81	<code>\sidecapwidth</code> , 240
<code>\headheight</code> , 80, 82, 83, 86, 431	<code>\sideparvshift</code> , 281
<code>\headmargin</code> , 82, 83, 86	<code>\spinemargin</code> , 82, 83, 181, 431
<code>\headsep</code> , 80, 81, 83, 86, 389, 431	<code>\stanzaskip</code> , 296
<code>\headwidth</code> , 168	<code>\stockheight</code> , 81, 83, 431
<code>\heavyrulewidth</code> , 259–261	<code>\stockwidth</code> , 81, 83, 386, 431
<code>\hsize</code> , 310	<code>\tabcolsep</code> , 267
<code>vindent</code> , 296	<code>\textfloatsep</code> , 223, 225
<code>\indexcolsep</code> , 341	<code>\textheight</code> , 76, 82–84, 284, 432
<code>\indexrule</code> , 341	<code>\textwidth</code> , 76, 78, 82, 83, 168, 181, 268, 431
<code>\intextsep</code> , 223	<code>\thanksmarksep</code> , 115
<code>\itemsep</code> , 339	<code>\thanksmarkwidth</code> , 115
<code>\leftmargini</code> , 296	<code>\topmargin</code> , 83
<code>\lightrulewidth</code> , 259–261	<code>\topsep</code> , 186
<code>\linewidth</code> , 192, 242	<code>\topskip</code> , 82, 84, 105
<code>\lowermargin</code> , 431, 432	<code>\trimedge</code> , 82, 83, 86, 386, 431
<code>\lxvchars</code> , 75, 76	<code>\trimtop</code> , 82, 83, 86, 431
<code>\marginparpush</code> , 81–83	<code>\unitlength</code> , 182, 207, 290, 389
<code>\marginparsep</code> , 81–83, 240, 281, 389, 390	<code>\uppermargin</code> , 82, 83, 431, 432
<code>\marginparwidth</code> , 81–83, 240, 281, 389, 390	<code>\verbatimindent</code> , 316
<code>\midchapskip</code> , 128, 129, 150	<code>\versewidth</code> , 295, 296
<code>\midPoemTitleskip</code> , 302	<code>\vgap</code> , 296, 297, 299
<code>\normalrulethickness</code> , 168	<code>\vindent</code> , 296
<code>\oddsidemargin</code> , 83	<code>\vleftmargin</code> , 296
<code>\onelineskip</code> , 102, 376, 398	<code>\vleftskip</code> , 300
<code>\paperheight</code> , 71, 81, 83, 431, 432	<code>\vrightskip</code> , 300
<code>\paperwidth</code> , 71, 81, 83, 181, 386, 431	<code>\width</code> , 310
<code>\parindent</code> , 76, 102, 177, 179, 250	<code>\xlvchars</code> , 75, 76
<code>\parsep</code> , 339	Length ... already defined, 433
<code>\parskip</code> , 101, 102, 177, 197, 245	<code>leqno</code> (option), 66
<code>\partopsep</code> , 186	<code>\let</code> , 204, 404
<code>\pfbreakskip</code> , 157	letterpaper, 69
<code>\prechapterprecisshift</code> , 151	letterpaper (option), 63, 64, 67, 86, 88
<code>\pwayii</code> , 389, 390	letterspacing, 57
<code>\pwayi</code> , 389, 390	lettrine (package), 398
<code>\ragrparindent</code> , 179	<code>\lightrulewidth</code> (length), 259–261
<code>rigid</code> , 407, 409	Limit controls ..., 418
<code>rubber</code> , 407, 409	<code>\limits</code> , 418
<code>\sidebarhsep</code> , 283, 284	Limits for array ... , 431
<code>\sidebartopsep</code> , 283, 284	line,
<code>\sidebarvsep</code> , 283, 284	orphan, 104, 105
<code>\sidebarwidth</code> , 283, 284	widow, 104, 105
<code>\sidecapraise</code> , 242	<code>\line</code> , 424, 429
<code>\sidecapsep</code> , 240, 242	line number, 296, 297, 306, 317, 326
	font, 300, 318
	frequency, 300, 318

position, 300, 319  
 reset, 318  
 line too long, *see* overfull lines  
 \linebreak, 422  
 \linenumberfont, 300, 318  
 \linenumberfrequency, 300, 318  
 \linewidth (length), 192, 242  
 lining,  
     figures, 58  
 Linotype Aldus, 29  
 list, 182–187  
     characters, 49  
     contents, 44  
     illustration, 44  
     new, 183–185  
     new list of, 207–211  
     spaces, 320  
     symbol, 48  
     tight, 183  
 \list, 185  
 list (environment), 118, 182, 183, 186,  
     296, 397  
 \listanswername, 207  
 \listfigurename, 194, 195, 378  
 \listofanswers, 207  
 \listoffigugres, 378  
 \listoffigures, 165, 189, 194, 197  
 \listoffigures\*, 189, 194  
 \listofplates, 210  
 \listoftables, 165, 189, 192, 194, 197,  
     378  
 \listoftables\*, 189, 194  
 \listplatename, 210  
 \listtablename, 194, 195, 378  
 \llap, 135  
 lmodern (package), 65  
 \LoadClass, 381, 425–427  
 LoF, 4, 189, 191, 193–195, 197, 198, 202,  
     208, 210, 385, 390, 423  
 lof (file), 190  
 \lofmark, 195  
 log (file), 321, 380, 401, 415  
 Lonely \item ..., 425  
 lonely line, 38  
 \long, 404  
 longtable (package), xvii, 268  
 \loop, 413, 414

\loosesubcaptions, 239  
 LoT, 4, 189, 191, 193–195, 197, 208, 210,  
     385, 390  
 lot (file), 190  
 \lotmark, 195  
 lowercase,  
     figures, 58  
 \lowermargin, 82  
 \lowermargin (length), 431, 432  
 lrbox (environment), 309  
 alsoname, 346  
 Lucas, Édouard, 411  
 Luecking, Daniel, 37, 199  
 \lxvchars (length), 75, 76  
 lyhne (chapterstyle), 142  
 Lyhne, Anders, 142

## M

\m@ne, 406  
 \m@th, 158  
 madsen (chapterstyle), 142  
 Madsen, Lars, 127, 136, 142, 148, 170, 202  
 main matter, 3, 5, 6, 40, 42, 44, 47, 121, 337  
 main memory size, 423  
 \mainmatter, 121–123, 125, 169  
 \mainmatter\*, 121, 125  
 majuscule, 35  
 \makeatletter, 193, 229, 245, 280, 391,  
     403, 422  
 \makeatother, 193, 229, 245, 280, 391,  
     403, 422  
 makebst (package), 340  
 \makechapterstyle, 132, 134  
 \makeevenfoot, 167, 168  
 \makeevenhead, 167, 168  
 \makefootrule, 168  
 \MakeFramed, 310, 311  
 MakeFramed (environment), 310  
 \makeglossary, 350, 353  
 \makeheadposition, 168, 368  
 \makeheadrule, 168  
 \makeheadstyles, 159, 160  
 makeidx (package), xix, 380  
 MakeIndex,  
 \makeindex, 340, 341

