The Memoir Class

The Memoir Class

for

Configurable Typesetting

User Guide

Peter Wilson

THP
The Herries Press

The Herries Press, Normandy Park, WA.

Printed in the World

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

10 09 08 07 06 05 04 03 02 01   15 14 13 12 11 10 9

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

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.

Contents

List of Figures

List of Tables

Preface

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

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

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

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

Peter Wilson
Seattle, WA
June 2001

Introduction to the first edition

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 listed in the Bibliography. Lamport [Lam94] is of course the original user manual for LaTeX, while the Companion series, for example [GMS94], go into further details and auxiliary programs. The Comprehensive TeX Archive Network (CTAN) is a valuable source of free information and the LaTeX system itself. For general questions see the FAQ [FAQ] which also has pointers to 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]. The question of how to use different fonts with LaTeX is left strictly alone; Alan Hoenig's book [Hoe98] is the best guide to this that I know of.

The first part of the manual discusses briefly some aspects of book design and typography, independently of the means of typesetting. Among the several books on the subject listed in the Bibliography I prefer Bringhurst's *The Elements of Typographic Style* [Bri92]. I have used the LaTeX book design, which is the default memoir class style, in typesetting Part I.

The second part then goes into some detail on how the memoir class can be used to implement a particular design.

With two exceptions, the memoir class has all the capabilities of the standard LaTeX book class. These exceptions are:

- The old LaTeX v2.09 font commands — `\rm` (roman), `\tt` (`typewriter`), `\sf` (sans), `\bf` (**bold**), `\sl` (*slanted*), `\it` (*italic*), and `\sc` (SMALL CAPS) — are not supported and will give error messages if used.

  The `\em` (*emphasis*) command is supported but gives a warning message if used.

- There are no commands for making titles. This is because title pages are usually designed individually for each book. The titling package [Wil01g] can be used if you want the titling commands. The package provides extended titling facilities when compared to those in the standard LaTeX classes.

I hope that, apart from these, the class supports everything that the book class provides.

The major extra functions provided by the class include:

- Font sizes for the main text of 9, 10, 11, 12 and 14pt.

- A reasonably intuitive means of setting the page, text and margin sizes.

- Page trimming marks.
- If really required, typesetting as though in the olden typewriter days (double spaced, raggedright, no hyphenation, typewriter font).
- Configurable sectional headings, with several predefined styles for chapter headings and simple methods for defining new ones.
- 'Anonymous' section breaks (e.g., a blank or decorated line or two).
- Configurable page headers and footers, with several predefined styles and simple methods for defining new ones.
- Configurable captions, with several predefined captioning styles and methods for defining new ones.
- Ability to define new 'List of...' and new floats with their accompanying captions and 'List of...'.
- Control over whether the 'List of...', bibliography, index, etc., appear in the Table of Contents.
- Support for epigraphs.

Also, along the way you will find other more minor but still useful things.

As Part II progresses I demonstrate some of the changes that the memoir class lets you easily make to the normal LaTeX page and titling style.

Type conventions

The following conventions are used in the manual:

- 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.`

Caveats

At the moment both this manual and the class code are in a beta state. That is, they hopefully serve the purposes they are intended for, but it is probable that there are errors, poor explanations and missing elements. So, be warned that I'm sure that there will be further releases and these may not be entirely compatible with the current release.

That said, I will be grateful for any constructive comments that anyone[1] may have, especially regarding errors, mispeaking, and desireable enhancements. I can be reached either by posting to `comp.text.tex` or more directly at `peter.r.wilson@boeing.com`.

[1]I have received valuable comments from Javier Bezos (`jbezos@wanadoo.es`), Sven Bovin (`sven.bovin@chem.kuleuven.ac.be`), Scott Pakin (`pakin@uiuc.edu`), and Paul Stanley (`pstanley@essexcourt-chambers.co.uk`) .

TeX 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 arbitrary. Most non-technical books are typeset on a fixed grid as they do not have arbitrary insertions into the text.

TeX is designed to handle arbitrary sized inserts in an elegant manner, and does this by allowing vertical spaces to stretch and shrink a little so that the actual height of the typeblock is constant. Therefore LaTeX, being built on top of TeX, does not typeset on a fixed grid, and it would be a very major task to try and make it do so; this has been tried but as far as I know nobody has been successful.

The manual includes many unbreakable names of LaTeX commands, some of which stick out into the margin. The way of getting rid of these is to rewrite the text so that they don't come at the end of a typeset line. This is tedious and I haven't done it because I expect the manual to be revised and that would throw off any hand tweaking done now.

Introduction to the second edition

Since the first edition of the manual was published the memoir class has undergone some changes and extensions. The changes, to remove some unfortunate errors, are upwards compatible. The extensions, by their nature, are not upward compatible.

The main extensions and changes include:

- A subfigure option to counteract an unfortunate interplay[2] if the subfigure package is used with the class.
- An article option so that article class typesetting may be simulated.
- Incorporation of the essential code from the titling package [Wil01g] (to support the article option).
- Incorporation of the essential code from the abstract package [Wil01a] (to support the article option).

The description of how to modify the ToC, LoF and LoT headings in the first edition was completely wrong, as was the corresponding description of the `\newlistof` macro. No noticeable changes have been made to the class code but the descriptions now reflect reality. I must have been a few bricks short of a full load when I wrote the original.

There are other more minor changes and extensions[3] which you may find if you recall the first edition.

There was no mention of typesetting verse in the first edition, although the class does provide the normal LaTeX `verse` environment. A poem is usually individually typeset as the appearance often has an effect on the emotional response when reading it. The verse package [Wil01k] may be useful when typesetting poetry.

[2]Discovered by Ignasi Furió Caldentey (`ignasi@ipc4.uib.es`).

[3]With thanks to, among others, Kevin Lin (`kevinlin@runtop.com.tw`) and Adriano Pascoletti (`pascolet@dimi.uniud.it`).

Introduction to the third edition

Since the second edition of the manual was published the memoir class has been upgraded from beta code to a production release. Extensions have been made to both the class and this manual.

The main extensions and changes include:

- An openleft option to enable chapters to start on verso pages.
- Incorporation of the essential code from the verse package [Wil01k] for more flexibility when typesetting poetry.
- Replacement of the macro called `\makepshook` by `\makepsmarks`. Note that this is a non-upward compatible change.
- The first part of the manual has been reorganised and extended, principally by providing more typesetting examples.
- As usual, minor glitches have been removed from both the code and the manual; each revision, of course, eliminates the gap between advertisement and reality.

There are other more minor changes and extensions[4] which you may find if you recall the second edition.

[4]With thanks to, among others, Ignasi Furió Caldentey (`ignasi@ipc4.uib.es`), Daniel Richard G. (`skunk@mit.edu`) and Vladimir Ivanovic (`vladimir@acm.org`).

Introduction to the fourth edition

Since the third edition of the manual was published the memoir class has been upgraded from version 1.0 to version 1.1. Modifications have been made to both the class and this manual.

The main extensions and changes include:

- The subfigure option is no longer required.
- Subfloats have been added to the class. Steve Cochran kindly gave permission for me to use some of his subfigure package code for this.
- Some packages still use the old, deprecated LaTeX version 2.09 font commands. I have reluctantly introduced an option to enable the old, deprecated font commands to be used.
- The class now works harmoniously with the natbib package [Dal99].
- As usual, minor glitches have been removed from both the code and the manual; each revision hopefully eliminates the gap between advertisement and reality.

There are other more minor changes and extensions[5] which you may find if you have used earlier editions.

[5]With thanks to, among others, William Adams (`WillAdams@aol.com`) Ignasi Furió Caldentey (`ignasi@ipc4.uib.es`), Steven Douglas Cochran (`sdc+@cs.cmu.edu`), Henrik Holm (`henrik@tele.ntnu.no`), and Rolf Niepraschk (`niepraschk@ptb.de`).

Introduction to the fifth edition

Since the fourth edition of the manual was published the memoir class has been upgraded from version 1.1 to version 1.2. Modifications have been made to both the class and this manual.

The main extensions and changes include:

- The size options have been extended[6] to include a `17pt` option.
- A few font sizes corresponding to the font size commands (e.g., `\Large`) have been changed to give regular steps between sizes.
- Boxes that can break over pages and/or contain verbatim text.
- Several ways of dealing with verbatim text, including footnotes that can contain verbatim text.
- Some control over the typesetting of footnotes. Unfortunately this necessitated some changes in the methods for styling thanks notes.
- Comment environments.
- Convenient methods for file input and output.
- Additional `\provide...` commands.
- The `description` environment has been modified to match the appearance of the standard environment. The original memoir form is still available as the `blockdescription` environment.
- A new optional argument has been added to the `\chapter` and `\chapter*` commands for setting header texts.

As usual, minor glitches have been removed from both the code and the manual. Hopefully, new ones have not been introduced.

[6]At the request of Vittorio De Martino whose children use LaTeX for their school projects.

Introduction to the sixth edition

Since the fifth edition of the manual was published the memoir class has been upgraded from version 1.2 to version 1.6. Many new functions have been added to the class and this manual has been updated to reflect all the additions.

The main extensions and changes include:

- Major extensions for typesetting arrays and tabulars, including continuous tabulars and automatic tabulation.
- Major extensions to footnote styles and the ability to have multiple series of footnotes.
- Major extensions for indexing, including one column and multiple indexes.
- Major extensions to crop marks.
- `\tableofcontents` and friends can be used multiple times.
- Section titles (as well as numbers) can be referenced.
- Sheet numbers and access to the numbers of the last sheet and last page.
- Various methods for formatting numbers.
- Better cooperation with the chapterbib and natbib packages.

There are many more minor additions. As usual, glitches have been removed from both the code and the manual. Hopefully, new ones have not been introduced.

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, Donald Arseneau, Jens Berger, Karl Berry, Javier Bezos, Sven Bovin, Alan Budden, Ignasi Furió Caldenty, Ezequiel Martín Cámara, David Carlisle, Steven Douglas Cochran, Michael Downes, Victor Eijkhout, Danie Els, Robin Fairbairns, Simon Fear, Kai von Finkel, Daniel Richard G, Romano Giannetti, Kathryn Hargreaves, Sven Hartrumpf, Florence Henry, Cartsten Heinz, Peter Heslin, Morton Høgholm, Henrik Holm, Vladimir Ivanovich, Stefan Kahrs, Jøgen Larsen, Kevin Lin, Matthew Lovell, Daniel Luecking, Lars Madsen, Vittorio De Martino, Frank Mittelbach, Rolf Niepraschk, Patrik Nyman, Heiko Oberdiek, Scott Pakin, Adriano Pascoletti, Robert, Chris Rowley, Bernd Raichle, Doug Schenck, Rainer Schöpf, Paul Stanley, Reuben Thomas, Bastiaan Niels Veelo, Emanuele Vicentini, Jürgen Vollmer, and others.

If I have inadvertently left anyone off the list I apologise, and please let me know so that I can correct the omisssion.

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.

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 give the font size and leading separated by a slash; for instance 10/12 for a 10pt font set with a 12pt leading, or 12/14 for a 12pt font set with a 14pt leading.

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

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

Table 1: 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

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 1 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 is a recent invention. It tends to be used in page markup languages, like PostScript[7], in order to make to make calculations quicker and easier.

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

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

Part I

Art and Theory

Chapter 1

The Parts of a Book

1.1 Introduction

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

1.2 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 differences 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 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. The first few pages in the front matter are not paginated and so do not have folios. The remainder are paginated and the folios are usually expressed as roman numerals. Not all books have all the elements described below.

The first page is a recto *half 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.

Table 1.1: Front matter

Element	Page	Paginated	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.

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

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 editions[1] 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.

1.3 Main matter

The main matter forms the heart of the book.

All pages within the main matter are paginated, 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.

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

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. The chapters may be grouped within *parts* which would then be the highest level of division within the book. Typically both 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. Chapters also start on recto pages but in this case the text of the chapter starts on the same page as the chapter title.

Chapters may be divided into sections, either numbered or unnumbered, with the numbering scheme starting afresh within each chapter. Similarly sections may be partitioned into subsections 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 subdivisions 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 front 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.

1.4 Back matter

The back matter is optional but if present conveys information ancilliary to that in the main matter.

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.

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, *4to*	2	$a/2 \times b/2$	2	4	8
Octavo, *8vo*	3	$b/4 \times a/2$	4	8	16
16mo	4	$a/4 \times b/4$	8	16	32
32mo	5	$b/8 \times a/4$	16	32	64
64mo	6	$a/8 \times b/8$	32	64	128

1.5 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 *signature* of several smaller sheets. An unfolded sheet is called a *broadside*. Folding a sheet in half produces a one 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 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 trimmed from the top, bottom and foredge 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.

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.

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 1/20 th 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'.

Chapter 2

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 availabilty 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 $11 \times 8\ 1/2$ inches. Similarly, one of the standard sizes for a business envelope is $4\ 1/8 \times 9\ 1/2$ 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.

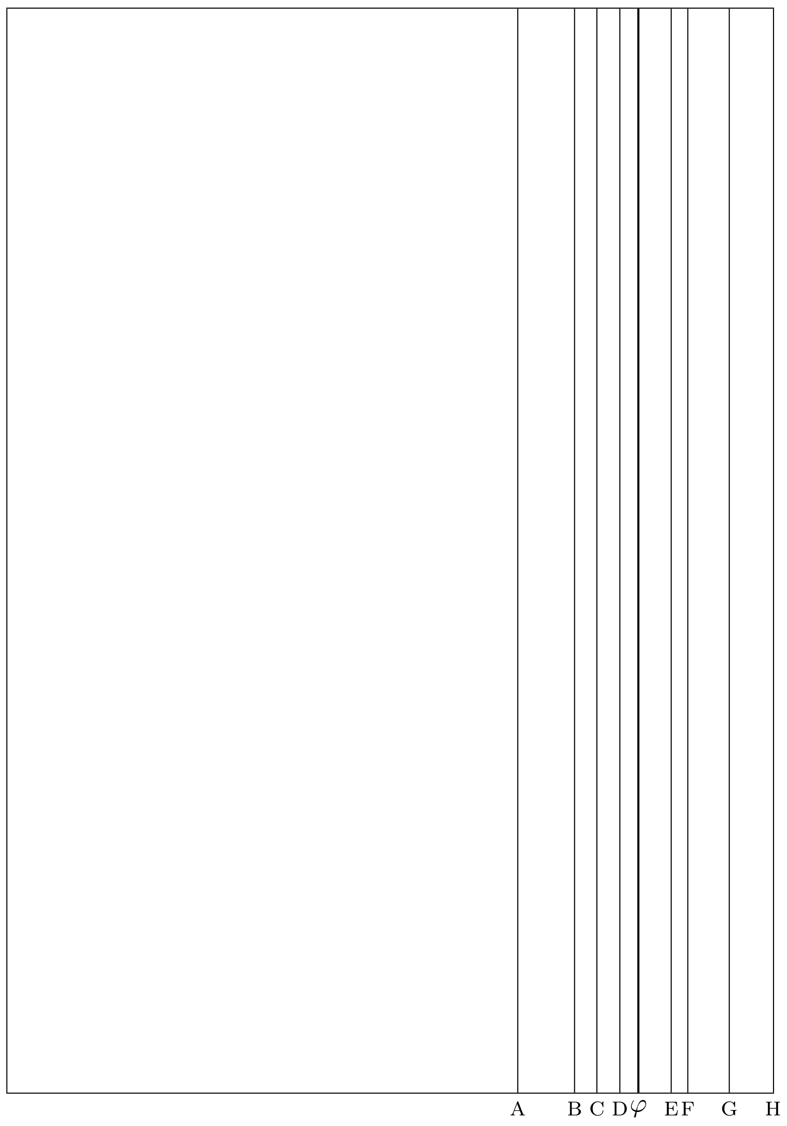
Over the years books have been produced in an almost infinite variety of proportions, where by *proportion* I mean the ratio of the height to the width of a rectangle. However, certain proportions occur time after time throughout the centuries and across many different countries and civilizations. This 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). These 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. It is an interesting fact, though, that it is impossible to fold a sheet of paper, no matter how large and thin, more than six times altogether. 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 property to those of medieval times. However, in this case each fold reproduces the 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), and A5 sheets.