- MakeIndex (program), xv, xix, 344, 347,  
     349–352, 355, 396  
     configuration file, 344  
     output styling, 347–349  
     raw data, 344–347  
 \MakeLowercase, 388  
 \makeoddfoot, 167, 168  
 \makeoddhead, 167, 168  
 \makeother, 422  
 \makepagenote, 356  
 \makepagestyle, 167, 168  
 \makepsmarks, 168, 169, 172, 174  
 \makerunningwidth, 168  
 \MakeShortVerb, 315  
 \makethanksmark, 116  
 \makethanksmarkhook, 116  
 \maketitle, 107, 108, 111–114, 116, 119,  
     165, 275, 426, 428, 433  
 \maketitlehooka, 112  
 \maketitlehookb, 112  
 \maketitlehookc, 112  
 \maketitlehookd, 112  
 \MakeUppercase, 169, 170  
 Manutius, Aldus, 24  
 \marg, 383  
 margin, 18, 24, 26, 27, 29, 39, 40, 66, 69,  
     77, 78, 82, 118, 180–182, 193, 202,  
     362, 389  
     fore-edge  
         fore-edge, 386  
     fore-edge, *see also* outer, 71, 77, 78,  
         86, 165, 171, 385  
     fore-edge, 77  
     inner, *see also* spine, 24, 29, 77, 86, 171  
     left, 130, 152, 153, 155, 177, 179, 180,  
         191, 192, 198, 296  
     lower, 77, 79, 86  
     outer, *see also* fore-edge, 26, 40, 77,  
         80, 86, 171, 173, 180  
     right, 179, 180, 191, 193, 196, 201, 290  
     specify width, 77  
     specifying size, 77–80, 82  
     spine, *see also* inner, 24, 29, 71, 77, 78,  
         82, 86, 180  
     top, *see also* upper, 386  
     upper, *see also* top, 77, 79, 80, 82, 86,  
         385  
     margin note, 280  
         moved, 280, 281  
         specifying the margin, 280  
         text for particular margin, 280  
         wrong margin, 280  
 marginalia, 14, 18, 27, 69, 81–82, 86, 171,  
     231, 385, 386, 389  
 \marginbox, 156  
 \marginhead, 156  
 \marginpar, 273, 280, 281, 283, 425–428,  
     433  
 Marginpar on page ..., 428, 433  
 \marginparpush (length), 81–83  
 \marginparsep (length), 81–83, 240,  
     281, 389, 390  
 \marginparwidth (length), 81–83, 240,  
     281, 389, 390  
 \markboth, 165, 166  
 markers, 165, 172  
 \markright, 165, 166  
 math,  
     in caption or title, 416  
 math mode, 405  
 math unit, 191  
 \mathindent, 66  
 mathpazo (package), 76  
 \max, 366  
 \maxdeadcycles, 420  
 maximum length, 416  
 maximum number, 416  
 \maxsecnumdepth, 124, 125  
 maxsecnumdepth (counter), 122  
 \maxtocdepth, 194  
 McLean, Ruari, 24  
 mcrownvopaper (option), 63  
 mdemyvopaper (option), 63  
 \mdseries, 94  
 measure, xxiii  
     45 characters, 75  
     65 characters, 75  
     narrow, 56–57  
 \medievalpage, 88, 89  
 mediumvopaper (option), 64  
 \medspace, 377  
 mem10.clo (file), 432  
 \memappchapinfo, 382  
 \memappchapstarinfo, 382

- \memapppageinfo, 383
- \memapppagestarinfo, 383
- \membicaptioninfo, 383
- \membionenumcaptioninfo, 383
- \membitwonumcaptioninfo, 383
- \membookinfo, 382
- \membookstarinfo, 382
- \memcaptioninfo, 383
- \memchapinfo, 382
- \memchapstarinfo, 382
- \memdskip, 103
- \memdskipstretch, 103
- memexsupp (package), xvi
- \memfontenc, 65
- \memfontfamily, 65
- \memfontpack, 65
- \memglodesc, 352
- \memglonum, 352
- \memgloref, 352
- \memgloterm, 352
- \memgobble, 379
- memhfixc (package), xix, 381, 382
- \memjustarg, 379
- \memleadpageinfo, 383
- \memleadpagestarinfo, 383
- \memlegendinfo, 383
- memman* (headstyles), 160–162
- \memnamedlegendinfo, 383
- memoir (class), xv, xxi, 63, 159, 192, 194, 209, 295
  - error, 429–432
  - warning, 432–434
- \mempartinfo, 382
- \mempartstarinfo, 382
- \mempoeminfo, 383
- \mempoemstarinfo, 383
- \memPoemTitleinfo, 383
- \memPoemTitlestarinfo, 383
- \memsecinfo, 382
- \memsecstarinfo, 382
- memsty (package), 204, 396, 398
- \memUChead, 170
- \mergepagefloatstyle, 174, 175
- \meta, 383
- Metafont, 93
- \midbicaption, 236
- \midbookskip, 126
- \midchapskip (length), 128, 129, 150
- midpage (package), 233
- \midpartskip, 126
- \midPoemTitleskip (length), 302
- \midrule, 259, 260
- \midsloppy, 104
- midsloppypar (environment), 104
- millimetre, xxiv
- Minion, 20, 385
- minipage,
  - in float, 217
- minipage (environment), 103, 217, 232, 237, 242, 243, 273, 309, 390
- \miniscule, 98, 101
- minus (keyword), 407
- minuscule, 35
  - Carolingian, 27, 35
  - humanist, 26
- \minusname, 373, 378
- minus, 407, 409
- Misplaced &, 418
- Misplaced \cr, 418
- Misplaced \noalign, 419
- Misplaced \omit, 419
- Misplaced \span, 418
- Missing = inserted ..., 419
- Missing @-exp ..., 426
- Missing # inserted ..., 419
- Missing \$ inserted, 419
- Missing \cr inserted, 419
- Missing \endcsname inserted, 419
- Missing \endgroup inserted, 419
- Missing \right inserted, 419
- Missing { inserted, 419, 420
- Missing }inserted, 419
- Missing \begin{document}, 425
- Missing arg: token ignored, 431
- Missing control sequence inserted, 420
- Missing delimiter(. inserted)., 420
- Missing number ..., 420
- Missing p-arg ..., 426
- mixing fonts, 95
- \mkern, 159
- mlargecrownvopaper (option), 63
- mm, xxiv
- mode, 405
  - display math, 405

horizontal, 405  
 internal vertical, 405  
 math, 405  
 restricted horizontal, 405  
 vertical, 405  
 Monophoto Garamond, 24  
 monospaced font, 75  
 Monotype Barbon, 23  
 \morecmidrules, 260  
 moreverb (package), xix, 380  
 Morison, Stanley, 23  
 \movetoevenpage, 369, 370  
 \movetooddpage, 370  
 moving argument, 335, 416  
 ms (option), 66  
 msmallroyalvopaper (option), 63  
 \multfootsep, 274  
 multicol (package), 204, 220, 310  
 multicol (environment), 204  
 \multicolumn, 255, 266, 425, 426, 431  
 multind (package), 295  
 \multiply, 407–409  
 \multiput, 424  
 myheadings (pagestyle), 164–166

## N

\n@me@number, 373  
 name,  
   index, 341  
 named,  
   number, 370, 371  
 \namedlegend, 231–233  
 \namenumberand, 372, 373, 378  
 \namenumbercomma, 372, 378  
 nameref (package), 334  
 \nameref, 335  
 \nameref, 335, 336  
 \namesubappendices, 124  
 \nametest, 368  
 natbib (package), 338, 340, 349  
 \Needspace, 369  
 \needspace, 295, 306, 369  
 needspace (package), xix, 380  
 \Needspace\*, 369  
 new,  
   counter, 327

environment, 327  
 New Century, 76  
 New Century Schoolbook, 108  
 \newarray, 374, 431  
 \newblock, 339  
 \newcolumn, 251, 253, 254, 265, 433  
 \newcommand, 127, 254, 324, 366, 399, 403, 418  
 \newcommand\*, 367, 420  
 \newcomment, 314  
 \newcount, 406  
 \newcounter, 208, 365, 406, 426  
 \newdimen, 407  
 \newenvironment, 418, 424  
 newfile (package), xix, 380  
 \newfixedcaption, 233  
 \newfloat, 213, 214, 220  
 \newfootnoteseries, 276  
 \newfootseries, 276  
 \newif, 410  
 \newinputstream, 321, 433  
 \newleadpage, 127  
 \newleadpage\*, 127  
 \newlength, 407, 409  
 \newline, 123, 422, 427  
 \newlistentry, 208–210  
 \newlistof, 123, 207, 208  
 \newloglike, 366  
 \newloglike\*, 366  
 \newoutputstream, 321, 433  
 \newpage, 369  
 \newsavebox, 309  
 \newskip, 407, 409  
 \newsfloat, 219, 220  
 \newtheorem, 424, 426  
 nextpage (package), xix, 369, 380  
 \nextpageref, 211  
 No array called ..., 431  
 No \author given, 428  
 No counter ... defined, 426  
 No more to read from stream ..., 433  
 No positions in optional float specifier ..., 428  
 No room for a new ..., 426  
 No room for a new read, 321  
 No room for a new write, 321  
 No \title given, 426

\noalign, 419  
 \nobibintoc, 337  
 \nobvbox, 317, 318  
 \nochangemarks, 361  
 \nocite, 339  
 \noDisplayskipStretch, 103  
 \nofiles, 401  
 \noglossaryintoc, 353  
 \noindent, 288  
 \noindexintoc, 341  
 \nolimits, 418  
 \nonzeroparskip, 101, 102  
 \noprelistbreak, 365  
 \normalbottomsection, 123  
 \normalcaption, 226, 227  
 \normalcaptionwidth, 227  
 \normalfont, 76, 93  
 \normalmarginpar, 280  
 \normalrulethickness, 174  
 \normalrulethickness (length), 168  
 \normalsize, 97, 98, 178  
 \normalsubcaption, 239, 240  
 Not a letter, 420  
 Not defined: ..., 431  
 Not in outer par mode, 426  
 Not redefinable: ..., 431  
 \notedivision, 358, 359, 378  
 \noteentry, 358  
 \noteidinnotes, 358, 359  
 \noteinnotes, 359  
 \notenuminnotes, 358, 359  
 \notenumintext, 358  
 \notepageref, 357–359  
 \notesname, 358, 378  
 \nouppercaseheads, 165, 170  
*ntgl* (chapterstyle), 142, 162  
*ntgl* (headstyles), 161, 162  
 \nthstring, 371  
 number, 370  
     cardinal, 58  
     formatting, 58  
     named, 370, 371  
     numeric, 370  
     ordinal, 58  
     representation, 370  
 Number too big, 420  
 numbered lines, 317

\numberline, 191, 202–204  
 \NumberPoemTitle, 301  
 numeric,  
     number, 370  
 \NumToName, 371  
 \numtoName, 371  
 \numtoname, 371, 372  
 Nyman, Patrick, 178

## O

\oarg, 383  
 Oberdiek, Heiko, 382  
 octavo, 7, 9  
 \oddsidemargin (length), 83  
 old-style,  
     figures, 58  
 oldfontcommands (option), 67, 430  
 oldpaper (option), 64  
 colorchapnum, 148  
 colorchaptitle, 148  
 \omit, 419  
 onecolabstract (environment), 119  
 \onecolglossary, 353  
 \onecolindex, 341  
 onecolumn (option), 66, 213  
 \OnehalfSpacing, 103  
 \onelineskip (length), 102, 376, 398  
 oneside (option), 66, 67, 85, 128  
 Only one column-spec. allowed, 431  
 \openany, 128  
 openany (option), 66, 128  
 openbib (option), 66, 339  
 \openinputfile, 323  
 \openleft, 128  
 openleft (option), 66, 128  
 \openoutputfile, 322  
 \openright, 128  
 openright (option), 66, 128  
 option, , xxi  
     \*pt, 65, 432  
     10pt, 64, 65, 67, 98  
     11pt, 64, 98  
     12pt, 64, 67, 98  
     14pt, 64, 98, 168  
     17pt, 64, 65, 98, 431  
     20pt, 64, 65, 98

- 25pt, 64, 98, 431
- 30pt, 64, 98
- 36pt, 65, 98
- 48pt, 65, 98
- 60pt, 65, 98
- 9pt, 64, 98
- a3paper, 63
- a4paper, 63, 86, 88
- a5paper, 63
- a6paper, 63
- article, 66, 122, 123, 128, 132, 151
- b3paper, 63
- b4paper, 63
- b5paper, 63
- b6paper, 63
- broadsheetpaper, 64
- crownvopaper, 64
- dbillpaper, 64
- demyvopaper, 64
- draft, 66, 169, 329, 361
- ebook, 63, 67
- executivepaper, 64
- extrafontsizes, 65, 431
- final, 66, 329, 361
- fleqn, 66
- foolscapvopaper, 64
- imperialvopaper, 64
- landscape, 63, 85
- largecrownvopaper, 64
- largepostvopaper, 64
- ledgerpaper, 64
- legalpaper, 64
- leqno, 66
- letterpaper, 63, 64, 67, 86, 88
- mcrownvopaper, 63
- mdemyvopaper, 63
- mediumvopaper, 64
- mlargecrownvopaper, 63
- ms, 66
- msmallroyalvopaper, 63
- oldfontcommands, 67, 430
- oldpaper, 64
- onecolumn, 66, 213
- oneside, 66, 67, 85, 128
- openany, 66, 128
- openbib, 66, 339
- openleft, 66, 128
- openright, 66, 128
- postvopaper, 64
- pottvopaper, 64
- royalvopaper, 64
- sectionbib, 338
- showtrims, xvi, 66, 362
- smalldemyvopaper, 64
- smallroyalvopaper, 64
- statementpaper, 64
- superroyalvopaper, 64
- titlepage, 112, 117
- twocolumn, 66, 67, 117–119, 213
- twoside, 66, 67, 85, 128, 281
- Option clash for ..., 426
- Optional argument is not one of: ..., 431
- Optional argument of \twocolumn ..., 429, 433
- \or, 414
- ordinal,
  - number, 58
- \ordinal, 371
- \OrdinalToName, 372
- \ordinaltoName, 372
- \ordinaltoname, 372
- \ordscript, 371
- orientation, 45
  - landscape, 45
  - portrait, 45
- orphan, 38
  - line, 104, 105
- postbibhook, 338
- outline,
  - font, 93, 97, 101
- \output, 420
- Output loop ..., 420
- Output stream ... is already defined, 433
- Output stream ... is not open, 433
- Output stream ... is open, 433
- \oval, 429
- \oval, \circle, or \line size unavailable, 429
- Overfull \hbox ..., 420
- Overfull \vbox ..., 420
- overfull lines, 104
- overfull, 405
- \overridescapmargin, 242
- \overridesidecapmargin, 430