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.



A $2:1$	D $5:3$	F $3:2$
B $9:5$	φ $1.618:1$ ($\varphi:1$)	G $1.414:1$ ($\sqrt{2}:1$)
C $1.732:1$ ($\sqrt{3}:1$)	E $1.538:1$	H $4:3$

Figure 2.1: Some page proportions

In books where the illustrations are the primary concern, the shape of the illustrations is 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 accomodated on a 0.83 : 1 page.

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

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

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 Pythagoran 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.

$$\varphi + 1 = \varphi^2 \tag{2.2}$$
$$\varphi - 1 = 1/\varphi \tag{2.3}$$

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

It also has a very simple definition as the continued fraction

$$\varphi = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cdots}}}} \tag{2.4}$$

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 numbers[3]:

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

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 $(= 1.618\ldots)$, 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, striking relationship between φ and the Fibonacci series. Define the Fibonacci numbers as F_n, where

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

Then

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

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 paterns (for example see [CG96]). Martin Gardener [Gar66] reports on studies that claimed that the average ratio of a person's height to the height of the navel is 1.618+ — suspiciously close to φ.

[2]Book of the Abacus.

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

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 head and/or foot 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)

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		2.9 left
$3/2$	1.701	$w/9$	s	$2s$	$7s/3$		2.9 right
$3/2$	$\pi/2$	$w/13$	$2s$	$10s/3$	$30s/7$		2.10 left
$3/2$	$3/2$	$w/9$	$3s/2$	$2s$	$3s$		2.10 right
$3/2$	1.68	$w/23$	$2s$	$5s$	$2s$		2.11 left
$3/2$	$3/2$	$w/10$	$2s$	$5s/2$	$2.85s$		2.11 right
1.48	1.376	$w/12$	$7s/4$	$2s$	$7s/2$		2.12 left
$13/9$	$\sqrt{2}$	$w/30$	$2s$	$9s/2$	$4s$	$s/2$	2.12 right
$\sqrt{2}$	φ	$w/9$	s	$2s$	$2s$		2.13 left
$\sqrt{2}$	φ	$w/8$	s	$5s/3$	$5s/3$		2.13 right
$7/5$	1.641	$w/7$	s	$8s/5$	$8s/5$		2.14 left
1.294	φ	$0.176w$	$1.03s$	$1.685s$	$13s/9$		2.14 right
1.294	$13/9$	$w/12$	s	$2s$	$10s/7$	$s/2$	2.15 left
$9/7$	$19/9$	$2w/5$	$5s/8$	$5s/8$	$5s/6$		2.15 right
$5/4$	$13/11$	$w/10$	$3s/2$	$2s$	$8s/3$		2.16 left
$7/6$	$17/15$	$w/13$	s	s	$7s/5$	.382	2.16 right
e/π	0.951	$w/9$	s	$2s$	$3s/2$		2.17 left
$5/7$	$2/3$	$w/9$	$s/2$	$2s/3$	s	$s/3$	2.17 right

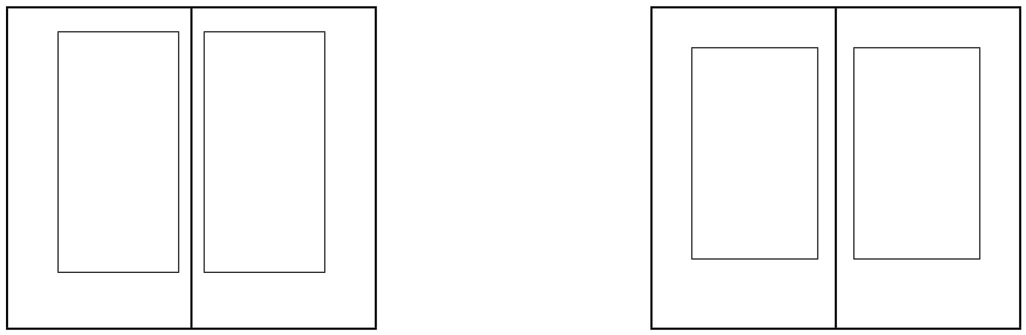


Figure 2.2: Two spreads: (Left) Canada, 1992. (Right) England, 1970.

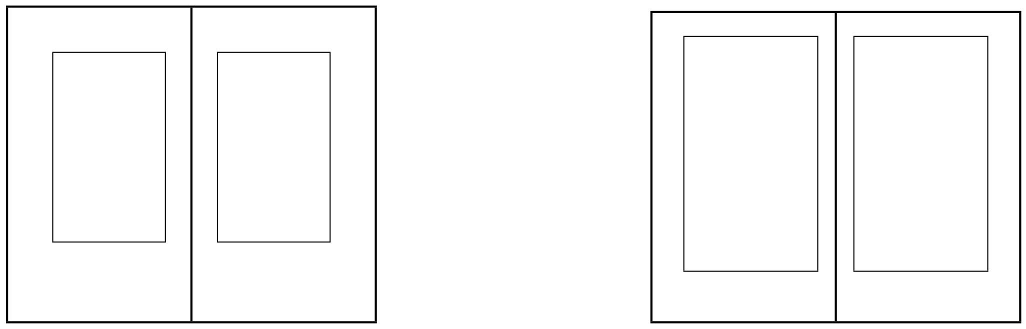


Figure 2.3: Two spreads: (Left) USA, 1909. (Right) England, 1964.

f = foot margin (bottom margin)

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

The designs are also shown in Figures 2.2 to 2.17. 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 Syntax. The original size is 227×132mm. 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 216×125mm and is set in $12/13 \times 22$ Centaur.

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 184×107mm. On the right is another book from the

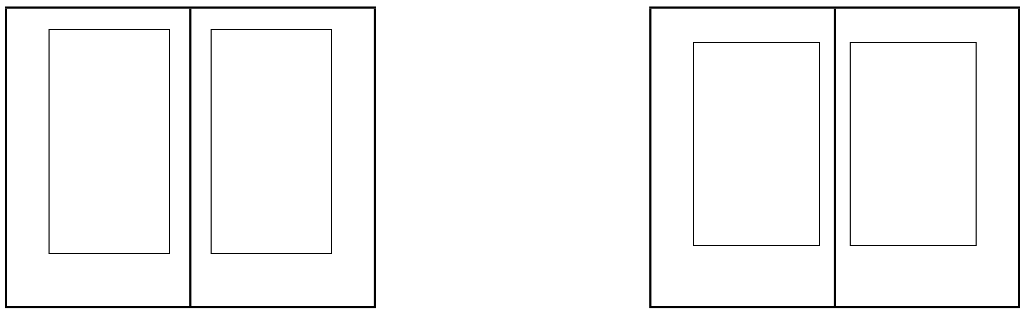


Figure 2.4: Two spreads: (Left) France, 1559. (Right) Canada, 1995.

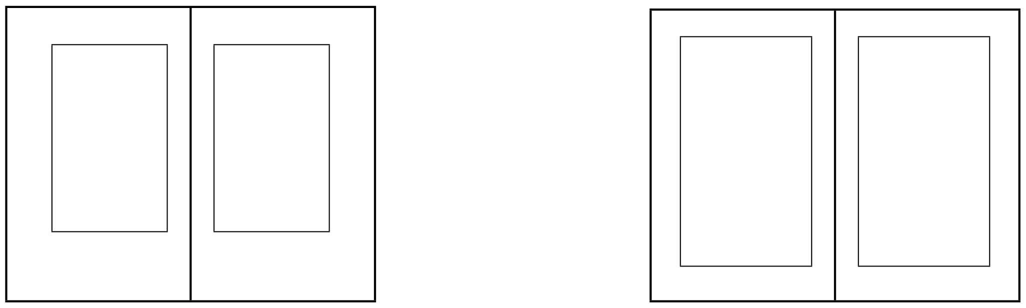


Figure 2.5: Two spreads: (Left) USA, 1949. (Right) USA, 1990.

Folio Society — *Three Men in a Boat* by Jerome K. Jerome printed in 1964. The original size is 215 × 128mm and is typeset with Ehrhardt at 11/12 × 22.

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 × 23 on a page size of 231 × 143mm.

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 × 22 on a page size of 240 × 150mm. 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 × 150mm.

Microcosmographica Academia by F. M. Cornford is shown in Figure 2.6. Despite its

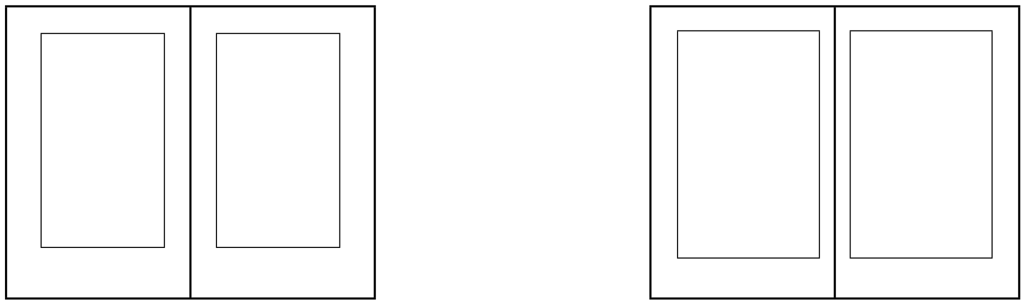


Figure 2.6: Two spreads: (Left) England, 1908. (Right) USA, 1993.

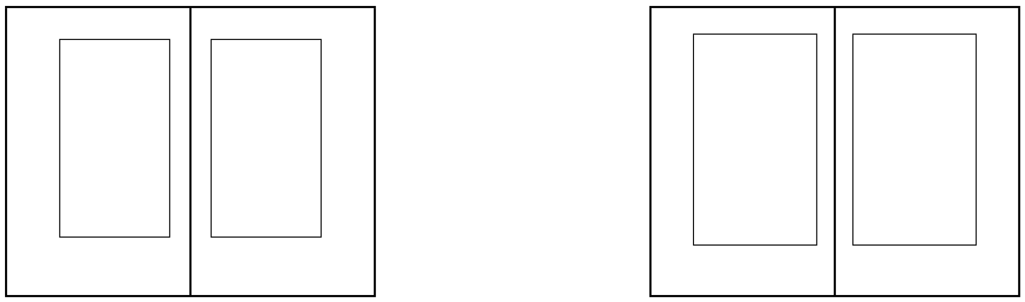


Figure 2.7: Two spreads: (Left) USA, 1931. (Right) England, 1968.

title, it is written in English and was published by Bowes & Bowes, London, in 1908. It is a dryly humourous look at academic politics as practised in Cambridge University at the turn of the nineteenth century (possibly the twentieth as well). It is set with 14pt leading on 22pc. The original page size is 216×136mm. The right of this figure illustrates a book with another unusual title — *The Alphabet Abcedarium* 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 227.5×150mm. The book gives a history of each letter of the latin alphabet. One unusual feature is that there is a deep footer on each page showing many examples of typefaces of the letter being described.

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 231×147mm. 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. 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. This is typeset in Monotype Barbon with 17.5 leading on a 26pc measure on a 253×162mm page.

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 233×150mm. Ruari McLean's *The Thames and Hudson Manual of Typography* (1988)

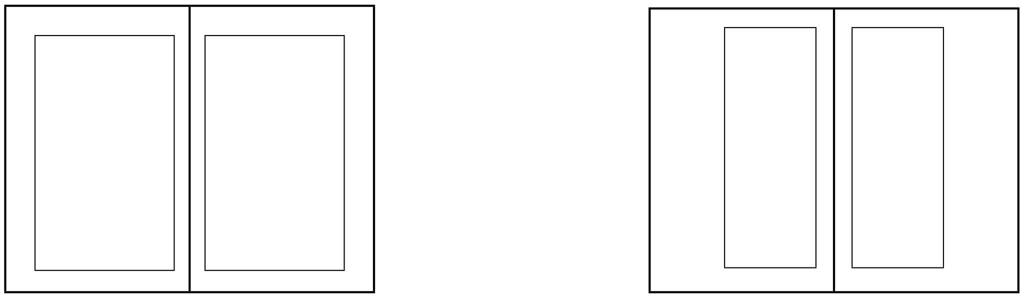


Figure 2.8: Two spreads: (Left) USA, 1994. (Right) England 1988.

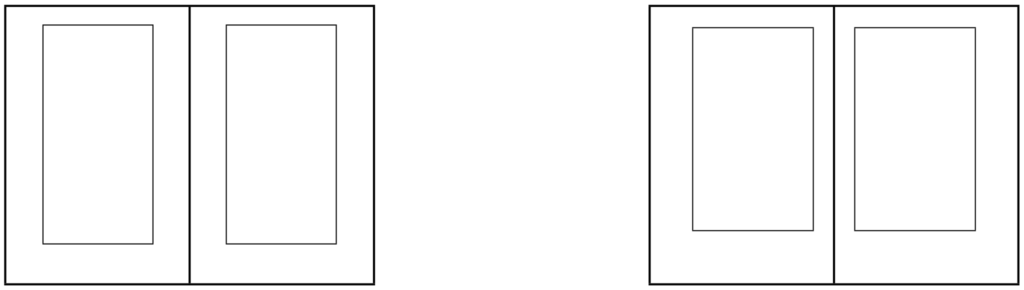


Figure 2.9: Two spreads: (Left) Italy, 1523. (Right) Italy 1499.

is at the right in Figure 2.8. This is typeset in 10/11 × 20 Monophoto Garamond on a 240 × 156mm page. The wide foredge is used for small illustrations. Notes are also set in this margin rather than at the foot of the page.

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 2.9 (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 2.9 (right) is also a simple 3 : 2 ratio. The proportions of the typeblock, being 1.7, 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 Barramada, 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 Book and Manuscript Library at Yale is shown at the left of Figure 2.10. 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 Girosle* (Navigation and discovery of Upper India and

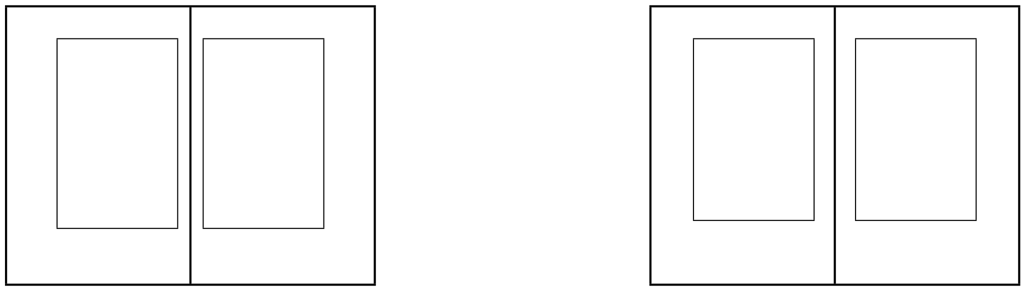


Figure 2.10: Two spreads: (Left) France/Portugal, 1530. (Right) Gutenberg, C15th.

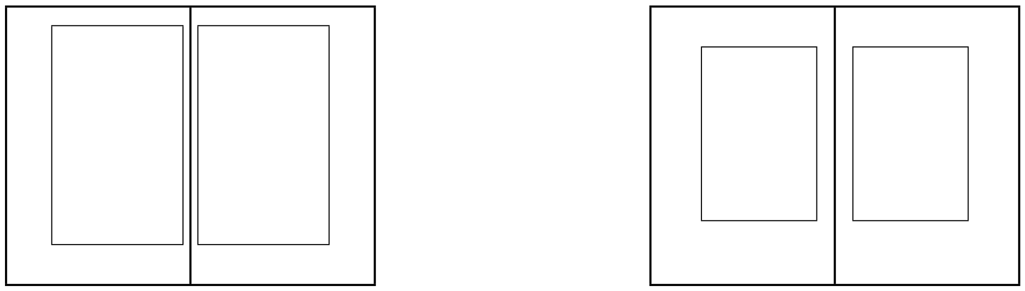


Figure 2.11: Two spreads: (Left) Persia, 1525. (Right) USA, 1975.

the Isles of Molucca where the cloves grow) is written in a beautiful humanistic miniscule. There are 27 lines to a page, which is 286×190mm and made of vellum. The text measure is 29.5 and the 'leading' is 21pt. The wide fore 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 2.10 (right). This set of proportions was also often used in medieval incunabula[4] 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 2.18.

Two versions of the same publication are shown in Figure 2.11. On the left is a Persian manuscript *Khamsch of Nizami* written about 1525. The page size is about 324×216mm. 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 300×200mm, slightly smaller than the original but in the same proportions. The typeblock is 32pc wide and the type is set with a 15pt leading.

Frederic Goudy was a prolific American type designer. Shown at the left of Figure 2.12 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

[4]Early books, especially those printed before 1500.

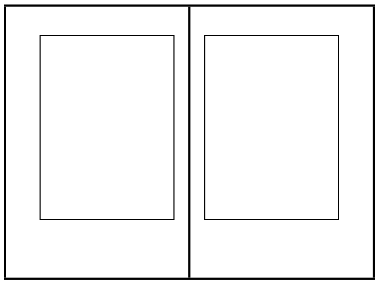
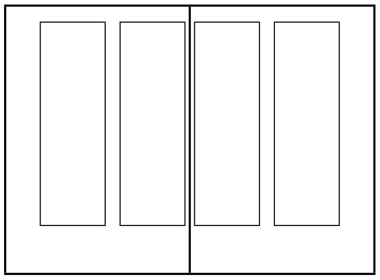



Figure 2.12: Two spreads: (Left) USA, 1952. (Right) England, 1087.

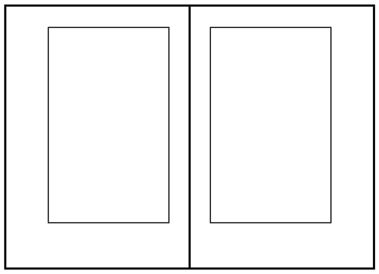
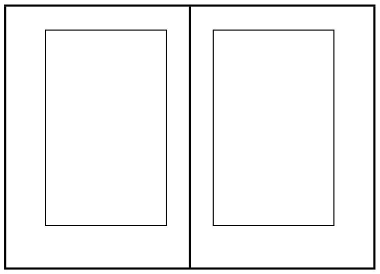



Figure 2.13: Two spreads: (Left) ISO (1). (Right) ISO (2).

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 2.12 (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 Caroline miniscule 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 2.13 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 Printer's Christmas books is at the left of Figure 2.14. 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 295×210mm with a measure of 31pc set with 15pt leading. 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 2.15. 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 other layout in this figure is B. W. Robinson's

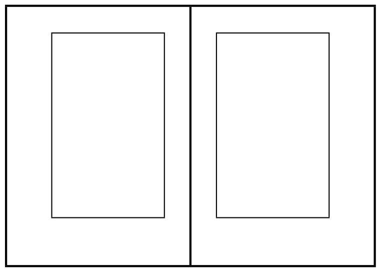
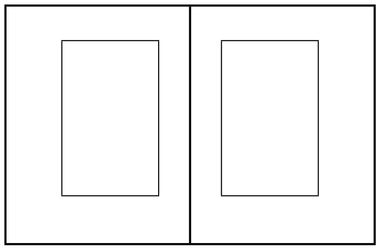



Figure 2.14: Two spreads: (Left) England, 1973. (Right) LaTeX 10pt book style.

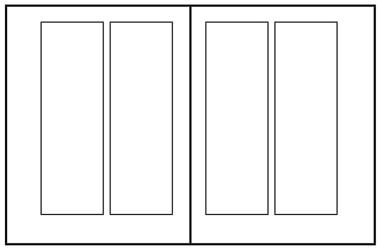
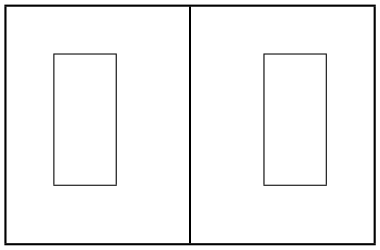



Figure 2.15: Two spreads: (Left) USA, 1967. (Right) England, 1982.

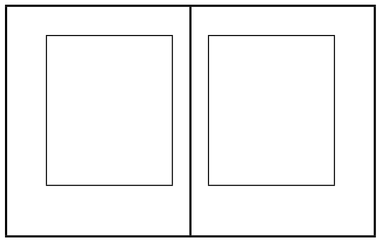
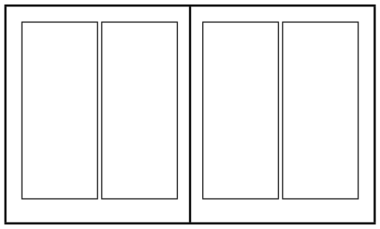



Figure 2.16: Two spreads: (Left) England, 1972. (Right) Switzerland, 1980.

Kuniyoshi: The Warrior Prints published by Phaidon, Oxford in 1982. The page size is 310×242mm 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 195×150mm and is illustrated on the left of Figure 2.16. The amount of text on a page varies and there are many line drawings, some of which take a double spread. 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 280×240mm, considerably larger than its companion in the figure, yet with much smaller margins.

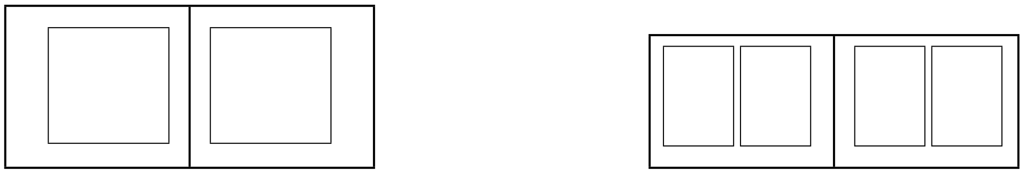


Figure 2.17: Two spreads: (Left) England, 1969. (Right) USA, 1989.

Two more layouts for illustrated books are given in Figure 2.17. 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. Shown on the right of Figure 2.17 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.

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 Gutenberg's books [Tsc91, pages 44–57], which is shown in Figure 2.18. 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 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 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

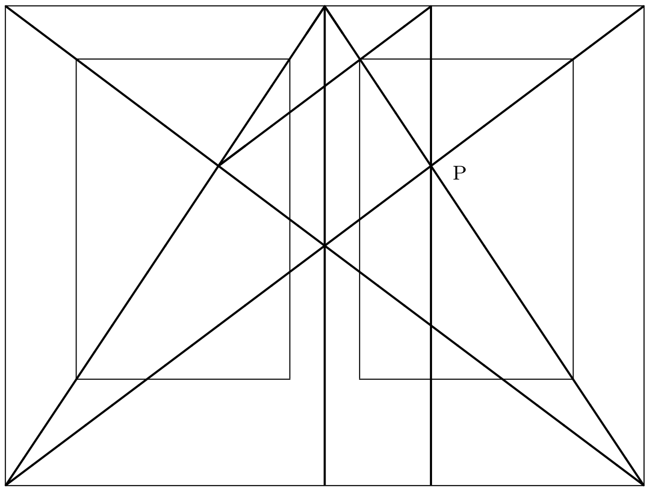


Figure 2.18: The construction of the Gutenberg page design

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 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 [Bri92] 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 Bringhurt's copyfitting table. For example, it suggests that a font with a length of 130pt should be set on a measure of about 26pc for a single column or in an 18pc wide column if there are multiple columns.

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.

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

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.

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, but this is usually under the control of the designer. There may be vertical rivulets 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; this not only will cause a rivulet 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.

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.

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 *miniscules*. 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. 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 read[5] 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 serifed 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

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

Ill

does it stand for 'Ill', '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 emphasing 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.

   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-seriffed fonts

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

Schriver [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 serifed typefaces for continuous reading, but for titling the choice is wide open.

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. As Robert Bringhurst said, 'A widow has a past but no future'. Typographically, widows should be avoided. 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.

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. Bringhurt's memory trick for orphans is, 'An orphan has a future but no past'.

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 this[6]

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

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 should not be indented. 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.

IT IS EASIER to start a paragraph with a raised capital than one that is dropped. 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 accomodate 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.

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

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.

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 document, except perhaps the title and half title pages in a book, should be 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 and rear 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.

From now on 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 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 must blend with the style used for the typeblock.

Chapter 3

Picky points

3.1 Introduction

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

3.2 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 3.1 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. Figure 3.2 illustrates some text typeset with different line spacings. The normal interword spacing is used in the samples.

3.3 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 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 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 3.1: Interword spacings

This paragraph is set solid — the interline spacing is the same as the font size. The normal interword spacing is used. 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 paragraph is set with the normal interline spacing for the font. The normal interword spacing is used. 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 paragraph is set with the interline spacing 20% greater than is normal for the font. The normal interword spacing is used. 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 3.2: Interline spacings

Acroynms 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 undesireable, 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.

3.4 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. [*s*ituation *n*ormal — *a*ll *f*——d *u*p.]

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.

3.5 Punctuation

Quotation marks

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

‘There’s glory for you!’
‘I don’t know what you mean by “glory” ’, Alice said.
Humpty Dumpty smiled contemptuously. ‘Of course you don’t — till I tell you. I meant “there’s a nice knock-down argument for you”!’
‘But “glory” doesn’t mean “a nice knock-down argument” ’, Alice objected.
‘When *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’.

“There’s glory for you!”
“I don’t know what you mean by ‘glory,’ ” Alice said.
Humpty Dumpty smiled contemptuously. “Of course you don’t — till I tell you. I meant ‘there’s a nice knock-down argument for you!’ ”
“But ‘glory’ doesn’t mean ‘a nice knock-down argument,’ ” Alice objected.
“When *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.”

Figure 3.3: Quotation marks: top English, bottom American

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 3.3 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 3.3 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.

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 means[1] 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 one[2], which is set off within commas, or when the note refers to a complete sentence.[3]

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 positions[4] 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 xxii. I tried various schemes in identifying 'PostScript' as being a registered trademark of Adobe Systems Incorporated. Among the discarded trials were:

> ...languages like PostScript$^{\mathrm{TM}}$, 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 PostScript[5], 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.

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

> **memoir,** *n.* ...

instead of

> **memoir**, *n.* ...

[1]Except to some I know.
[2]I hope that this is a phrase.
[3]Is this mark in the correct place?
[4]Marks 2 and 3.
[5]PostScript is a registered trademark of Adobe Systems Incorporated.

3.6 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 ragged right. Ragged right 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.

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 ragged right 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.

3.7 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 w o r d being e m p h a s i s e d, as has been done twice in this sentence. As an aside, for me at least, that extra spacing 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 aproaches: 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.

3.8 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.

3.9 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.

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

Chapter 4

Electronic books

4.1 Introduction

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.

This brief chapter includes some suggestions for the layout of Ebooks, based on my experience with 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.

4.2 Observations

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.

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.

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 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.

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 therefor 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 frontmatter 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 after I have tried to print the page.

Part II

Practice

Chapter 5

Starting off

This chapter uses the *headings* pagestyle; pagestyles are described in §14.

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 are a range of stock paper sizes from which to make a selection. These include:

a3paper	for a stock size of 420 × 297 millimeters
a4paper	for a stock size of 297 × 210 millimeters
a5paper	for a stock size of 210 × 148 millimeters
a6paper	for a stock size of 148 × 105 millimeters
b3paper	for a stock size of 500 × 353 millimeters
b4paper	for a stock size of 353 × 250 millimeters
b5paper	for a stock size of 250 × 176 millimeters
b6paper	for a stock size of 176 × 125 millimeters
letterpaper	for a stock size of 11 × 8.5 inches
legalpaper	for a stock size of 14 × 8.5 inches
executivepaper	for a stock size of 10.5 × 7.25 inches
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.

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

5.2 Type size options

The class offers a wider range of type sizes than usual. These are:

9pt	for 9pt type
10pt	for 10pt type
11pt	for 11pt type
12pt	for 12t type
14pt	for 14pt type
17pt	for 17pt type

These options are mutually exclusive. The default type size is 10pt.

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 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 succeding lines indented by `\bibindent` (the default is for an entry to run continuously with no indentations).
article	typesetting simulates the article class, but the `\chapter` command is not disabled. 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]{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). This may have a deleterious affect on the color of some pages. Serendipitously this has happened on page 9 where there is additional space between the paragraphs (caused by the next sectional division having to be put at the top of the next page). You may wish to compare that page with the following one to see the difference in the colors.

I could have made the page run short by inserting `\raggedbottom` at an appropriate place, followed later by a `\flushbottom`.

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.

Chapter 6

Laying out the page

This chapter is 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.

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 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 usually 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. For our purposes we only consider desktop publishing.

6.3 The page

We only consider one page per piece of stock.

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 foredge 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. Figure 6.2 shows the stock for a recto page, with a page layout, illustrating the main layout parameters. These may be changed individually by `\setlength` or by using the commands described below.

In the code for the standard classes it says:

> 'The variables `\paperwidth` and `\paperheight` should reflect the physical paper size after trimming. For desk printer output this is usually the real paper size since there is no post-processing. Classes for real book production will probably add other paper sizes and additionally the production of crop marks for trimming.'

This class has introduced the additional lengths `\stockwidth` and `\stockheight` to denote the physical paper size *before* trimming.