**P**

p (position argument), 214, 220, 224

package, , xxi, 118, 327, 391

abstract, xix, 117, 380

afterpage, 210, 233

alltt, 293, 395

amsfonts, 425

answer, 331

appendix, xix, 380

array, xvii, xix, 249, 380

babel, xv

booktabs, xvii, xix, 249, 259, 380

bringhurst, 391, 392

calc, 84, 86, 284

ccaption, xvii, xix, 225, 380

change page, 368

chapterbib, 338

chngcntr, xix, 365, 380

chngpage, xix, 180, 368

color, 142, 310, 395

comment, 315

crop, xx, 380

datetime, 365

dcolumn, xvii, xix, 249, 380

delarray, xvii, xix, 249, 380

endnotes, 355

enumerate, xix, 182, 380

epigraph, xix, 287, 380

fancybox, 311

fancyhdr, xv, xvi, xx, 164, 380

fixltx2e, 173, 220, 395

flafter, 221

footmisc, 274, 276

framed, xix, 310, 313, 380

geometry, xvi, xx, 69, 380

graphicx, 142, 144, 150, 395

hanging, 178

hyperref, xv, xix, 194, 238, 334, 343,

344, 381, 382

ifmtarg, xix, 380

ifpdf, xix, 380

index, xix, 295, 380

jurabib, 340

latexsym, 425

layouts, 88, 89, 196, 395, 396

ledmac, 274, 276

lettrine, 398

lmodern, 65

longtable, xvii, 268

makebst, 340

makeidx, xix, 380

mathpazo, 76

memexsupp, xvi

memhfixc, xix, 381, 382

memsty, 204, 396, 398

midpage, 233

moreverb, xix, 380

multicol, 204, 220, 310

multind, 295

nameref, 334

natbib, 338, 340, 349

needspace, xix, 380

newfile, xix, 380

nextpage, xix, 369, 380

pagenote, 355, 380

parskip, xix, 380

patchcmd, xix, 368, 380

pifont, 158

sectsty, 121

setspace, xix, xx, 102, 380

shortvrb, xix, 315, 380

showidx, xix, 380

sidecap, xx, 227, 380

subfig, xvii

subfigure, xx, 236, 380

tabularx, xvii, xix, 249, 380

titleref, xix, 334, 380

titlesec, xx, 121, 380

titling, xix, 108

tocbibind, xix, 189, 380

tocloft, xv, xix, 189, 380

to cvsec2, 125, 194

url, 395

verbatim, xix, 316, 380

verse, xix, 380

wrapfig, 243

xcolor, 310, 395

xtab, xvii, 269

packages before class, 76

page, xxiii, 74

break, 268, 269

copyright, 43, 44

current number, 333

- half-title, 20, 22, 24, 31, 43, 44
- height, 71, 77, 79
- location, 82
- of floats, 174
- part title, 44
- reference, 334
- size, 71, 81
- specifying size, 71, 80
- title, 22, 24, 31, 43
- width, 71, 77, 78
- page break,
  - asynchronous, 401
- page color, 34, 67
- page (counter), 163
- Page height already too large, 426
- page layout, 69, 81
  - check parameters, 82
  - class parameters, 71, 81, 84
  - default, 69
  - design, 71
  - dvips, 85
  - LaTeX parameters, 70, 81, 84
  - parameters, 82
  - PDF, 85
- \pageaiai, 63
- \pageaiv, 63
- \pageav, 63
- \pageavi, 63
- \pagebiii, 63
- \pagebiv, 63
- \pagebroadsheet, 64
- \pagebv, 63
- \pagebvi, 63
- \pagecrownvo, 64
- \pagedbill, 64
- \pagedemyvo, 64
- \pageexecutive, 64
- \pagefoolscapvo, 64
- \pageimperialvo, 64
- \pageinnotes, 359
- \pagelargecrownvo, 64
- \pagelargepostvo, 64
- \pageledger, 64
- \pagelegal, 64
- \pageletter, 64
- \pagemdemyvo, 63
- \pagemediumvo, 64
- \pagemetriccrownvo, 63
- \pagemlargecrownvo, 63
- \pagemsmallroyalvo, 63
- \pagename, 378
- \pagenote, 355, 356, 358–360
- pagenote (counter), 357, 358
- pagenote (package), 355, 380
- \pagenotesubhead, 359
- \pagenumbering, 163
- \pagenumbering\*, 163
- \pageold, 64
- \pagepostvo, 64
- \pagepottvo, 64
- \pageref, 211, 333, 428, 429
- \pagerefname, 334, 378
- \pageroyalvo, 64
- \pagesmalldemyvo, 64
- \pagesmallroyalvo, 64
- \pagestatement, 64
- pagestyle, xxi, 69
  - book*, 125, 165
  - chapter*, 123, 128, 165, 196
  - class*, 164
  - cleared*, 165
  - combining*, 174
  - companion*, 165, 171–174, 396
  - empty*, 112, 127, 164, 165, 167, 169, 196
  - epigraph*, 165, 292
  - float pages*, 174–176
  - floatcomp*, 174
  - headings*, 69, 164, 165, 169
  - index*, 173
  - index pages*, 173
  - myheadings*, 164–166
  - part*, 125, 165
  - plain*, 108, 112, 164–166, 169, 174, 290
  - Ruled*, 165
  - ruled*, 69, 164, 165, 172
  - section*, 130
  - specifying*, 164–173
  - title*, 108, 165
  - titlingpage*, 112, 165
- \pagestyle, 130, 164, 169, 195, 208
- \pagesuperroyalvo, 64
- \pagetofootnote, 359, 360
- \pagewidth, 78

pagination, xxiii, 3, 163

changing, 163

interrupt, 163

Palatino, 29, 75, 76

paper, xxiii, 6, 9–11, 14, 41, 63, 69, 88, 210

antique, 10

Arches, 11

coated, 10

cover, 10

Curtis Rag, 11

decorative, 10

end, 7, 201

English finish, 10

Fabrizio, 11

Glatfelter, 11, 20–22

impregnated, 10

Ingres, 10

Japanese, 10

Kozo, 10

Linweave Early American, 11

machine finish, 10

machine-made, 10

Mohawk superfine, 11

moldmade, 10

pigmented, 10

printing, 9

size, 14, 63

A3, 63

A4, 63, 86, 181

A5, 63

A6, 63

B3, 63

B4, 63

B5, 63

B6, 63

broadsheet, 64

crown octavo, 64

demy octavo, 64

dollar bill, 64

executive, 64

foolscap octavo, 64

imperial octavo, 64

large crown octavo, 64

large post octavo, 64

ledger, 64

legal, 64

letterpaper, 14, 28, 29, 63, 64, 86, 181, 386

medium octavo, 64

metric crown octavo, 63

metric demy octavo, 63

metric large crown octavo, 63

metric small royal octavo, 63

old, 64

post octavo, 64

pott octavo, 64

royal octavo, 64

small demy octavo, 64

small royal octavo, 64

statement, 64

super royal octavo, 64

special, 9

stock, 69

Strathmore, 11

super-calendered, 10

text, 10

Warren's Old Style, 11

wrapping, 9

writing, 9

`\paperheight` and/or `\trimtop` are too large ..., 431

`\paperheight` (length), 71, 81, 83, 431, 432

`\paperwidth` and/or `\trimedge` are too large ..., 431

`\paperwidth` (length), 71, 81, 83, 181, 386, 431

`\par`, 206, 207, 416, 420

paragraph, 31, 34, 38–40, 67, 177–181, 201, 293, 385, 388

block, 154, 177, 226, 389

reasons against, 177

hanging, 154, 178–179, 181, 389

in list, 178

indentation, 39, 118, 157, 177, 180, 227, 288

outdentation, 177

typeset as a unit, 178, 180

`\paragraph`, 121, 124, 156, 197, 377

paragraph break,

invisible, 281

Paragraph ended before ..., 420

paragraph indentation, 102

- \paragraphfootnotes, 275
- \paragraphfootstyle, 276
- \paraheadstyle, 160
- parallel texts, 269
- \parbox, 156, 250, 265
- \parfillskip, 201
- \parg, 383
- \parindent (length), 76, 102, 177, 179, 250
- \parnopar, 281, 282
- \parsep (length), 339
- \parshape, 181, 295
- parskip (package), xix, 380
- \parskip (length), 101, 102, 177, 197, 245
- part, 5, 6, 44, 121, 191–193
  - number, 6, 122
- \part, 66, 121, 122, 124–128, 165, 197–199, 292, 378, 382
- part* (pagestyle), 125, 165
- part title
  - page, 44
- \part\*, 382
- \partmark, 127
- \partname, 126, 378
- \partnamefont, 126, 160
- \partnamenum, 126
- \partnumberline, 199
- \partnumfont, 126, 160
- \partopsep (length), 186
- \partrefname, 334, 378
- \parttitlefont, 126, 127, 160
- de Parville, Henri, 411
- \patchcmd, 431
- patchcmd (package), xix, 368, 380
- \patchcommand, 368
- patverse (environment), 297, 299, 300, 305, 374, 430
- patverse\* (environment), 300, 430
- pc, xxiv
- \Pcstyle, 396
- PDF, 85
- pdf (file), 321
- pdfLaTeX, 375, 376
- pedersen* (chapterstyle), 121, 142, 148
- Pedersen, Troels, 142
- pederson* (chapterstyle), 161
- period, *see also* full stop, 47
- \pfbreak, 157, 158
- \pfbreak\*, 157, 158
- \pfbreakdisplay, 157, 158
- \pfbreakskip (length), 157
- \phantomsection, 194
- Phidias, 16
- pica, xxiii, xxiv
- picture (environment), 402, 423, 424, 429
- pifont (package), 158
- pilcrow (¶), 38, 182
- plain* (pagestyle), 108, 112, 164–166, 169, 174, 290
- \plainbreak, 157
- \plainbreak\*, 157
- \plainfancybreak, 157
- \plainfancybreak\*, 157
- \plainfootnotes, 275
- \plainfootstyle, 276
- \PlainPoemTitle, 301, 303
- plates, 210
- play, 49
  - act, 49
  - scene, 49
  - speaker, 49
  - stage directions, 49
- Please type a command ..., 421
- plus (keyword), 407
- plus, 407, 409
- \pmname, 365
- \pmname, 365, 378
- \pnchap, 359
- \pnschap, 359
- poem, , 49
  - author, 303
- poem title, 301, 303
  - in ToC, 301, 303
  - styling, 305
- poemline (counter), 300
- \PoemTitle, 300, 301, 383
- \poemtitle, 303, 334, 383
- \PoemTitle\*, 301
- \PoemTitlefont, 302
- \PoemTitleheadstart, 301, 302
- \poemtitlemark, 301
- \PoemTitlenumfont, 302
- \poemtitlepstyle, 301

---

\poemtitlestarmark, 301	\printchaptertitle, 129, 195, 388
\poemtitlestarpstyle, 301	\printendnotes, 324
\poemtoc, 301, 303	\printfibterm, 413
point, xxiii, xxiv	\printfigures, 328
pool size, 424	\printglossary, 350, 355
\poptabs, 426	\printindex, 340, 341
portrait, 45	\printnotes, 357, 358
orientation, 45	\printnotes*, 357–359
\postauthor, 111	\printpageinnotes, 359
\postbibhook, 338	\printpagenotes, 356
\postcaption, 227, 236	\printpagenotes*, 356
\postchapterprecis, 151, 212	\printpartname, 126
\postdate, 111	\printpartnum, 126
\postnoteinnotes, 358	\printparttitle, 126, 127
\posttitle, 111	\printPoemTitlenonum, 301, 302
postvopaper (option), 64	\printPoemTitlenum, 301, 302
pottvopaper (option), 64	\printPoemTitletitle, 301, 302
\Ppstyle, 396	\printtime, 365
preamble, 67, 86, 116, 125, 173, 193, 194,	\printtime*, 365
197, 200, 227, 280, 327, 340, 391,	\printXtitle, 195
392, 396, 425, 427	\priorpageref, 211
\preauthor, 111	program,
\prebibhook, 292, 338	BibTeX, xv, 339, 340
\precaption, 227, 236	dvips, 85
\prechapterprecis, 151, 212	ghostview, 85
\prechapterprecisshift (length),	gsvie32, 85
151	MakeIndex, xv, xix, 344, 347, 349–
\precisfont, 151	352, 355, 396
\precistocfont, 151, 203, 212	xindy, 344
\precistocfont, 151, 203, 212	proportion, 14, 16
\predate, 111	book, 71
\Pref, 334, 378	margin, 77
\pref, 334, 378	page, 14–16, 18–31
preface, 4, 44	typeblock, 18–31, 75
\preglossaryhook, 353	protect, 159, 330, 331
\preindexhook, 292, 341	\protect, 159, 192, 193, 204, 247, 322,
preliminaries, 3	326, 331, 335, 359, 416
\prenoteinnotes, 358	\providecommand, 366, 403, 404, 418
\pretile, 111	\providecommand*, 403
\printanswer, 329	\providecounter, 366, 433
\printanswers, 329	\provideenvironment, 366, 433
\printbookname, 126	\providefixedcaption, 233
\printbooknum, 126	\providelength, 366, 433
\printbooktitle, 126	\provideloglike, 366
\printchaptername, 129, 134	\provideloglike*, 366
\printchapternonum, 130, 134, 135	pseudo (counter), 327
\printchapternum, 129	psnfss, 75

\pstyle, 396  
 pt, xxiv  
 \published, 112  
 \pushtabs, 426  
 \pushtabs and \poptabs don't match, 426  
 \pwayii (length), 389, 390  
 \pwayi (length), 389, 390

## Q

\qbezier, 423  
 qfame (environment), 313  
 qframe (environment), 313  
 \qitem, 287, 288  
 qshade (environment), 313  
 quad, xxv  
 \quarkmarks, 363, 365  
 quarto, 7  
 question (counter), 330  
 questions and answers, 328–330  
 \quietref, 330  
 quotation, 45, 54, 177, 180, 287, 290  
 quotation (environment), 117, 180, 182, 183, 313, 427  
 quotation marks, 54–55  
 quote (environment), 151, 180, 182, 183, 212

## R

r (position argument), 269, 270  
 rabbit, 17  
 \raggedbottom, 67, 104, 123, 274, 369  
 \raggedbottomsection, 123  
 raggedleft, 179  
 \raggedleft, 179, 180, 226, 239, 243, 265  
 raggedright, 56, 179, 283  
   index, 57  
 \raggedright, 179, 226, 239, 243, 265, 283  
 \raggedyright, 179  
 \ragrparindent (length), 179  
 FrameTitle, 312  
 ranging,  
   figures, 58  
 \readboxedverbatim, 323

\readboxedverbatim\*, 323  
 \readline, 323  
 \readstream, 323  
 \readverbatim, 323  
 \readverbatim\*, 323  
 recto, xxiii  
 Redefining primitive column ..., 433  
 \ref, 238, 328, 330, 333, 428, 429  
 reference,  
   by name, 334–336  
   by number, 333–334  
   current value, 333  
   set current value, 333, 334  
   to appendix, 334  
   to book, 334  
   to chapter, 334  
   to counter, 343  
   to figure, 334  
   to label, 333  
   to page, 333, 334, 343  
   to part, 334  
   to section, 334  
   to table, 334  
   unexpected result, 333, 335  
 Reference ... on page ..., 429  
 reference mark, *see also* endnote mark,  
   footnote mark, 46, 47, 280  
 \refstepcounter, 329, 333, 334, 406  
 registered trademark, 379  
 \registrationColour, 363  
 \relax, 170, 406  
 \renewcommand, 134, 168, 191, 198, 199, 232, 245, 322, 366, 367, 369, 403, 404, 418  
 \renewcommand\*, 242, 367, 403  
 \renewenvironment, 418  
 \renewfixedcaption, 233  
 \renewleadpage, 127  
 \renewleadpage\*, 127  
 reparticle (chapterstyle), 132  
 \repeat, 413  
 report (class), xv, 111, 117, 134, 159, 192  
 \reportnoidxfile, 341  
 representation,  
   number, 370  
 \RequireAtEndClass, 381  
 \RequireAtEndPackage, 380, 381

\RequirePackage, 76, 426  
 \RequirePackage or \LoadClass in  
     Options Section, 426  
 \RequireXeTeX, 376  
 \resetbvlinenumber, 318  
 \resizebox, 150  
 \restorepagenumber, 163  
 \storetrivseps, 186  
 \reversemarginpar, 280  
 \reversesideparfalse, 281  
 \reversesidepartrue, 281  
 \right, 255, 256, 417, 419  
 \rightmark, 166, 169, 171–173  
 Riverside Press, 20  
 \rmfamily, 94  
 Rogers, Bruce, 22  
 roman numerals, 163  
 Roman, 163  
 roman, 163  
 royalvopaper (option), 64  
 rubricator, 39  
 rule,  
     in margin, 311  
     invisible, 174  
     thickness, 80, 168, 174  
 ruled,  
     float, 215  
 Ruled (pagestyle), 165  
 ruled (pagestyle), 69, 164, 165, 172  
 Runaway argument, 421  
 Runaway definition, 421  
 Runaway preamble, 421  
 Runaway text, 421  
 running BibTeX, 340

## S

Sanderson, Donald, 108  
 sans, 35  
 save stack size, 424  
 \savebox, 309  
 \savepagenumber, 163  
 \savetrivseps, 186  
 \saythanks, 119  
 \sbox, 309  
 Scala Sans, 20  
 \scapmargleftfalse, 240

\scapmarglefttrue, 240  
 scene  
     play, 49  
 Schusterjungen, *see* orphan, 38  
 scrbook (class), 139, 162  
 \scriptsize, 97, 98  
 \scshape, 94  
 \secheadstyle, 160  
 secnumdepth (counter), 122, 124, 125  
 section, *see also* subhead, 6  
     design, 21, 29  
 \section, 121–124, 132, 153, 154, 172,  
     192, 197, 204, 283, 333, 335, 382,  
     389  
 section (chapterstyle), 130, 132, 134, 135  
 section (pagestyle), 130  
 \section\*, 122  
 sectional division, *see also* subhead  
     level number, 124  
 sectionbib (option), 338  
 \sectionmark, 171  
 \sectionrefname, 334, 378  
 sectsty (package), 121  
 \see, 346, 347, 378  
 \seealso, 346, 347, 378  
 \seename, 347, 378  
 \semiisopage, 88, 89  
 serif, 35  
 serif versus sans-serif type, 35–38  
 \setaftersecskip, 388  
 \setaftersskip, 154  
 \setarrayelement, 375  
 \setbeforesecskip, 388  
 \setbeforesskip, 153  
 \setbiblabel, 338, 339  
 \setbvlینums, 318  
 \setcolsepandrulе, 79, 80  
 \setcounter, 163, 189, 326, 366, 406,  
     424, 426, 431  
 \setDisplayskipStretch, 103  
 \setfloatlocations, 220  
 \sethangfrom, 154, 155, 389  
 \setheaderspaces, 80–82  
 \setheadfoot, 80, 82  
 \setlength, 71, 81, 118, 129, 168, 183,  
     198, 242, 245, 288, 289, 296, 409  
 \setlrmargins, 77, 79, 82