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⟩}
\stockheight \stockwidth
```

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 spread; when you open a book in the middle what you see is a spread — a verso page on the left and a recto page on the right

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

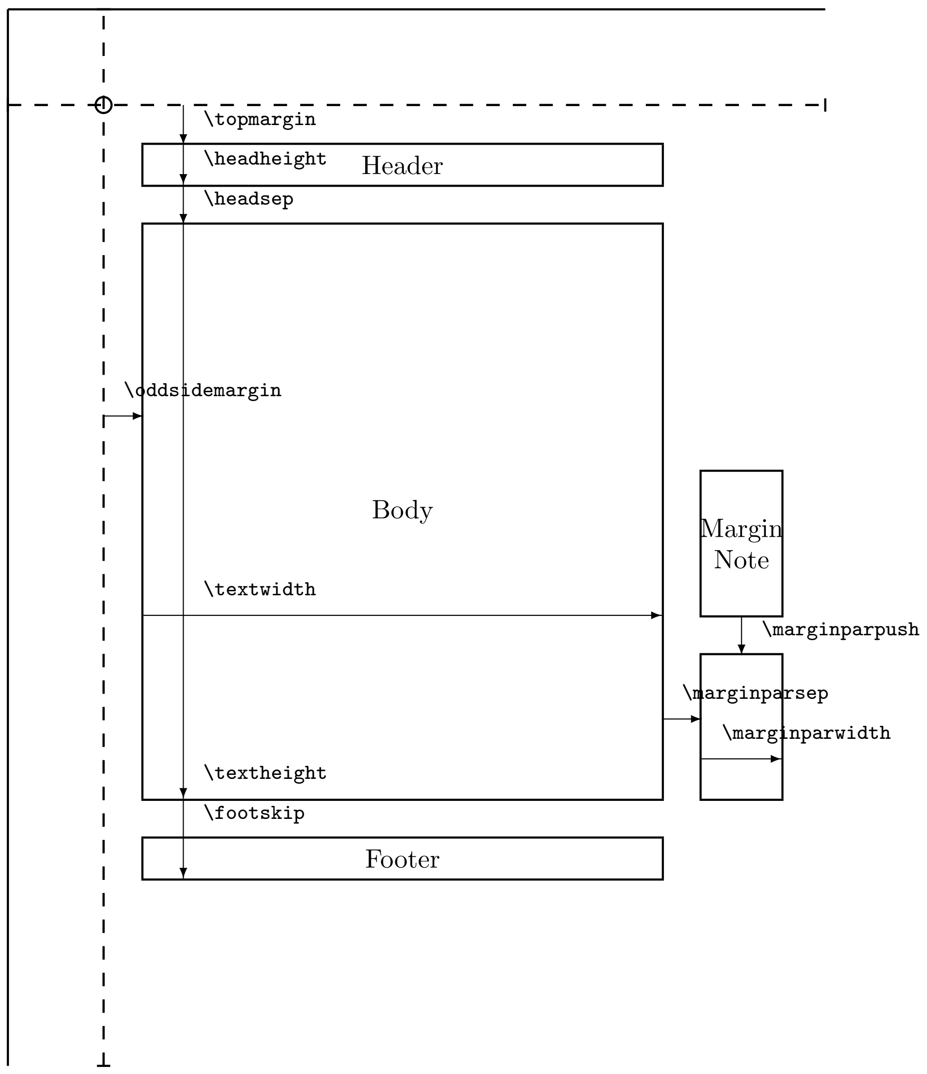


Figure 6.1: LaTeX page layout parameters for a recto page

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

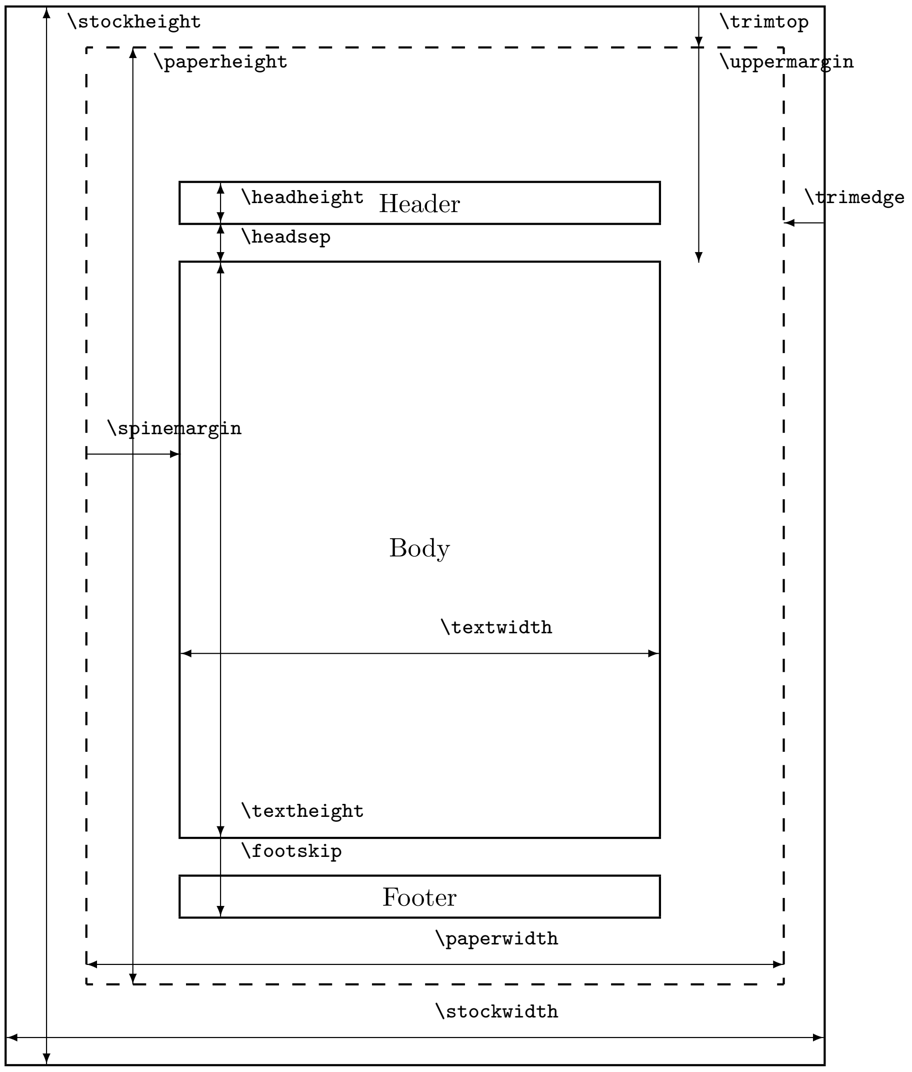


Figure 6.2: The main memoir class page layout parameters for a recto page

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⟩}
\paperheight \paperwidth
```

The command `\settrimmedsize` can be used to specify the height and width of the page (after any trimming). Initially the page size is made the same as the stock size, as set by the paper size option. 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.

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

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

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

```
\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 foredges of the pages and at the bottom, otherwise the book would be unopenable and unreadable.

```
\settrims{⟨top⟩}{⟨edge⟩}
\trimtop \trimedge
```

The command `\settrims` can be used to specify the amount intended to be trimmed from the top (⟨*top*⟩) and foredge (⟨*edge*⟩) 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.

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

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

specifies trimming 2in from the top and foredge 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
\addtolengh{\trimtop}{-\paperheight}  % \trimtop = \stockheight - \paperheight
\setlength{\trimedge}{\stockwidth}
```

Table 6.1: Length of CMR lowercase alphabets

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

```
\addtolength{\trimedge}{-\paperwidth}
```
which will set all the trimming to be at the top and foredge. If you wanted, say, equal trims at the top and bottom you could go on as
```
\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 6.1 gives the lowercase alphabet lengths for a range of Computer Modern Roman font sizes; this may be used in conjunction with Table 2.2 on page 25 when deciding on an appropriate textwidth.

```
\xlvchars \lxvchars
```

Based on this table, the two lengths `\xlvchars` and `\lxvchars` are approximately the lengths of a line of text with 45 or 65 characters, respectively, for the type size selected for the document.

If you are using a different font 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} % in log file
\endgroup      % end the grouping
```

The `\typeout` macro will print the result to the terminal and the log file.

Morten Høgholm has done some curve fitting to the data in Table 2.2. He found that the expressions

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

and

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

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

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

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

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

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

are: `\lxvchars` = 293.93684pt, and `\xlvchars` = 179.56801pt.

Continuing on this theme, Morten also wrote:

> ...I was defining some environments that had to have `\parindent` as their indentation. For some reason I just wrote `1.5em` instead of `\parindent` because I knew that was the value. What I had overlooked was that I had loaded the mathpazo package [Pug02], thus altering various `\fontdimens`. Conclusion: the environment would insert 1.5 em = 18.0 pt, whereas the `\parindent` was only 17.6207 pt.
>
> This, and other related situations can be avoided if one places
>
> `\RequirePackage{⟨font-package⟩}\normalfont`
>
> before `\documentclass`, but I have to this day never seen this suggested. I would believe that most document classes have settings that depend on *current* font settings, which they should do for such things as `\parindent`. However the decision to let Computer Modern be the default font in LaTeX causes these dimensions to be set to erroneous values...

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

```
\settypeblocksize{⟨height⟩}{⟨width⟩}{⟨ratio⟩}
\textheight \textwidth
```

The command `\settypeblocksize` is the same as `\settrimmedsize` except that it sets the `\textheight` and `\textwidth` for the typeblock. For instance, here are three ways of specifying a 6in by 3in typeblock:

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

Table 6.2: Arguments and results for `\setlrmargins`

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

The typeblock has to be located on the page. There is a relationship between the page, typeblock and margins. The sum of the spine margin, the foredge 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 foredge margin, so that the white space is equally distributed around the sides of the text.

There is usually 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 where the width of the typeblock is fixed and the margins are adjustable, and the other where the size of the margins determines the width of the typeblock.

```
\setlrmargins{⟨spine⟩}{⟨edge⟩}{⟨ratio⟩}
\spinemargin \foremargin
```

The command `\setlrmargins` can be used to specify the side margins[1] with the width of the page and the typeblock being fixed.

Either zero or one argument values are required, with any unvalued argument being denoted by an asterisk. There are several cases to consider and these are tabulated in Table 6.2.

In the Table, S is the calculated spine margin, E is the calculated foredge 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.

[1] Only the spine margin is noted in Figure 6.2; the foredge margin is at the opposite side of the typeblock.

Table 6.3: Arguments and results for `\setlrmarginsandblock`

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

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 foredge margin to be 1.2in (i.e., the foredge 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):

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

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

The command `\setlrmarginsandblock` can be used to specify the spine and foredge 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 6.3. 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 foredge margin to be 1.2in (i.e., the foredge 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):

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

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

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

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

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

Table 6.4: Arguments and results for `\setulmargins`

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

Table 6.5: Arguments and results for `\setulmarginsandblock`

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

The command `\setulmargins` can be used to specify the upper and lower margins[2] 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 6.4. The `\setlrmargins` command maintains the relationship

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

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

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

```
\setcolsepandrule{⟨colsep⟩}{⟨thickness⟩}
\columnsep \columnseprule
```

[2]Only the upper margin is noted in Figure 6.2; the lower margin is the distance between the bottom of the typeblock and the bottom of the page.

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

This completes 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 foredge of the page, either in the header of 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{⟨headheight⟩}{⟨footskip⟩}
\headheight \footskip
```

The command `\setheadfoot` sets the `\headheight` parameter to the value ⟨*headheight*⟩ and the `\footskip` parameter to ⟨*footskip*⟩.

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

The command `\setheaderspaces` is similar to `\setulmargins`. Using the notation U for the upper margin (as before), H_h for the `\headheight`, H_s for the `\headsep` and D for the `\headdrop`, where the `\headdrop` is the distance between the top of the trimmed page and the top of the header[3], 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 6.6. 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 6.3 shows the marginal note parameters on a verso page, and also illustrates that some parameters control different positions on the stock.

[3]The head drop is not shown in Figure 6.2.

Table 6.6: Arguments and results for `\setheaderspaces`

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

```
\setmarginnotes{⟨separation⟩}{⟨width⟩}{⟨push⟩}
\marginparsep \marginparwidth \marginparpush
```

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

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.
- 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.

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

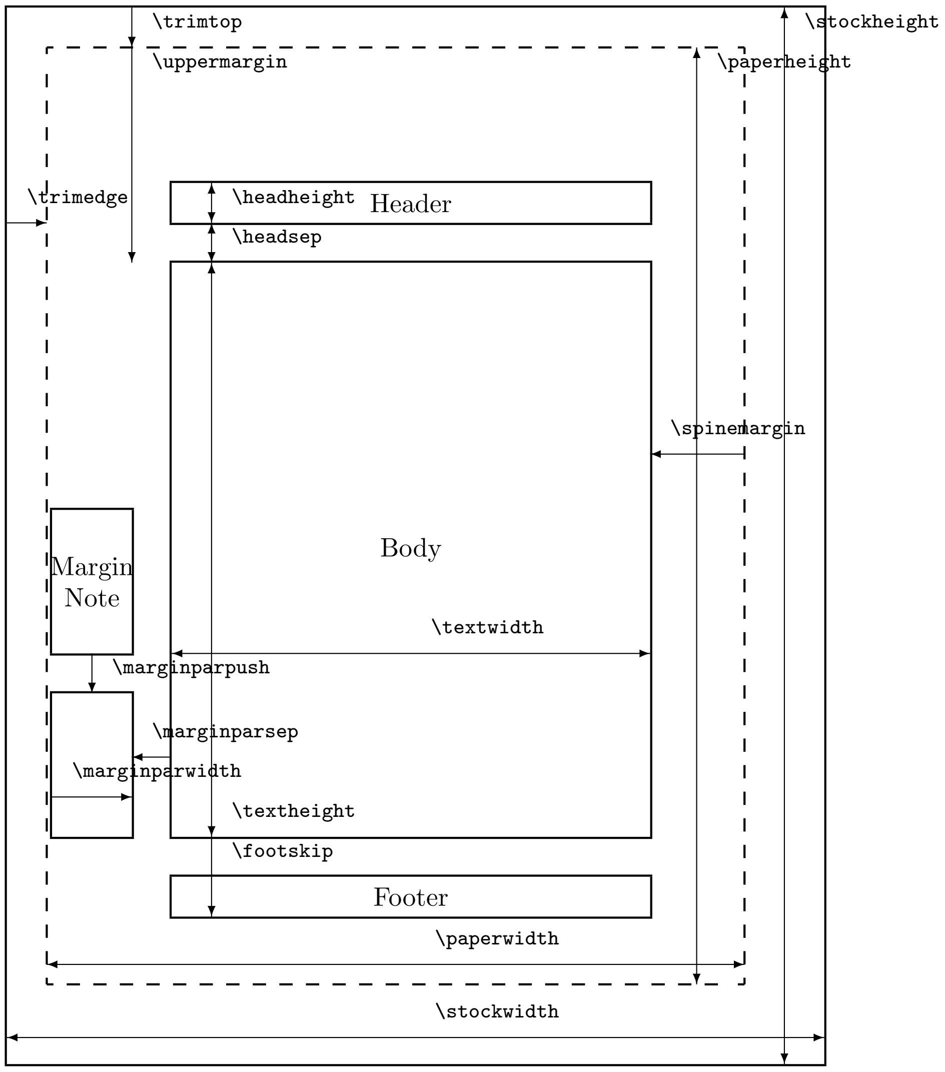


Figure 6.3: The memoir class page layout parameters for a verso page

- 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.
- 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 `\headdrop`.
- 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 are independent; 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 6.2 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 6.7.

```
\checkandfixthelayout
```

The macro `\checkandfixthelayout` takes all the class layout parameters, checks that they have 'sensible' values (e.g., the `\textwidth` is not negative), and then calculates the values for all those LaTeX layout parameters that differ from the class 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
```

The command `\typeoutlayout` may be used at any point in the document to display the current class layout parameter values on the terminal and to write them to the log file. The `\typeoutstandardlayout` does the same thing, except that it outputs the values of the standard parameters.

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\pdfvorigin=0pt\pdfvorigin=1in\fi
  \ifdim\pdfhorigin=0pt\pdfhorigin=1in\fi
  \fi\fi}
```

Table 6.7: The class and LaTeX page layout parameters

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

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

```
\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`. 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 be (at least for an HP 5SiMx LaserJet duplex printer):

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

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 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 foredge margin to the spine 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 297×210mm and 11×8.5in; A4 is taller and narrower than letterpaper.