---

\setlrmarginsandblock, 78, 82  
\setlxvchars, 75  
\setmarginnotes, 81, 82, 240, 284  
\setpagebl, 89  
\setpageml, 89  
\setpagetl, 89  
\setparahook, 156  
\setpnumwidth, 196, 197  
\setrmarg, 196  
\setseheadstyle, 388  
\setsecnumdepth, 124, 125, 194, 431  
\setsecnumformat, 154–156  
\setShheadstyle, 153  
\setShook, 155  
\setsidcaps, 240  
\setsidebarheight, 283, 284  
\setsidebars, 284  
\setsidecappos, 241, 430  
\setsidecaps, 240  
\setSindent, 153  
\SetSingleSpace, 103  
setspace (package), xix, xx, 102, 380  
\setstocksize, 71, 81  
\setsubseheadstyle, 389  
\settocdepth, 189, 194, 391, 431, 434  
\settodepth, 409  
\settoheight, 409  
\settowidth, 409  
\settrimmedsize, 71, 74, 76, 81, 89  
\settrims, 74, 82  
\settypeblocksize, 76, 82, 368  
\setulmargins, 79, 80, 82  
\setulmarginsandblock, 79, 82  
\setupmaintoc, 206  
\setupparasubsecs, 206, 207  
\setupshorttoc, 205  
\setverbatim, 315  
\setverbatimfont, 315  
\setverselinenums, 300  
\setxlvchars, 75  
sexto, 7  
\sfffamily, 94, 246  
\sffseries, 385  
shaded (environment), 310, 311, 313  
sheet, 71  
sheetsequence (counter), 364  
\Shook, 155  
\shortsubcaption, 239, 240  
shortvrb (package), xix, 315, 380  
\showcols, 254  
showidx (package), xix, 380  
\showindexmarks, 343  
\showtrimeson, 362  
showtrims (option), xvi, 66, 362  
\showtrimsoff, 362  
\showtrimson, 362  
side note, 281  
    adjust position, 281  
    specifying the margin, 281  
    text for particular margin, 281  
sidebar, 148, 149, 283  
    location, 283  
    styling, 283  
\sidebar, 283  
\sidebarfont, 283, 284  
\sidebarform, 283  
\sidebarhsep (length), 283, 284  
\sidebarmargin, 283  
\sidebartopsep (length), 283, 284  
\sidebarvsep (length), 283, 284  
\sidebarwidth (length), 283, 284  
sidecap (package), xx, 227, 380  
\sidecapfloatwidth, 242  
\sidecapmargin, 240, 242, 432  
\sidecapraise (length), 242  
\sidecapsep (length), 240, 242  
\sidecapstyle, 241  
\sidecaption, 432  
sidecaption (environment), 240, 241, 243  
\sidecapwidth (length), 240  
sidecontcaption (environment), 241  
sidelegend (environment), 241  
sidenamedlegend (environment), 241  
sidenote, 21, 26  
\sidepar, 281  
\sideparswitchfalse, 281  
\sideparswitchtrue, 281  
\sideparvshift (length), 281  
signature, xxiii, 6–9  
single column,  
    index, 340, 341  
\SingleSpacing, 103  
\sixt@@n, 406

- size,
  - page, 71, 81
  - paper, 63
  - stock, 63, 69, 71, 81, 85
  - typeblock, 82
- Size substitutions ..., 429
- `\skip`, 407–409, 426
- `\slashfrac`, 373
- `\slashfracstyle`, 373
- `\sloppy`, 104, 422
- `\sloppybottom`, 105
- `sloppypar` (environment), 104
- `\slshape`, 94
- `\small`, 97, 98, 178, 266, 289, 320, 385, 397, 398
- `smalldemyvopaper` (option), 64
- `smallroyalvopaper` (option), 64
- `snugshade` (environment), 310, 311
- `snugshaded` (environment), 310
- Some shapes ..., 429
- Something's wrong ..., 426
- Sorry, but I'm not ..., 421
- `source2e.tex` (file), 403
- southall* (chapterstyle), 144, 149
- space,
  - at start of paragraph, *see* paragraph indentation
  - between lines, *see* leading
  - double, *see* (ouble spacing)i
  - gobble, 329
  - inter-paragraph, 101–102
  - interline, 51
  - interword, 51, 104
  - thin, 54
- `\spacefactor`, 422
- `\span`, 417, 418
- speaker
  - play, 49
- `\specialindex`, 343
- `\specialrule`, 260
- `\spinewidth` and/or `\textwidth` and/or `\foremargin` are too large ..., 431
- `\spinewidth` (length), 82, 83, 181, 431
- spread, 18–31, 71, 77
- `\Sref`, 334, 378
- stage directions, 49
- standard class, xv
- stanza,
  - end, 293
  - indent alternate lines, 299, 303
  - indent pattern, 299, 307
  - last line, 296
    - numbering, 296
  - line break, 297, 304
    - indent, 297
  - long line, 304
  - number, 300, 306
  - prevent page break, 296
  - space, 296
- `\stanzaskip` (length), 296
- start new page, 233
- `statementpaper` (option), 64
- `\stepcounter`, 406
- stock, xxiii, 66, 69–71, 74, 81, 82, 86, 386
  - commercial printing, 69
  - default, 63
  - height, 71
  - paper, 69
  - size, 63, 69, 71, 81, 85
    - A3, 63
    - A4, 63, 69
    - A5, 63
    - A6, 63
    - B3, 63
    - B4, 63
    - B5, 63
    - B6, 63
  - broadsheet, 64
  - crown octavo, 64
  - demy octavo, 64
  - dollar bill, 64
  - ebook, 63
  - executive, 64
  - foolscap octavo, 64
  - imperial octavo, 64
  - large crown octavo, 64
  - large post octavo, 64
  - ledger, 64
  - legal, 64
  - letterpaper, 63, 64, 69
  - medium octavo, 64
  - metric crown octavo, 63
  - metric demy octavo, 63

- metric large crown octavo, 63
- metric small royal octavo, 63
- old, 64
- post octavo, 64
- pott octavo, 64
- royal octavo, 64
- small demy octavo, 64
- small royal octavo, 64
- statement, 64
- super royal octavo, 64
- specifying size, 71, 81
- specifying trimming, 74
- trimmed, 71, 74
- width, 71
- stock paper size option, 71
- \stockaiii, 63
- \stockaiv, 63
- \stockav, 63
- \stockavi, 63
- \stockbiii, 63
- \stockbiv, 63
- \stockbroadsheet, 64
- \stockbv, 63
- \stockbvi, 63
- \stockcrownvo, 64
- \stockdbill, 64
- \stockdemyvo, 64
- \stockexecutive, 64
- \stockfoolscapvo, 64
- \stockheight (length), 81, 83, 431
- \stockimperialvo, 64
- \stocklargecrownvo, 64
- \stocklargepostvo, 64
- \stockledger, 64
- \stocklegal, 64
- \stockletter, 64
- \stockmdemyvo, 63
- \stockmediumvo, 64
- \stockmetriccrownvo, 63
- \stockmlargecrownvo, 63
- \stocksmallroyalvo, 63
- \stockold, 64
- \stockpostvo, 64
- \stockpottvo, 64
- \stockroyalvo, 64
- \stocksmalldemyvo, 64
- \stocksmallroyalvo, 64
- \stockstatement, 64
- \stocksuperroyalvo, 64
- \stockwidth (length), 81, 83, 386, 431
- \stop, 421
- stream, 321–330
  - check open, 322
  - close input, 323
  - close output, 322
  - input, 321, 323
  - limited number, 321
  - new input, 321
  - new output, 321
  - output, 321, 322, 328, 329
- Stream ... is not open, 433
- \strictpagecheck, 180, 281, 369
- \strictpagecheck, 242
- \stringtoarray, 375
- \strip@pt, 389
- subappendices (environment), 124
- \subbottom, 220, 237
- \subcaption, 220, 236, 237
- \subcaptionfont, 239
- \subcaptionlabelfont, 239
- \subcaptionref, 238
- \subcaptionsize, 239
- \subcaptionstyle, 239
- \subconcluded, 238
- subfig (package), xvii
- subfigure (package), xx, 236, 380
- subhead, 6, 121
  - moved, 123
  - near bottom of page, 123
- \subitem, 343
- \subparagraph, 377
- \subparagraph, 121, 124, 154, 197
- \subparaheadstyle, 160
- subscript, 34
- \subseheadstyle, 160
- \subsection, 121, 122, 124, 153, 197, 206, 382
- \subsection\*, 324
- \subsubitem, 343
- \subsubseheadstyle, 160
- \subsubsection, 121, 124, 154, 197
- \subtop, 220, 237
- Suggested extra height ..., 426
- superroyalvopaper (option), 64

superscript, 34, 379  
 \suppressfloats, 220  
 symbol, 6, 55, 379  
   list, 48  
 symbols (environment), 185  
 \symbolthanksmark, 114  
 Syntax, 385  
 syntax (environment), 397

## T

t (position argument), 214, 220, 224  
 Tab overflow, 426  
 tabbing (environment), 424, 426  
 \tabcolsep (length), 267  
 table, 3, 14, 32, 34, 57–58, 174, 192, 209,  
   213, 216, 225, 229, 231–233, 236,  
   385, 390, 391  
   column  
     cut-in head, 257  
     decked head, 257  
     head, 257  
     spanner head, 257  
     spanner rule, 257  
     stub, 257  
   half and half, 259  
   long, 268  
   reference, 334  
   row fill, 261–263  
   rule, 258–261  
     horizontal, 257  
     vertical, 257  
   subtable, 236–240  
 table (environment), 213, 231, 232, 269,  
   425  
 \tablename, 213, 232, 378  
 \tableofcontents, 123, 165, 189, 190,  
   192, 194, 197, 207, 378, 401, 402  
 \tableofcontents\*, 189, 194  
 \tablerefname, 334, 378  
 \tabsoff, 316  
 \tabson, 316  
 tabular, 263–266, 417  
   automatic, 270–271  
     by column, 271  
     by row, 270  
     column style, 270

    column width, 270, 271  
     position, 270  
     table width, 270, 271  
   column width, 263, 264  
   continuous, 269  
     position, 269  
   controlling width, 263  
   free, 268–271  
     caption, 269  
   intercolumn space, 263, 264, 267  
   new column type, 265  
   raggedright, 265  
   row spacing, 267  
   rule, 264  
   vertical position, 267  
   X column, 263, 266  
 tabular (environment), 216, 243, 249,  
   251, 254, 263–265, 267, 269, 273,  
   397, 417, 419, 420, 423, 425, 426,  
   430, 431  
 tabular\* (environment), 249, 263, 264,  
   267  
 tabularx (environment), 249, 263–266,  
   434  
 tabularx (package), xvii, xix, 249, 380  
 \tabularxcolumn, 265  
 tandh (chapterstyle), 144, 162  
 tandh (headstyles), 161, 162  
 \tensunitsep, 372  
 TeX,  
   error, 415–424  
   warning, 415–422  
 TeX capacity exceeded ..., 421–424  
 tex (file), 321  
 text,  
   figures, 58  
 Text line contains ..., 421  
 Text page ... contains only floats, 429  
 \textasteriskcentered, 182  
 \textbf, 94  
 \textbullet, 182  
 \textendash, 182  
 \textfloatsep (length), 223, 225  
 \textfraction, 223, 224  
 \textheight (length), 76, 82–84, 284,  
   432  
 \textit, 94

- \textmd, 94
- \textperiodcentered, 182
- \textregistered, 379
- \textrm, 94
- \textsc, 94
- \textsf, 94
- \textsl, 94
- \textsubscript, 374
- \textsuperscript, 379
- \textsuperscript, 374
- \texttrademark, 379
- \texttt, 94
- \textup, 94
- textwidth, 75
- \textwidth (length), 76, 78, 82, 83, 168, 181, 268, 431
- thanks, 107, 114
  - styling, 114–116
- \thanks, 107, 112, 114–116, 119, 275, 277, 433
- \thanksfootextra, 116
- \thanksfootmark, 115, 116
- \thanksfootpost, 116
- \thanksfootpre, 116
- \thanksheadextra, 114, 115
- \thanksmark, 115
- \thanksmarksep (length), 115
- \thanksmarkseries, 114
- \thanksmarkstyle, 115
- \thanksmarkwidth (length), 115
- \thanksrule, 116
- That makes 100 errors ..., 421
- thatcher* (chapterstyle), 144
- Thatcher, Scott, 144
- \the, 407–409
- The ... font command is deprecated ..., 433
- The ‘extrafontsizes’ option ..., 431
- The combination of argument values ..., 431
- The counter will not be printed ..., 433
- The file needs format ..., 426
- \theauthor, 113, 114
- \thebibliography, 378
- thebibliography (environment), 165, 337, 338, 353, 426, 433
- \thechapter, 326
- \thectr, 365
- \thedate, 113, 114
- \theglossary, 378
- theglossary (environment), 350, 351, 353
- \theindex, 378
- theindex (environment), 165, 341, 343, 353
- \thepage, 163, 364
- \thepoem, 301
- \thepoemline, 300
- There were multiply defined labels, 429
- There were undefined references, 429
- There’s no line to end here, 427
- \thesheetsequence, 364
- \thetitle, 113, 114
- \thinspace, 377
- This can’t happen ..., 421
- This may be a LaTeX bug, 427
- \thispagestyle, 164
- \thr@@, 406
- \threecolumnfootnotes, 275
- \threecolumnfootstyle, 276
- tightcenter (environment), 397
- \tightlist, 183
- \tightlists, 183
- \tightsubcaptions, 239
- Times, 75, 76
- Times Roman, 75
- \tiny, 97, 98
- tip in, 9, 163, 210
- title, 107
  - styling, 108–114
- \title, 107, 111, 113, 426, 433
- title page, 3, 21, 43, 107, 108, 112–113, 379
  - bastard, 3
  - half-title, 3
- title* (pagestyle), 108, 165
- titledframe (environment), 313
- titlepage (environment), 112
- titlepage (option), 112, 117
- \titleref, 334, 335
- titleref (package), xix, 334, 380
- titlesec (package), xx, 121, 380
- titling,
  - figures, 58
- titling (package), xix, 108

titlingpage (environment), 112, 113,  
     116, 164, 165  
 titlingpage (pagestyle), 112, 165  
 \tmarkbl, 362  
 \tmarkbm, 362  
 \tmarkbr, 362  
 \tmarkml, 362  
 \tmarkmr, 362  
 \tmarktl, 362  
 \tmarktm, 362  
 \tmarktr, 362  
 ToC, 3, 4, 118, 122–124, 127, 150, 151, 159,  
     172, 189, 191–208, 210–212, 301,  
     303, 309, 333, 335, 337, 341, 370,  
     385, 388, 391, 416, 423  
     controlling entries, 214  
     design, 20–22, 29  
 toc (file), 151, 189, 190, 203, 212, 321, 356,  
     401, 402  
 tocbibind (package), xix, 189, 380  
 tocdepth (counter), 189, 194, 205, 208,  
     214  
 tocloft (package), xv, xix, 189, 380  
 \tocmark, 195  
 \tocnameref, 335  
 tocvsec2 (package), 125, 194  
 token, 400  
 Too deeply nested, 427  
 Too many }'s, 421  
 Too many columns ..., 427  
 Too many unprocessed floats, 427  
 \topfraction, 223–225  
 \topmargin (length), 83  
 topnumber (counter), 223  
 \toprule, 259, 260  
 \topsep (length), 186  
 \topskip, 389  
 \topskip (length), 82, 84, 105  
 totalnumber (counter), 223  
 to, 405  
 \tracingtabularx, 264, 265  
 trademark, 379  
 \traditionalparskip, 101, 102  
 \tref, 334, 378  
 trim,  
     specifying, 82  
 \trimedge (length), 82, 83, 86, 386, 431