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

The stocksize is defined by the class option, but we have to work out the trims to reduce the stock to the page. To make life easier, we will only trim the foredge and the bottom of the stock, so the `\trimtop` is zero. The `\trimtop` and `\trimedge` are easily specified by

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

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 foredge margins are specified just by the value of the golden section, via

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

The only remaining calculation to be done is the `\headdrop` 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

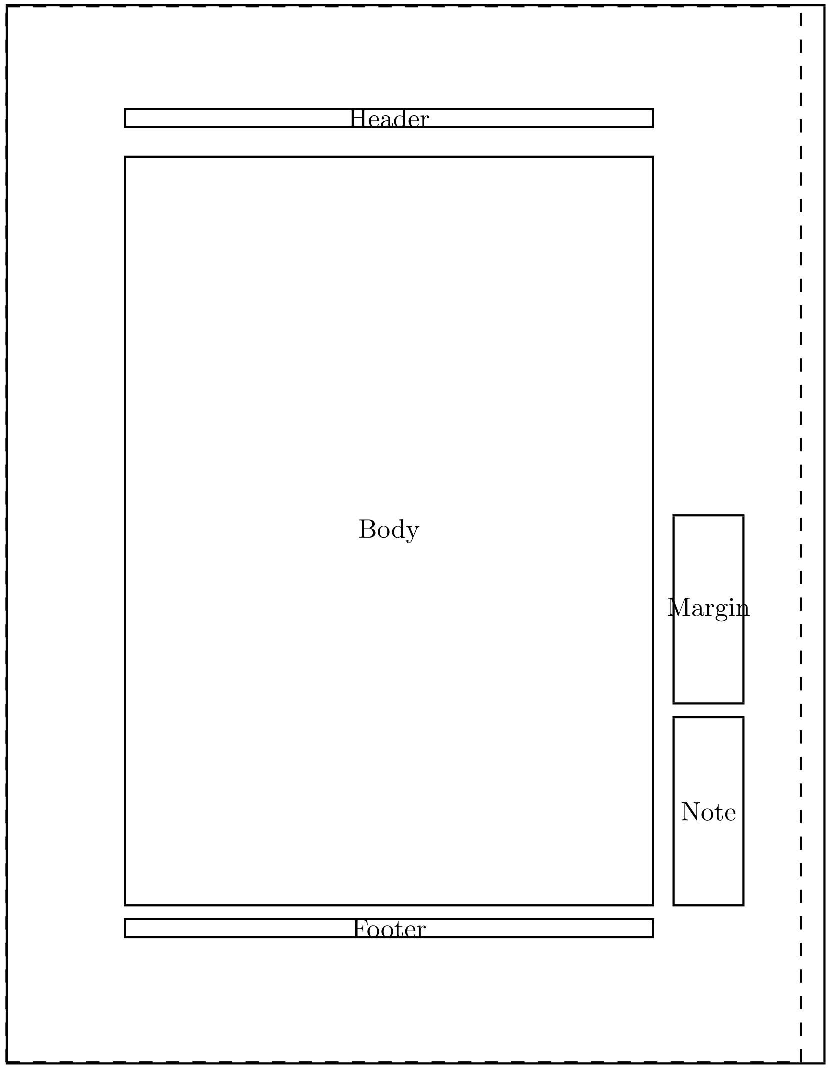
```
\checkandfixthelayout
```

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{\trimedge}{\stockwidth}
\addtolength{\trimedge}{-\paperwidth}
\settypeblocksize{7.75in}{33pc}{*}
\setulmargins{4cm}{*}{*}
\setlrmargins{1.25in}{*}{*}
```

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



Lengths are to the nearest pt.

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

Figure 6.4: The recto page layout for this manual

```
\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 §6.6, 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,[4] has suggested that the following might be better for typesetting the manual on A4 paper.

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

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

Chapter 7

Titles and abstracts

7.1 Introduction

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.

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. Further, additional controls are provided for the typesetting of the `abstract` environment in general.

7.2 Titles

Styling the titling

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

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

```
\newcommand{\maketitle}{%
   \vspace*{\droptitle}
   \maketitlehooka
   {\pretitle \title \posttitle}
   \maketitlehookb
   {\preauthor \author \postauthor}
   \maketitlehookc
```

```
    {\predate \date \postdate}
    \maketitlehookd
    \thispagestyle{title}
  }
```

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

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

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

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

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

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

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

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

```
\droptitle
```

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

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

```
\maketitlehooka \maketitlehookb
\maketitlehookc \maketitlehookd
```

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

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

You can then say:

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

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

\begin{titlingpage} text \end{titlingpage}

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}
```

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 the standard classes' habit of automatically killing the titling commands after `\maketitle` has been issued. You can have multiple `\title`, `\author`, `\date` and `\maketitle` commands in your document if you wish. For example, some reports are issued with a title page, followed by an executive summary, and then they have another, possibly modified, title at the start of the main body of the report. This can be accomplished like this:

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

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

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

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

Styling the thanks

The class provides a configurable `\thanks` command.

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

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

```
\thanksmarkseries{alph}
```

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

```
\continuousmarks
```

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

```
\thanksheadextra{⟨pre⟩}{⟨post⟩}
```

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

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

```
\thanksmark{⟨n⟩}
```

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

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

```
\thanksmarkstyle{⟨defn⟩}
```

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

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

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

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

```
\thanksmarkwidth
```

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

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

giving the default position.

```
\thanksmarksep
```

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

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

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

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

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

```
\thanksfootmark
```

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

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

```
\makethanksmark
\makethanksmarkhook
```

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

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

which does nothing.

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

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

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

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

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

The parameters for the vertical positioning of footnotes and thanks notes, and the default `\footnoterule` are the same (see Figure 13.1 on page 174). You will have to change one or more of these if the vertical spacings of footnotes and thanks notes are meant to be different.

7.3 Abstracts

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

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

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}
```

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

```
        \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 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.

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

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 runin heading; these two declarations are mutually exclusive. Note that the `\abstractnum` declaration has no effect if the abstract is in the `\frontmatter`.

```
\abstractnamefont
\abstracttextfont
```

These two commands can be redefined to change the fonts used for typesetting the heading (defined via `\abstractname`) of the `abstract` environment and the font for typesetting the text of the abstract, respectively. 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.

`\abslabeldelim{⟨text⟩}`

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

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

`\absnamepos`

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

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

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

`\abstitleskip`

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

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

will typeset the heading flush left.

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

8 Document divisions

The chapter is typeset using the sectional headings styles specified at the end of the chapter.

8.1 Introduction

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 headings. 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.

8.2 Book divisions

As described earlier there are three main logical divisions to a book; the front, main 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.

```
\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.

8.3 Sectional divisions

The memoir class lets you divide a document up into seven levels of named divisions. They range from part through chapter and down to sub-paragraph. A particular sectional division is specified by one of the commands `\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`. All the sectional commands, except for chapters, have the same form, so rather than describing each one in turn I will use `\section` to stand for any one of them.

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

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

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

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

The `\part`⟨*title*⟩ command puts the part name (default `Part`), number and ⟨*title*⟩ on a page by itself. The numbering of parts has no effect on the numbering of chapters.

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

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

The `\chapter` command starts a new page and puts the chapter name (default `Chapter`), number and ⟨*title*⟩ at the top of the page. It restarts the numbering of any `\section`s 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 divisional heading that is too close to the bottom of a page is moved to the top of the following page. If this happens and `\flushbottom` is in effect, the contents of the short page are stretched to make the last line flush with the bottom of the typeblock.

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

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

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

```
\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

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

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*⟩ space (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.

$*\quad*\quad*$

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

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

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

$\clubsuit\quad\diamondsuit\quad\clubsuit$

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

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

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

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

```
\tableofcontents \tableofcontents*
\listoffigures \listoffigures*
\listoftables \listoftables*
```

In the standard classes the command `\tableofcontents` typesets a Table of Contents (ToC) at the point in the document where it is issued. In the memoir class it also adds *its* title to the ToC. There is a starred version, `\tableofcontents*`, which does not enter itself into the ToC. So, in this class `\tableofcontents*` is equivalent to the standard classes' `\tableofcontents`. The class also provides the `\listoffigures` and `\listoftables` commands which typeset a List of Figures (LoF) and a List of Tables (LoT), also entering their titles into the ToC. The starred versions of these make no ToC entry.

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

```
\appendix
\appendixname
```

In the standard classes the `\appendix` command changes the numbering of chapters to an alphabetic form and changes the names of chapters from `\chaptername` (default `Chapter`) to the value of `\appendixname` (default `Appendix`). Thus, the first and any subsequent `\chapter`s after the `\appendix` command will be 'Appendix A ...', 'Appendix B ...', and so on. This class provides the same command plus some other ways of dealing with appendices.

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

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

```
\addappheadtotoc
\appendixtocname
```

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

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

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

```
\begin{subappendices} text \end{subappendices}
\namesubappendixtrue \namesubappendixfalse
```

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 `\namesubappendixtrue` *before* the `subappendices` environment, the sub-appendix number in the body of the document will be preceded by the value of `\appendixname`. The `\namesubappendixfalse` declaration may be used to switch off this behaviour.

Table 8.1: Division levels

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

8.4 Numbering

Each type of sectional division has an associated *level* as shown in Table 8.1. 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 parts will be numbered.

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

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

```
\setsecnumdepth{subsection}
```

The command `\maxsecnumdepth` sets the maximum value for `secnumdepth` that can be used in the document. You can use `\setsecnumdepth` anywhere you want in the document to (temporarily) change the numbering level.

By default, the class sets:

```
\maxsecnumdepth{section}
\setsecnumdepth{section}
```

The commands `\mainmatter` and `\mainmatter*` set the `secnumdepth` to the value specified by `\maxsecnumdepth`.

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

8.5 Part headings

Part titles *always* start on a new page with the *part* pagestyle.

Several aspects of the typesetting of the `\part` title are configurable.

```
\beforepartskip \afterpartskip
```

These two commands effectively control the spacing before and after the part title. Their default definitions are:

```
\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 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}
```

```
\midpartskip
```

The length `\midpartskip` is the vertical distance between the part number line and the part title. Its default value is 20pt and can be changed by `\setlength` or `\addtolength`. It is probably better to have the spacing in terms of the `\baselineskip`, which will change depending on the font size, instead of a fixed amount.

```
\printpartname \partnamefont
\partnamenum
\printpartnum \partnumfont
```

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. All these can be changed to obtain different effects. For example, to have 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}{}
```

`\printparttitle{`⟨*title*⟩`} \parttitlefont`

The title is typeset by `\printparttitle` using the font specified by `\parttitlefont`. 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

```
\renewcommmand{\parttitlefont}{\normalfont\Huge\scshape\raggedleft}
```

The `\parttitlefont` font is also used by `\appendixpage`, or its starred version, when typesetting an appendix page.

8.6 Chapter headings

The chapter headings are configurable in much the same way as part headings, but in addition there are some built in chapter styles that you may wish to try, or define your own.

Chapters *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.

```
\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 §17.12) `\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
```

```
\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 commmand `\chapterstyle{fred}`. Different styles can be used in the same document. The predefined styles include:

default The normal LaTeX style.

section The heading is typeset like a section; that is, there is just the number and the title on one line. See Chapter 10 on page 105 for an example.

hangnum Like the *section* style except that the chapter number is put in the margin. See Chapter 11 on page 129 for an example.

companion This produces chapter headings like those the the *LaTeX Companion* series of books. See Chapter 14 on page 183 for an example.

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.

demo Try this one to see what it does. On the other hand you can look at Chapter 15 on page 195 to see what it looks like.

```
\beforechapskip \afterchapskip
```

These two lengths define the spacing before and after the chapter heading. Their default values are 50pt and 40pt, respectively.

```
\midchapskip
```

The length `\midchapskip` is the vertical distance between the chapter number line and the chapter title. Its default value is 20pt and can be changed by `\setlength` or `\addtolength`. It is probably better to have the spacing in terms of the `\baselineskip`, which will change depending on the font size, instead of a fixed amount.

```
\printchaptername \chapnamefont
\chapternamenum
\printchapternum \chapnumfont
```

The macro `\printchaptername` typesets the chapter name (the value of `\chaptername`, default `Chapter`) using the font specified by `\chapnamefont`. The default is the `\bfseries` font in the `\huge` size. Likewise the chapter number is typeset by `\printchapternum` using the font specified by `\chapnumfont`, which has the same default as `\chapnamefont`. The macro `\chapternamenum`, which is defined to be a space, is called between printing the chapter name and the number.

```
\printchaptertitle{⟨title⟩} \chaptitlefont
```

The title is typeset by `\printchaptertitle` using the font specified by `\chaptitlefont`. By default this is a `\bfseries` font in the `\Huge` size.

```
\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`.

Defining a chapter style

The following illustrates the essential parts of the code for typesetting a numbered chapter heading.

```
\chapterheadstart
\printchaptername \chapternamenum \printchapternum
\afterchapternum
\printchaptertitle{The chapter title}
\afterchaptertitle
```

```
\printchapternonum
```

The essential code for an unnumbered chapter is simpler than when there is a number to be typeset.

```
\chapterheadstart
\pintchapternonum
\printchaptertitle{The chapter title}
\afterchaptertitle
```

The various macros in the above are initially defined as:

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

A new style is specified by changing the definitions of this last set of macros and/or the various font and skip specifications.

\makechapterstyle{⟨*style*⟩}{⟨*text*⟩}

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

As an example, here is the code for defining the *section* chapter style.

```
\makechapterstyle{section}{%
  \renewcommand{\printchaptername}{}
  \renewcommand{\chapternamenum}{}
  \renewcommand{\chapnumfont}{\chaptitlefont}
  \renewcommand{\printchapternum}{\chapnumfont \thechapter\space}
  \renewcommand{\afterchapternum}{}
}
```

In this style, `\printchaptername` is vacuous, so the normal 'Chapter' is never typeset. The same font is used for the number and the title, and the number is typeset with a space after it. The macro `\afterchapternum` is vacuous, so the chapter title will be typeset immediately after the number.

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

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

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

```
\makeatletter
\newlength{\numberheight}
\newlength{\barlength}
\makechapterstyle{veelo}{%
  \setlength{\beforechapskip}{40pt}
  \setlength{\midchapskip}{25pt}
  \setlength{\afterchapskip}{40pt}
  \renewcommand{\chapnamefont}{\normalfont\LARGE\flushright}
  \renewcommand{\chapnumfont}{\normalfont\HUGE}
  \renewcommand{\chaptitlefont}{\normalfont\HUGE\bfseries\flushright}
  \renewcommand{\printchaptername}{%
    \chapnamefont\MakeUppercase{\@chapapp}}
  \renewcommand{\chapternamenum}{}
  \setlength{\numberheight}{18mm}
  \setlength{\barlength}{\paperwidth}
  \addtolength{\barlength}{-\textwidth}
  \addtolength{\barlength}{-\spinemargin}
  \renewcommand{\printchapternum}{%
    \makebox[0pt][l]{%
      \hspace{.8em}%
      \resizebox{!}{\numberheight}{\chapnumfont \thechapter}%
      \hspace{.8em}%
      \rule{\barlength}{\numberheight}
    }
  }
  \makeoddfoot{plain}{}{}{\thepage}
}
\makeatother
```

If you implement this 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.

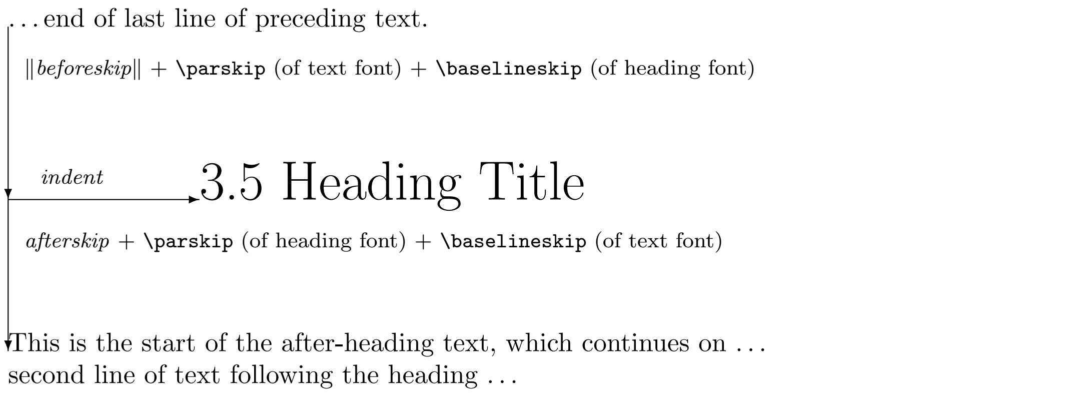

. . . end of last line of preceding text.
‖*beforeskip*‖ + \parskip (of text font) + \baselineskip (of heading font)
indent
3.5 Heading Title
afterskip + \parskip (of heading font) + \baselineskip (of text font)
This is the start of the after-heading text, which continues on . . .
second line of text following the heading . . .

Figure 8.1: Displayed sectional headings

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.

8.7 Lower level headings

The lower level headings — sections down to subparagraphs — are also configurable, but there is nothing corresponding to chapter styles.

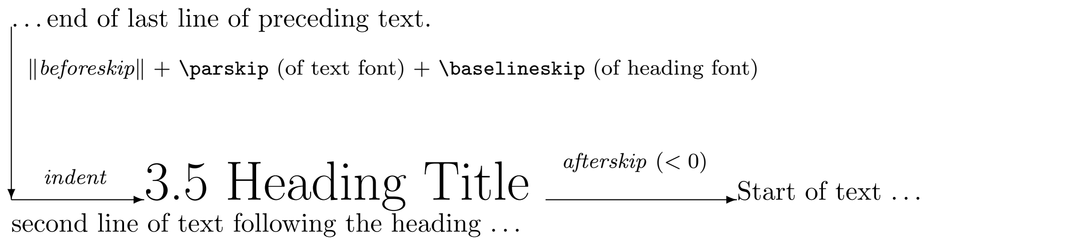
There are essentially three things that may be adjusted for a heading: (a) the vertical distance between the baseline of the text above the heading to the baseline of the heading title, (b) the indentation of the heading from the left hand margin, and (c) the style (font) used for the heading title. Additionally, a heading may be runin 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 8.1 shows the parameters controlling a displayed sectional heading and Figure 8.2 shows the parameters for a runin heading.

In the following I will use `S` to stand for one of `sec`, `subsec`, `subsubsec`, `para` or `subpara`, which are in turn shorthand for `section` through to `subparagraph`.

```
\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 = sec`) it is

```
-3.5ex plus -1ex minus -.2ex
```



. . . end of last line of preceding text.
‖beforeskip‖ + \parskip (of text font) + \baselineskip (of heading font)
indent
3.5 Heading Title
afterskip (< 0)
Start of text . . .
second line of text following the heading . . .

Figure 8.2: Runin sectional headings

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{`⟨*text*⟩`}`

This macro specifies the style (font) for the sectional number and title. As before, the default value of the ⟨*text*⟩ argument depends on the level of the heading. For a `\subsection` (i.e., `S=subsec`) it is `\large\bfseries\raggedright`, to typeset in the `\bfseries` font in the `\large` size; the title will also be set ragged right (i.e., there will be no hyphenation in a multiline title).

Note that the very last element in the ⟨*text*⟩ 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

```
\setsubsubsecheadtstyle{\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}}
\setsecheadstyle{\ruledsec}
```

`\setafterSskip{`⟨*skip*⟩`}`

If the value of the ⟨*skip*⟩ length argument is positive it is the space to leave between the display heading and the following text. If it is negative, then the heading will be runin 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 runin heading, it is `-1em`.

```
\@hangfrom{⟨stuff⟩}
\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{⟨stuff⟩}
\setsecnumformat{⟨code⟩}
```

Internally all the titling macros use a 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

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

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

Table 8.2: Default fonts for sectional headings

`\partnamefont`	`\huge\bfseries`
`\partnumfont`	`\huge\bfseries`
`\parttitlefont`	`\Huge\bfseries`
`\chapnamefont`	`\normalfont\huge\bfseries`
`\chapnumfont`	`\normalfont\huge\bfseries`
`\chaptitlefont`	`\normalfont\Huge\bfseries`
`\secheadstyle`	`\Large\bfseries\raggedright`
`\subsecheadstyle`	`\large\bfseries\raggedright`
`\subsubsecheadstyle`	`\normalsize\bfseries\raggedright`
`\paraheadstyle`	`\normalsize\bfseries`
`\subparaheadstyle`	`\normalsize\bfseries`

8.8 Heading styles for this chapter

As a reference, Table 8.2 lists the default fonts used for the sectional headings. These fonts are all bold but in different sizes depending on the division level.

The commands described below have been put just before the start of this chapter.

For this chapter I have used sans fonts instead of bold fonts. The commands to do this are shown below for chapters down to subsubsections.

```
\renewcommand{\chapnumfont}{\normalfont\huge\sffamily}
\renewcommand{\chaptitlefont}{\chapnumfont}
\setsecheadstyle{\Large\sffamily\raggedright}
\setsubsecheadstyle{\large\sffamily\raggedright}
\setsubsubsecheadstyle{\normalsize\sffamily\raggedright}
```

The chapter heading is typeset like a section heading. I redefine `\printchaptername` and `\chapternamenum` to do nothing, which eliminates the printing of the chapter name and also anything that may be between the name and the number. Changing `\afterchapternum` to be just `\space` eliminates the vertical space that is normally between the number and the title, and instead puts a normal word space between the number and the title.

```
\renewcommand{\printchaptername}{}
\renewcommand{\chapternamenum}{}
\renewcommand{\afterchapternum}{\space}
```

Before the next chapter starts, I want to revert back to the default divisional styles, so the following code is used at the end of this chapter.

```
\chapterstyle{default}
\setsecheadstyle{\Large\bfseries\raggedright}
\setsubsecheadstyle{\large\bfseries\raggedright}
\setsubsubsecheadstyle{\normalsize\bfseries\raggedright}
```

Calling `\chapterstyle{default}` deals with the chapter heading. The other commands change the lower level fonts back to their default values.

8.9 Footnotes and headers

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 `\protect`ed, 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 sectional 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 header. This is unacceptable to any reader. So, a footnoted title should look like this:

```
\chapter[Title text]{Title text\footnote{Do you really have to do this?}}
```

8.10 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{⟨num-style⟩}
\pagenumbering*{⟨num-style⟩}
```

The macros `\pagenumbering` and `\pagenumbering*` cause the folios to be printed using ⟨*num-style*⟩ for the page number, where ⟨*num-style*⟩ 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 `\num-style{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 desireable to have a continuous sequence of numbers irrespective of their displayed form, which is where `\pagenumbering*` 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, as opposed to physically, 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}
```

Whether or not this will be necessary depends on the timing of the `\...pagenumber` commands and TeX's decisions on page breaking.

Chapter 9

Paragraphs and lists

This chapter is typeset with the default sectional headings.

9.1 Introduction

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.

9.2 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 9.1. 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.

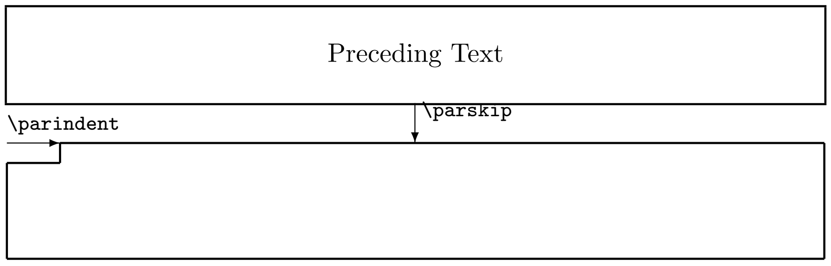


Figure 9.1: Paragraphing parameters

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 grounds. 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.

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`) in effect at the end of the paragraph.

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 is negative the first N lines of the paragraph will be indented while if N is positive the N+1 th lines onwards will be indented. This paragraph was set with `\hangpara{3em}{-3}`. There should be no space between the `\hangpara` command and the start of the paragraph.

```
\begin{hangparas}{⟨indent⟩}{⟨num⟩} text \end{hangparas}
```

The `hangparas` environment is like the `\hangpara` command except that every paragraph in the environment will be hung.

The code implementing the hanging paragraphs is the same as for the hanging package [Wil01f]. Examples of some uses can be found in [Thi99].

As noted eleswhere the sectioning commands use the internal macro `\@hangfrom` as part of the formatting of the titles.

```
\hangfrom{⟨text⟩}
```

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.

9.3 Flush and ragged

Flushleft text has the lefthand end of the lines aligned vertically at the lefthand margin and flushright text has the righthand end of the lines aligned vertically at the righthand margin. The opposites of these are raggedleft text where the lefthand ends are not aligned and raggedright where the righthand end of lines are not aligned. LaTeX normally typesets flushleft and flushright.

```
\begin{flushleft} text \end{flushleft}
\begin{flushright} text \end{flushright}
\begin{center} text \end{center}
```

Text in a `flushleft` environment is typeset flushleft and raggedright, while in a `flushright` environment is typeset raggedleft and flushright. In a `center` environment the text is set raggedleft and raggedright, and each line is centered. A small vertical space is put before and after each of these environments.

```
\raggedleft \raggedright \centering
```

The `\raggedleft` declaration can be used to have text typeset raggedleft and flushright, and similary the declaration `\raggedright` causes typesetting to be flushleft and raggedright. The declaration `\centering` typesets raggedleft and raggedright with each line centered. Unlike the environments, no additional space is added.

```
\raggedyright[⟨space⟩]
\ragrparindent
```

When using `\raggedright` in narrow columns the right hand edge tends to be too ragged, and paragraphs are not indented. Text set `\raggedyright` usually fills more of the available width and paragraphs are indented by `\ragrparindent`, which is initially set to `\parindent`. The optional ⟨*space*⟩ argument, whose default is 2em, can be used to adjust the amount of raggedness. As examples:

```
\raggedyright[0pt]   % typeset flushright
\raggedyright[1fil]  % same as \raggedright
\raggedyright[0.5em] % less ragged than \raggedright
```

Remember that LaTeX typesets on a per-paragraph basis, so that putting the sequence of `\centering`, `\raggedleft` declarations in the same paragraph will cause the entire paragraph to be typeset raggedleft and flushright — the `\centering` declaration is not the one in effect at the end of the paragraph.

9.4 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.

9.5 Changing the textwidth

The `quote` and `quotation` environments both locally change the textwidth, or more precisely, they temporarily increase the left and right margins by equal amounts. Generally speaking it is not a good idea to change the textwidth but sometimes it may be called for.

The commands and environment described below are similar to those in the `chngpage` package [Wil01b], but do differ in some respects.

```
\begin{adjustwith}{⟨left⟩}{⟨right⟩} text \end{adjustwith}
\begin{adjustwith*}{⟨left⟩}{⟨right⟩} text \end{adjustwith*}
```

The `adjustwidth` environment temporarily adds the length ⟨*left*⟩ to the lefthand margin and ⟨*right*⟩ to the righthand margin. That is, a positive length value increases the margin and hence reduces the textwidth, and a negative value reduces the margin and increases the textwidth. The `quotation` environment is roughly equivalent to

```
\begin{adjustwidth}{2.5em}{2.5em}
```

The starred version of the environment, `adjustwidth*`, is 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.

```
\strictpagechecktrue \strictpagecheckfalse
```

Odd/even page checking may be either strict (`\strictpagechecktrue`) or lazy (`\strictpagecheckfalse`). 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 desireable to have any wide figure stick out into the foredge margin where there is usually more room than at the spine margin. This can be accomplished by

```
\begin{figure}
\centering
\strictpagechecktrue
\begin{adjustwidth*}{0em}{-3em}
% the illustration
\caption{...}
\end{adjustwidth*}
\end{figure}
```

A real example in this manual is Table 11.4 on page 152, 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 foredge margins are 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 foredge margins, respectively) then the amount M to be added to the spine margin and subtracted from the foredge 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 foredge margin is then 2.5 inches, and 0.75 inches[1] must be added to the spine margin and subtracted from the foredge 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 spine margin and subtracted from the foredge margin to center the typeblock. An example usage is

```
\calccentering{\mylength}
\begin{adjustwidth*}{\mylength}{-\mylength}
text horizontally centered on the physical page
\end{adjustwidth*}
```

You do not necessarily have to define a new length for the purposes of `\calccentering`. Any existing length will do, such as `\unitlength`, provided it will be otherwise unused between performing the calculation and changing the margins (and that you can, if necessary reset it to its original value — the default value for `\unitlength` is 1pt).

[1]On A4 paper the result would be different.

9.6 Lists

Standard LaTeX provides 4 kinds of lists. There is a general `list` environment which you can use to define your own particular kind of list, and the `description`, `itemize` and `enumerate` lists (which are internally defined in terms of the general `list` environment).

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⟨style⟩
```

In a `description` list the style of the ⟨*label*⟩ is given by the ⟨*style*⟩ argument of the `\descriptionlabel` command. 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}[⟨mark⟩] \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 ⟨*mark*⟩ 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 ...
...
```

```
\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
```

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`. If you had noticed that the lists in this chapter have a different appearance than those in earlier chapters (or even if you hadn't noticed) then that is because the declaration `\tightlists` was put at the start of this chapter.

```
\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 9.2 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.

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}
```

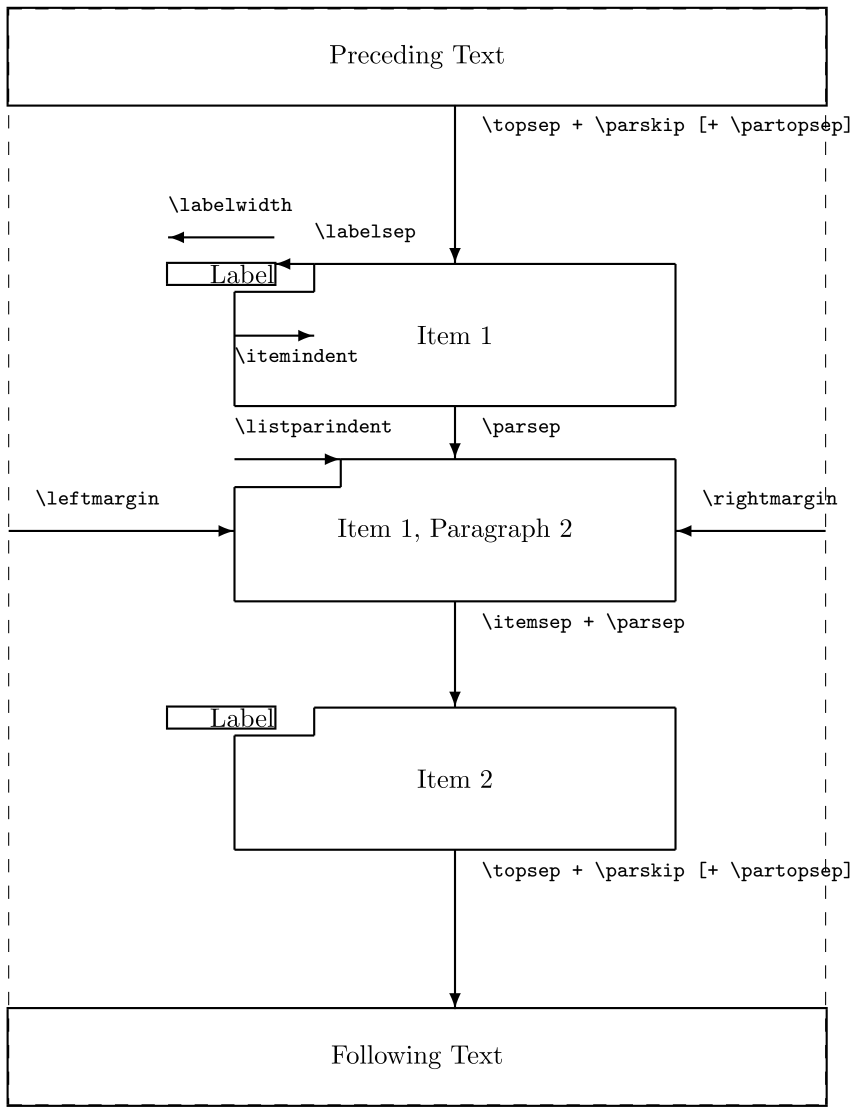


Figure 9.2: The layout parameters for general lists

```
    \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 `aglossary` that could be used for a glossary or other similar kind of listing.

```
% Glossary list
\newenvironment{aglossary}%
               {\begin{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}}}%
               {\end{list}}
\newcommand{\gloss}[1]{\item[#1]\mbox{}\\\nopagebreak}
```

In this case it gets used like this

```
\begin{aglossary}
\gloss{TERM 1} definition
\gloss{TERM 2} ...
\end{aglossary}
```

```
\zerotrivseps \restoretrivseps \savetrivseps
```

Several environments, such as `center`, are defined in terms of a `trivlist` (a very simple list form). There is vertical space before and after such an environment. If you don't want this, then the declaration `\zerotrivseps` eliminates such spaces. You can think of it being defined like:

```
\newcommand*{\zerotrivseps}{%
  \setlength{\topsep}{0pt}%
  \setlength{\partopsep}{0pt}}
```

To restore the spacing call the `\restoretrivseps` declaration. The command `\savetrivseps` stores the `\topsep` and `\partopsep` values, and `\restoretrivseps` sets them to whatever values were stored. The class itself calls `\savetrivseps` to keep the default values.