\trimFrame, 362  
 \trimLmarks, 362  
 \trimmark, 363  
 \trimmarks, 362  
 \trimNone, 362  
 \trimtop (length), 82, 83, 86, 431  
 \trimXmarks, 362, 363  
 trivialist (environment), 186, 187, 320,  
     397  
 Tschichold, Jan, 31  
 \ttfamily, 94  
 TUG, xx  
 Turing complete language, 399  
 Turing, Alan, 399  
 \tw@, 406  
 Two \documentclass commands, 427  
 Two \LoadClass commands, 427  
 \twocolglossary, 353  
 \twocolindex, 341  
 \twocolumn, 119, 429, 433  
 twocolumn (option), 66, 67, 117–119, 213  
 \twocolumnfootnotes, 275  
 \twocolumnfootstyle, 276  
 twoside (option), 66, 67, 85, 128, 281  
 \TX@verb, 266  
 type size, 64  
     default, 65  
 typeblock, 13, 18, 24, 26, 27, 31–35, 39–42,  
     67, 69, 71, 76–81, 86, 88, 113, 168,  
     171, 181, 182, 287, 288, 296, 385,  
     386, 388–390  
     height, 76, 77, 79  
     location, 77, 80, 82  
     size, 82  
     specifying size, 76, 79, 80, 82  
     width, 76–78  
 typeface, 75  
 \typeout, 75  
 \typeoutlayout, 84  
 \typeoutstandardlayout, 84  
 Typewriter, 75, 76, 266

## U

\ucminusname, 373, 378  
 Unbalanced output routine, 421  
 Unbalanced write command, 422

$\backslash$ unboldmath, 428  
 Undefined control sequence, 422  
 Undefined index file ..., 434  
 Undefined tab position, 427  
 Underfull  $\backslash$ hbox ..., 422  
 Underfull  $\backslash$ vbox ..., 422  
 underfull, 405  
 underline, 57  
 $\backslash$ undodrop, 290, 291  
 $\backslash$ unitlength (length), 182, 207, 290, 389  
 University of California Old style, 27  
 Unknown document division ..., 431  
 Unknown mark setting type ..., 432  
 Unknown numbering type ..., 432  
 Unknown option ..., 427  
 Unknown toplevel for ..., 434  
 Unrecognized argument for  
 $\backslash$ sidecapmargin, 432  
 Unused global option(s) ..., 429  
 $\backslash$ upbracefill, 261  
 $\backslash$ uppercaseheads, 165, 170, 171  
 $\backslash$ uppermargin and/or  $\backslash$ textheight  
 and/or  $\backslash$ lowermargin are too  
 large ..., 432  
 $\backslash$ uppermargin (length), 82, 83, 431, 432  
 $\backslash$ upshape, 94  
 url (package), 395  
 Use of ... doesn't match ..., 422  
 $\backslash$ usebox, 309  
 $\backslash$ usepackage, 380, 381, 425, 427  
 $\backslash$ usepackage before  $\backslash$ documentclass,  
 427  
 $\backslash$ usethanksrule, 116  
 Utopia, 76

## V

$\backslash$ valign, 417, 419  
 $\backslash$ vbox, 405, 416, 420, 422  
 $\backslash$ vector, 424  
 veelo (chapterstyle), 144, 149, 150  
 Veelo, Bastiaan, 144, 282  
 $\backslash$ verb, 264, 266, 315, 343  
 $\backslash$ verb, 427  
 $\backslash$ verb ended by end of line, 427  
 $\backslash$ verb illegal in command argument, 427  
 $\backslash$ verb may be unreliable ..., 434

$\backslash$ verb\*, 266, 315, 316  
 verbatim, 309, 315–321  
 changing font, 315  
 font, 320  
 frame, 317, 323  
 styling, 317  
 in argument, 309  
 new, 319–321  
 short, 315  
 with tab spaces, 316  
 wrap long lines, 316  
 write, 325, 327  
 $\backslash$ verbatim, 316, 320, 321  
 verbatim (environment), 187, 294, 315–  
 317, 319, 320, 323, 397, 423, 432  
 verbatim (package), xix, 316, 380  
 verbatim\* (environment), 315–317  
 $\backslash$ verbatim@font, 321  
 $\backslash$ verbatim@line, 321  
 $\backslash$ verbatim@processline, 321  
 $\backslash$ verbatim@startline, 321  
 $\backslash$ verbatimbreakchar, 316  
 $\backslash$ verbatimindent (length), 316  
 $\backslash$ verbatiminput, 323  
 $\backslash$ verbatiminput\*, 323  
 verbatimoutput (environment), 322  
 $\backslash$ verbfootnote, 274  
 versal, 39–40  
 $\backslash$ versal, 398  
 verse, 293–308  
 centering, 295, 296  
 end a line, 293  
 end a stanza, 293  
 indent line, 294, 295  
 indent pattern, 295, 305  
 indent space, 296, 297  
 long line, 295, 296  
 long lines, 294  
 multiple indexes, 295  
 prevent page break, 295  
 typesetting environments, 293–295  
 wrapped line indent, 296  
 verse (environment), xviii, 293–296,  
 298–300  
 verse (package), xix, 380  
 $\backslash$ verselinebreak, 297, 305  
 $\backslash$ verselinenumbersleft, 300

`\verselinenumbersright`, 300  
`\versewidth` (length), 295, 296  
 verso, xxiii  
 vertical,  
     centering, 379  
 vertical mode, 405  
*verville* (chapterstyle), 144  
 Verville, Guy, 144  
`\vfil`, 410  
`\vfill`, 410  
`\vfilneg`, 410  
`\vgap` (length), 296, 297, 299  
`\vin`, 296  
`\vindent` (length), 296  
`\vinphantom`, 297  
`\vleftmargin` (length), 296  
`\vleftoffline`, 298  
`\vleftskip` (length), 300  
`\vline`, 267  
`vmnipage` (environment), 103  
`vplace` (environment), 379  
`\vrightskip` (length), 300  
`\vskip`, 410  
`\vss`, 410

## W

warning,  
     LaTeX, 427–429  
     memoir class, 432–434  
     TeX, 415–422  
`\wd`, 405  
 widow, 38  
     line, 104, 105  
`\width` (length), 310  
 Wilson, Adrian, 28  
*wilsondob* (chapterstyle), 144, 162  
*wilsondob* (headstyles), 161, 162  
*wrapfig* (package), 243  
`\wrappingoff`, 316  
`\wrappingon`, 316  
 write,  
     verbatim, 329  
`\write`, 422  
`writefigure` (environment), 327  
`writeverbatim` (environment), 322,  
     323, 327

## X

X columns too narrow (table too wide),  
     266  
 X columns too narrow ..., 434  
*xcolor* (package), 310, 395  
`\xdef`, 404  
`Xdepth` (counter), 214  
 XeTeX, 376, 432  
 XeTeX is required to process this docu-  
     ment, 432  
`\Xheadstart`, 195  
*xindy* (program), 344  
`\xindyindex`, 344  
`\xlvchars` (length), 75, 76  
`\Xmark`, 195  
*xtab* (package), xvii, 269

## Y

You can't use ... in ..., 422  
 You can't use '`\spacefactor`' in vertical  
     mode, 422  
 You can't use 'macro parameter character  
     #' in ... mode, 422  
 You have requested release ..., 429  
 You have requested version ..., 429  
 You have used the '\*pt' option but file ...,  
     432

## Z

`\z@`, 397, 406, 409  
*Zapf*, Hermann, 35  
`\zerotrivseps`, 186



---

## Index of first lines

---

Beautiful Railway Bridge of the Silv'ry Tay, 344

Fury said to, 308

His judgement rendered, he dissolved the Thing, 298

I am François, which is unfortunate, 269

I used to love my garden, 293, 303

In a cavern, in a canyon, 306

In mathematics he was greater, 305

Je suis François, dont il me pois, 269

Prince, do not ask in a week, 96

Prince, n'enquerez de sepmaine, 96

Then God created Newton, 304

There was a young lady of Ryde, 306

There was a young man of Quebec, 303

There was an old party of Lyme, 294

What a funny thing is a flea, 304



## 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.