10 Tops and tails

This chapter uses the *section* chapterstyle.

10.1 Introduction

The following discussions are focussed on the elements at the start and end of a document; the Table of Contents at the start and the Index at the end. The functionality described is equivalent to the combination of the tocloft and tocbibind packages [Wil01i, Wil01h]

In the standard classes the typographic design of the Table of Contents (ToC), the List of Figures (LoF) and List of Tables (LoT) is fixed or, more precisely, it is buried within the class definitions.

10.2 LaTeX's ToC methods

This is a general description of how LaTeX does the processing for a Table of Contents (ToC). As the processing for List of Figures (LoF) and List of Tables (LoT) is similar I will just discuss the ToC.

First of all, remember that each sectional division has an associated level as listed in Table 8.1 on page 82. 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.

```
\maxtocdepth{⟨secname⟩}
\settocdepth{⟨secname⟩}
```

The memoir 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 analagous to the `\setsecnumdepth` command described in §8.4. 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].

```
\addcontentsline{⟨file⟩}{⟨kind⟩}{⟨text⟩}
```

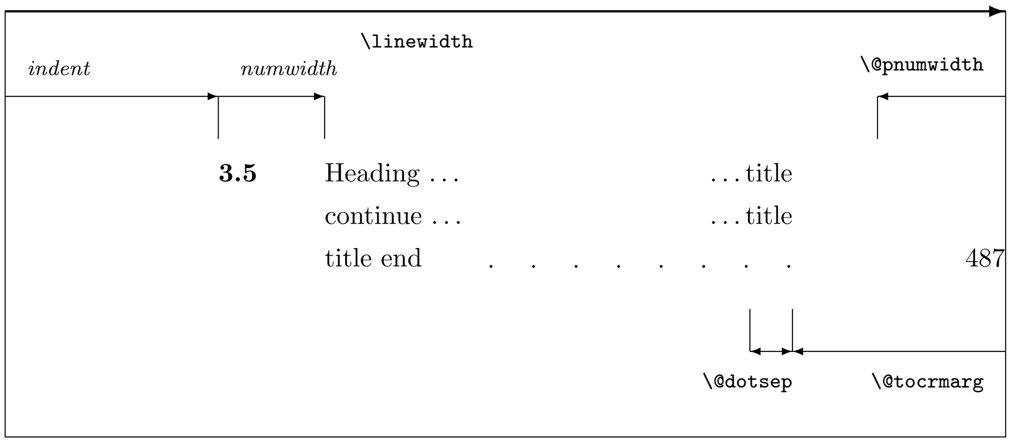


Figure 10.1: Layout of a ToC (LoF, LoT) entry

LaTeX generates a `.toc` file if the document contains a `\tableofcontents` command. The sectioning commands[1] put entries into the `.toc` file by calling the LaTeX `\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}`.

```
\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`.

```
\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 each ⟨*kind*⟩, LaTeX provides a command:
`\l@kind{`⟨*title*⟩`}{`⟨*page*⟩`}`
which performs the actual typesetting of the `\contentsline` entry.

```
\@pnumwidth{⟨length⟩}
\@tocrmarg{⟨length⟩}
\@dotsep{⟨number⟩}
```

The general layout of a typeset entry is illustrated in Figure 10.1. There are three internal LaTeX commands that are used in the typesetting. The page number is typeset flushright

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

Table 10.1: Indents and Numwidths (in ems)

Entry	Level	Standard		memoir class	
		indent	numwidth	indent	numwidth
part	-1	0	—	0	—
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

in a box of width `\@pnumwidth`, and the box is at the righthand margin. If the page number is too long to fit into the box it will stick out into the righthand margin. The title text is indented from the righthand margin by an amount given by `\@tocrmarg`. Note that `\@tocrmarg` should be greater than `\@pnumwidth`. Some entries are typeset with a dotted leader between the end of the title title text and the righthand margin indentation. The distance, in *math units*[2] between the dots in the leader is given by the value of `\@dotsep`. In the standard classes the same values are used for the ToC, LoF and the LoT.

The standard values for these internal commands are:

- `\@pnumwidth` = 1.55em
- `\@tocrmarg` = 2.55em
- `\@dotsep` = 4.5

The values can be changed by using `\renewcommand`, in spite of the fact that the first two appear to be lengths.

Dotted leaders are not available for Part and Chapter ToC entries

```
\numberline{⟨number⟩}
```

Each `\l@kind` macro is responsible for setting the general *indent* from the lefthand margin, and the *numwidth*. The `\numberline` macro is responsible for typesetting the number flushleft in a box of width *numwidth*. If the number is too long for the box then it will protrude into the title text. The title text is indented by (*indent + numwidth*) from the lefthand margin. That is, the title text is typeset in a block of width
(`\linewidth` - *indent* - *numwidth* - `\@tocrmarg`).

Table 10.1 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. Also, which somewhat surprises me, the table and figure entries are all indented.

```
\@dottedtocline{⟨level⟩}{⟨indent⟩}{⟨numwidth⟩}
```

[2]There are 18mu to 1em.

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 10.1, and ⟨*indent*⟩ and ⟨*numwidth*⟩ are the *indent* and *numwidth* as illustrated in Figure 10.1. 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 this class 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 `\protect`ed.

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 `\addtocontents` and `\addcontentsline` their arguments must be protected by preceding each fragile command with `\protect`. The result of the example above would be the following two lines in the `.toc` file (assuming that it is the second Part and is on page 34):

```
\contentsline {part}{II\hspace {1em}Part title}{34}
\mbox {}\hrulefill \par
```

If the `\protect`s were not used, then the second line would instead be:

```
\unhbox \voidb@x \hbox {}\unhbox \voidb@x \leaders \hrule \hfill \kern \z@ \par
```

- 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
  \settocdepth{chapter}
  \chapter{First appendix}
  ...
  ```

Remember, 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
```

10.3 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.

Changing the titles

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

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

Following LaTeX custom, the title texts are the values of the `\contentsname`, `\listfigurename` and `\listtablename` commands.

The commands for controlling the typesetting of the ToC, LoF and LoT titles all follow a similar pattern So for convenience (certainly mine, and hopefully yours) in the following descriptions I will use `Z` to stand for 'toc' or 'lof' or 'lot'. For example, `\Zmark` stands for `\tocmark` or `\lofmark` or `\lotmark`.

The code for typesetting the ToC title looks like:

```
\tocheadstart
\printtoctitle{\contentsname}
\tocmark
\thispagestyle{chapter}
\aftertoctitle
```

where the macros are described below.

```
\Zheadstart
```

This macro is called before the title is actually printed. Its default definition is

```
\newcommand{\Zheadstart}{\chapterheadstart}
```

```
\printZtitle{⟨title⟩}
```

The title is typeset via `\printZtitle`, which defaults to using `\printchaptertitle` for the actual typesetting.

```
\Zmark
```

These macros sets the appearance of the running heads on the ToC, LoF, and LoT pages. The default definition is equivalent to:

```
\newcommmand{\Zmark}{\@mkboth{\...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.

```
\afterZtitle
```

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{\printZtitle}[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{\printZtitle}[1]{\centering\Large\bfseries #1}
  ```

- Writing
  ```
  \renewcommand{\afterZtitle}{\thispagestyle{empty}\afterchaptertitle}
  ```
  will result in the first page of the listing using the *empty* pagestyle instead of the default *chapter* pagestyle.
- Doing
  ```
  \renewcommand{\afterZtitle}{%
    \par\nobreak \mbox{}\hfill{\normalfont Page}\par\nobreak}
  ```
  will put the word 'Page' flushright on the line following the title.

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 [Wil99a] or in [GMS94, page 34], and are repeated in Figure 10.1.

```
\cftdot{⟨text⟩}
```

In the default ToC typsetting only the more minor entries have dotted leader lines between the sectioning title and the page number. The class provides for general leaders for all entries. The 'dot' in a leader is given by the value of `\cftdot`. Its default definition is `\newcommand{\cftdot}{.}` which gives the default dotted leader. By changing `\cftdot` you can use symbols other than a period in the leader. For example

```
\renewcommand{\cftdot}{\ensuremath{\ast}}
```

will result in a dotted leader using asterisks as the symbol.

```
\cftdotsep
\cftnodots
```

Each kind of entry can control the separation between the dots in its leader (see below). For consistency though, all dotted leaders should use the same spacing. The macro `\cftdotsep` specifies the default spacing. However, if the separation is too large then no dots will be actually typeset. The macro `\cftnodots` is a separation value that is 'too large'.

```
\setpnumwidth{⟨length⟩}
\setrmarg{⟨length⟩}
```

The page numbers are typeset in a fixed width box. The command `\setpnumwidth` can be used to change the width of the box (LaTeX 's internal `\@pnumwidth`). The title texts will end before reaching the righthand margin. `\setrmarg` can be used to set this distance (LaTeX 's internal `\@tocrmarg`). Note that the length used in `\setrmarg` should be greater than the length set in `\setpnumwidth`. These values should remain constant in any given document.

This manual requires more space for the page numbers than the default, so the following was set in the preamble:

```
\setpnumwidth{2.55em}
\setrmarg{3.55em}
```

```
\cftparskip
```

Normally the `\parskip` in the ToC, etc., is zero. This may be changed by changing the length `\cftparskip`. Note that the current value of `\cftparskip` is used for the ToC, LoF and LoT, but you can change the value before calling `\tableofcontents` or `\listoffigures` or `\listoftables` if one or other of these should have different values (which is not a good idea).

Again for convenience, in the following I will use `X` to stand for any of the following:

- `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

```
\cftchapterbreak
```

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

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

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

```
\cftbeforeXskip
```

This length controls the vertical space before an entry. It can be changed by using `\setlength`.

```
\cftXindent
```

This length controls the indentation of an entry from the left margin (*indent* in Figure 10.1). It can be changed using `\setlength`.

```
\cftXnumwidth
```

This length controls the space allowed for typesetting title numbers (*numwidth* in Figure 10.1). 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.

```
    {\cftXfont {\cftXpresnum SNUM\cftXaftersnum\hfil} \cftXaftersnumb TITLE}
           {\cftXleader}{\cftXpagefont PAGE}\cftXafterpnum\par

    {\cftXfont TITLE}{\cftXleader}{\cftXpagefont PAGE}\cftXafterpnum\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
`{\cftXpresnum SNUM\cftaftersnum\hfil}`
is typeset within a box of width `\cftXnumwidth`.

```
\cftXfont
```

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

```
\cftXpresnum \cftXaftersnum \cftXaftersnumb
```

The section number is typeset within a box of width `\cftXnumwidth`. Within the box the macro `\cftXpresnum` is first called, then the number is typeset, and the `\cftXaftersnum` 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 `\cftXaftersnumb` 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.

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

In the ToC, the macros `\partnumberline` and `\chapternumberline` are responsible respectively for typesetting the `\part` and `\chapter` numbers. Internally they use `\cftXpresnum`, `\cftXaftersnum` and `\cftaftersnumb` as above. If you do not want, say, the `\chapter` number to appear you can do:

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

NOTE: Because the hyperref package [Rahtz02] does not understand the `\partnumberline` and `\chapternumberline` commands, if you use the hyperref package you will also have to use the memhfixc package, which comes with memoir.

```
\cftXleader
\cftXdotsep
```

`\cftXleader` 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 `\cftXdotsep` (`\@dotsep` in Figure 10.1). 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.

```
\cftXpagefont
```

This defines the font to be used for typesetting the page number. It can be changed by `\renewcommand`.

```
\cftXafterpnum
```

This macro is called after the page number has been typeset. Its default is to do nothing. It can be changed by `\renewcommand`.

```
\cftsetindents{⟨entry⟩}{⟨indent⟩}{⟨numwidth⟩}
```

The command `\cftsetindents` sets the ⟨*entry*⟩'s *indent* to the length ⟨*indent*⟩ and its *numwidth* to the length ⟨*numwidth*⟩. The ⟨*entry*⟩ argument is the name of one of the standard entries (e.g., `subsection`) or the name of entry that has been defined within the document. For example

```
  \cftsetindents{figure}{0em}{1.5em}
```

will make figure entries left justified.

This manual requires more space for section numbers in the ToC than the default (which allows for three digits). Consequently the preamble contains the following:

```
\cftsetindents{section}{1.5em}{3.0em}
\cftsetindents{subsection}{4.5em}{3.9em}
\cftsetindents{subsubsection}{8.4em}{4.8em}
\cftsetindents{paragraph}{10.7em}{5.7em}
\cftsetindents{subparagraph}{12.7em}{6.7em}
```

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 `\cftXfont`, `\cftXaftersnum`, `\cftXaftersnumb`, `\cftXleader` and `\cftXafterpnum`, either singly or in combination. For the sake of some examples, assume that we have the following initial definitions

```
  \newcommand{\cftXfont}{}
  \newcommand{\cftXaftersnum}{}
  \newcommand{\cftXaftersnumb}{}
  \newcommand{\cftXleader}{\cftdotfill{\cftXdotsep}}
  \newcommand{\cftXdotsep}{\cftdotsep}
  \newcommand{\cftXpagefont}{}
  \newcommand{\cftXafterpnum}{}
```

(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{\cftXdotsep}{\cftnodots}
  ```
- To put something (e.g., a name) before the title (number):
  ```
  \renewcommand{\cftXpresnum}{SOMETHING }
  ```
- To add a colon after the section number:
  ```
  \renewcommand{\cftXaftersnum}{:}
  ```
- To put something before the title number, add a colon after the the title number, set everything in bold font, and start the title text on the following line:

```
\renewcommand{\cftXfont}{\bfseries}
\renewcommand{\cftXleader}{\bfseries\cftdotfill{\cftXdotsep}}
\renewcommand{\cftXpagefont}{\bfseries}
\renewcommand{\cftXpresnum}{SOMETHING }
\renewcommand{\cftXaftersnum}{:}
\renewcommand{\cftXaftersnumb}{\\}
```

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\cftXpresnum\cftXaftersnum}
\addtolength{\cftXnumwidth}{\mylen} % add the extra space
```

- To set the section numbers flushright:

```
\setlength{\mylen}{0.5em}    % extra space at end of number
\renewcommand{\cftXpresnum}{\hfill} % note the double ‘l’
\renewcommand{\cftXaftersnum}{\hspace*{\mylen}}
\addtolength{\cftXnumwidth}{\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{\cftXfont}{\hfill\bfseries}
\renewcommand{\cftXleader}{}
```

- To set the page number immediately after the entry text instead of at the righthand margin:

```
\renewcommand{\cftXleader}{}
\renewcommand{\cftXafterpnum}{\cftparfillskip}
```

```
\cftparfillskip
```

By default the `\parfillskip` value is locally set to fill up the last line of a paragraph. Just changing `\cftXleader` puts horrible interword spaces into the last line of the title. The `\cftparfillskip` command is provided just so that the above effect can be achieved.

```
\cftpagenumbersoff{⟨entry⟩}
\cftpagenumberson{⟨entry⟩}
```

The command `\cftpagenumbersoff` will eliminate the page numbers for ⟨*entry*⟩ in the listing, where ⟨*entry*⟩ is the name of one of the standard kinds of entries (e.g., `subsection`, or `figure`) or the name of a new entry defined in the document.

The command `\cftpagenumberson` reverses the effect of a corresponding `\cftpagenumbersoff` for ⟨*entry*⟩.

One question that appeared on the `comp.text.tex` newsgroup asked how to get the titles of Appendices list in the ToC *without* page numbers. Here is a simple way of doing it.

```
...
\appendix
\addtocontents{toc}{\cftpagenumbersoff{chapter}}
\chapter{First appendix}
```

If there are other chapter type headings to go into the ToC after the appendices (perhaps a bibliography or an index), then it will be necessary to do a similar

```
\addtocontents{toc}{\cftpagenumberson{chapter}}
```

after the appendices to restore the page numbering in the ToC.

At this point, I leave it up to your ingenuity as to other effects that you can achieve. However, if you come up with further examples, let me know for possible inclusion in a later version of this document.

10.4 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` commmand that can be used to typeset a listing of answers under the title `\listanswername`, where the answer titles are in a `.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:

Answer 1. Hard
The ...

As mentioned above, the `\newlistof` command creates several new commands in addition to `\listofcom`, most of which you should now be familiar with. For convenience, assume that `\newlistof{...}{Z}{...}` has been issued so that `Z` is the new file extension and corresponds to the `Z` in §10.3. Then in addition to `\listofcom` the following new commands will be made available.

The four commands, `\Zmark`, `\Zheadstart`, `\printZtitle`, and `\afterZtitle`, are analagous to the commands of the same names described in §10.3 (internally the class uses the `\newlistof` macro to define the ToC, LoF and LoT). In particular the default definition of `\Zmark` is equivalent to:

```
\newcommand{\Zmark}{\@mkboth{listofname}{listofname}}
```

However, this may well be altered by the particular `\pagestyle` in use.

```
Zdepth
```

The counter `Zdepth` is analagous to the standard `tocdepth` counter, in that it specifies that entries in the new listing should not be typeset if their numbering level is greater than `Zdepth`. The default definition is equivalent to

```
\setcounter{Zdepth}{1}
```

```
\insertchapterspace
\addtodef{⟨macro⟩}{⟨prepend⟩}{⟨append⟩}
```

Remember that the `\chapter` command uses `\insertchapterspace` to insert vertical spaces into the LoF and LoT. If you want similar spaces added to your new listing then you have to modify `\insertchapterspace`. The easiest way to do this is via the `\addtodef` macro, like:

```
\addtodef{\insertchapterspaces}{}%
  {\addtocontents{ans}{\protect\addvspace{10pt}}}
```

The `\addtodef` macro is described later in §17.9.

The other part of creating a new 'List of...', is to specify the formatting of the entries, i.e., define an appropriate `\l@kind` macro.

```
\newlistentry[⟨within⟩]{⟨cntr⟩}{⟨ext⟩}{⟨level-1⟩}
```

The command `\newlistentry` creates the commands necessary for typesetting an entry in a 'List of...'. The first required argument, ⟨*cntr*⟩ is used to define a new counter called `cntr`, unless `cntr` is already defined. The optional ⟨*within*⟩ argument can be used so that `cntr` gets reset to one every time the counter called `within` is changed. That is, the first two arguments when `cntr` is not already defined, are equivalent to calling `\newcounter{`⟨*cntr*⟩`}[`⟨*within*⟩`]`. If `cntr` is already defined, `\newcounter` is not called. `cntr` is used for the number that goes along with the title of the entry.

The second required argument, ⟨*ext*⟩, is the file extension for the entry listing. The last argument, ⟨*level-1*⟩, is a number specifying the numbering level minus one, of the entry in a listing.

Calling `\newlistentry` creates several new commands. Assuming that it is called as `\newlistentry[within]{X}{Z}{N}`, where `X` and `Z` are similar to the previous uses of them (e.g., `Z` is the file extension), and `N` is an integer number, then the following commands are made available.

The set of commands `\cftbeforeXskip`, `\cftXfont`, `\cftXpresnum`, `\cftXaftersnum`, `\cftXaftersnumb`, `\cftXleader`, `\cftXdotsep`, `\cftXpagefont`, and `\cftXafterpnum`, are analagous to the commands of the same names described in §10.3. Their default values are also as described earlier.

The default values of `\cftXindent` and `\cftXnumwidth` are set according to the value of the ⟨*level-1*⟩ argument (i.e., `N` in this example). For `N=0` the settings correspond to those for figures and tables, as listed in Table 10.1 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@X` is an internal command that typesets an entry in the list, and is defined in terms of the above `\cft*X*` commands. It will not typeset an entry if `\Zdepth` is `N` or less, where `Z` is the listing's file extension.

The command `\theX` prints the value of the `X` counter. It is initially defined so that it prints arabic numerals. If the optional ⟨*within*⟩ argument is used, `\theX` is defined as

```
\renewcommand{\theX}{\thewithin.\arabic{X}}
```

otherwise as

```
\renewcommand{\theX}{\arabic{X}}
```

As an example of the independent use of `\newlistentry`, the following will set up for sub-answers.

```
\newlistentry[answer]{subanswer}{1}
\renewcommand{\thesubanswer}{\theanswer.\alph{subanswer}}
\newcommand{\subanswer}[1]{
   \refstepcounter{subanswer}
   \par\textbf{\thesubanswer) #1}
   \addcontentsline{ans}{subanswer{\protect\numberline{\thesubanswer}#1}}
\setcounter{ansdepth}{2}
```

And then:

```
\answer{Harder} The \ldots
  \subanswer{Reformulate the problem} It assists \ldots
```

will be typeset as:

Answer 2. Harder
The . . .
2.a) Reformulate the problem It assists . . .

By default the answer entries will appear in the List of Answers listing (typeset by the `\listofanswers` command). In order to get the subanswers to appear, the `\setcounter{ansdepth}{2}` command was used above.

To turn off page numbering for the subanswers, do
`\cftpagenumbersoff{subanswer}`

As another example of `\newlistentry`, suppose that an extra sectioning division below `subparagraph` is required, called `subsubpara`. The `\subsubpara` command itself can be defined via the LaTeX kernel `\@startsection` command. Also it is necessary to define a `\subsubparamark` macro, a new `subsubpara` counter, a `\thesubsubpara` macro and a `\l@subsubpara` macro. Using `\newlistentry` takes care of most of these as shown below (remember the caveats about commands with `@` signs in them).

```
\newcommand{\subsubpara}{\@startsection{subpara}
   {6}                          %                   level
   {\parindent}                 %  indent from left margin
   {3.25ex \@plus1ex \@minus .2ex}  %     skip above heading
   {-1em}       runin 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.

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 or recto 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[3] that print the page number before or after 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}
```

[3]These only work for arabic page numbers.

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 `\caption`s.

10.5 Extras

Some old style novels, and even some modern text books,[4] 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⟩}
```

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.

```
\prechapterprecis \postchapterprecis
```

[4]For example, Robert Sedgewick, *Algorithms*, Addison-Wesley, 1983.

The `\chapterprecishere` macro is intended for use immediately after a `\chapter`. The ⟨*text*⟩ 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.

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

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

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

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

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*⟩`}{`⟨*text*⟩`}{`⟨*page*⟩`}`
`\cftaddnumtitleline{`⟨*ext*⟩`}{`⟨*kind*⟩`}{`⟨*num*⟩`}{`⟨*title*⟩`}{`⟨*page*⟩`}`

The command `\cftaddtitleline` will write a `\contentsline` entry into ⟨*ext*⟩ for a ⟨*kind*⟩ entry with title ⟨*title*⟩ and page number ⟨*page*⟩. Any fragile commands used in the arguments must be protected. That is, an entry is made of the form:

```
\contentsline{kind}{title}{page}
```

The command `\cftaddnumtitleline` is similar to `\cftaddtitleline` except that it also includes ⟨*num*⟩ as the argument to `\numberline`. That is, an entry is made of the form

```
\contentsline{kind}{\numberline{num} title}{page}
```

As an example of the use of these commands, noting that the default LaTeX values for `\@pnumwidth` and `\@tocrmarg` are 1.55em and 2.55em respectively, one might do the following for a figure on the frontispiece page.

```
...
 this is the frontispiece page with no number
 draw or import the picture (with no \caption)
\cftlocalchange{lof}{4em}{5em} % make pnumwidth big enough for
                               % frontispiece and change margin
\cftaddtitleline{lof}{figure}{The title}{frontispiece}
\cftlocalchange{lof}{1.55em}{2.55em} % return to normal settings
\clearpage
...
```

Recall that a `\caption` command will put an entry in the `.lof` file, which is not wanted here. If a caption is required, then you can either craft one youself or, assuming that your general captions are not too exotic, use the `\legend` command (see later). If the illustration is numbered, use `\cftaddnumtitleline` instead of `\cftaddtitleline`.

10.6 Tails

The commands for producing a bibliography and an index are the same as for the standard classes, but there are two differences.

Bibliography

```
\begin{thebibliography}{⟨exlabel⟩}
\bibitem ...
\end{thebibliography}
\bibname
```

The bibliography is typeset within the `thebibliography` environment. This takes one required argument, ⟨*exlabel*⟩, which is a piece of text as wide as the widest label in the bibliography. The value of `\bibname` (default 'Bibliography') is used as the title.

```
\bibintoc \nobibintoc
```

The declaration `\bibintoc` will cause the `thebibliography` environment to add the title to the ToC, while the declaration `\nobibintoc` ensures that the title is not added to the ToC. The default is `\bibintoc`.

```
\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{\postbibhook}{%
CTAN is the Comprehensive TeX Archive Network and URLS for the
several CTAN mirrors can be found at \url{http://www.tug.org}.}
```

```
\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}
```

The definition of the `thebibliography` environment is:

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

```
\bibsection
```

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

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

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

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

If you use the natbib and/or the chapterbib packages [Dal99, Ars01b] with the sectionbib option, then `\bibsection` is changed appropriately to typeset as a numbered section.

The jurabib package [Ber02] redefines the `thebibliography` environment, providing its own methods for listing the items. However, the redefinition also eliminates the opportunity to add the Bibliography to the Table of Contents and to have some introductory text. To restore these to the class specification, put the following in your preamble after loading jurabib:

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

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

```
\bibitemsep
```

The natbib package provides a length, `\bibsep` which can be used to alter the vertical spacing between the entries in the bibliography. Setting `\bibsep` to 0pt removes any extra space between the entries. The equivalent length provided by the class for changing the space between bibliography entries is `\bibitemsep`, which by default is set to the default value of `\itemsep`.

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

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

```
\biblistextra
```

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

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

Index

The indexing commands have been significantly enhanced with respect to the standard classes and include the functionality provided by the makeidx, showidx and index packages [Bra94, Jon95]; these packages should not be used.

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

```
\onecolindextrue
\onecolindexfalse
```

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

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

The macro `\makeindex`, which must be put in the preamble if it is used, opens an `idx` file, which by default is called `jobname.idx`, where `jobname` is the name of the main LaTeX source file. If the optional ⟨*file*⟩ argument is given then a file called `file.idx` will be opened instead. The macro `\printindex` reads an `ind` file called `jobname.ind`, which should contain an `theindex` environment and the indexed items. If the optional ⟨*file*⟩ argument is given then the `file.ind` file will be read. The MAKEINDEX program is often used to convert an `idx` file to an `ind` file.

```
\begin{theindex} entries \end{theindex}
\indexname
```

The index is typeset in two columns within the `theindex` environment. 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. Its default value is 35pt. 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 twocolumn 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.}
```

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

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

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

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

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

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

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

```
\reportnoidxfilefalse
\reportnoidxfiletrue
```

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

```
\showindexmarktrue
\showindexmarkfalse
```

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

Indexing and the natbib package

The natbib package [Dal99] will make an index of citations if `\citeindextrue` is put in the preamble after the natbib package is called for.

```
\citeindexfile
```

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

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

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

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

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

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

before

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

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

```
\renewcommand{\citeindexfile}{authors} % write to authors.idx
\makeindex[\citeindexfile]
\usepackage{natbib}
```

```
\citeindextrue
...
\renewcommand{\indexname}{Index of citations}
\printindex[\citeindexfile]
```

or along the lines of

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

Populating the `idx` file

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

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

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

11 Captions and floats

This chapter uses the *hangnum* chapterstyle. Section numbers are also hung into the margin to match via the declaration `\hangsecnum`.

11.1 Introduction

Some publishers require, and some authors prefer, captioning styles other than the one style provided by standard LaTeX. This chapter describes how you can implement your own captioning style.

Some publishers require 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. Further, a mechanism is provided for defining new float environments.

The commands and environments described below are very similar to those supplied by the ccaption package [Wil01d].

11.2 Captions

Changing the caption style

```
\captiondelim{⟨delim⟩}
```

The default captioning style is to put a delimeter in the form of a colon between the caption number and the caption title. The command `\captiondelim` can be used to change the delimeter. For example, to have an en-dash instead of the colon, `\captiondelim{-- }` will do the trick. Notice that no space is put between the delimeter and the title unless it is specified in the ⟨*delim*⟩ parameter. The class initially specifies `\captiondelim{: }` to give the normal delimeter.

```
\captionnamefont{⟨fontspec⟩}
```

The ⟨*fontspec*⟩ specified by `\captionnamefont` is used for typesetting the caption name; that is, the first part of the caption up to and including the delimeter (e.g., the portion 'Table 3:'). ⟨*fontspec*⟩ can be any kind of font specification and/or command and/or text. This first part of the caption is treated like: `{\captionnamefont Table 3: }`, so font declarations, not font text-style commands, are needed for ⟨*fontspec*⟩ For instance, `\captionnamefont{\Large\sffamily}` to specify a large sans-serif font. The class initially specifies `\captionnamefont{}` to give the normal font.

```
\captiontitlefont{⟨fontspec⟩}
```

Similarly, the ⟨*fontspec*⟩ specified by `\captiontitlefont` is used for typesetting the title text of a caption. For example, `\captiontitlefont{\itshape}` for an italic title text. The class initially specifies `\captiontitlefont{}` to give the normal font.

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

Caption styles are set according to the `\captionstyle` declaration. Unless the optional ⟨*short*⟩ argument is given all captions are typeset according to ⟨*normal*⟩. If the optional ⟨*short*⟩ argument is specifed, captions that are less than one line in length are typeset according to ⟨*short*⟩. By default the name and title of a caption are typeset as a block (non-indented) paragraph.

Sensible values for the arguments include: `\raggedleft`, `\centering`, `\raggedright`, and `centerlastline`. The class initially specifies

```
\captionstyle{}
```

which gives the block paragraph style. The `\centerlastline` style gives a block paragraph but with the last line centered.

```
\hangcaption
\indentcaption{⟨length⟩}
\normalcaption
```

The command `\hangcaption` will cause captions to be typeset with the second and later lines of a multiline caption title indented by the width of the caption name. The command `\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` command undoes any previous `\hangcaption` or `\indentcaption` command. The class initially specifies `\normalcaption` to give the normal non-indented paragraph style.

```
\changecaptionwidth
\captionwidth{⟨length⟩}
\normalcaptionwidth
```

Issuing the command `\changecaptionwidth` will cause the captions to be typeset within a total width ⟨*length*⟩ as specified by `\captionwidth`. Issuing the command `\normalcaptionwidth` will cause captions to be typeset as normal full width captions. The class initially specifies `\normalcaptionwidth` and `\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.

Table 11.1
REDESIGNED TABLE CAPTION STYLE

three	III
five	V
eight	VIII

```
\precaption{⟨pretext⟩}
\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{}` and `\postcaption{}` to give the normal appearance.

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 apperance 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}
\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}
\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 11.1, 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} \hline
 ...
\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 11.1:

```
\newcommand{\mycaption}[2][\@empty]{
  \captionnamefont{\sffamily\hfill}
  \captiondelim{\hfill}
  \captionstyle{\centerlastline\\}
  \captiontitlefont{\scshape}
  \setlength{\belowcaptionskip}{10pt}
  \ifx #1\@empty \caption{#2}\else \caption[#1]{#2}}
```

NOTE: Any code that involves the `@` sign must be either in a package (`.sty`) file or enclosed between a `\makeatletter` ... `\makeatother` pairing.

The code for the Table 11.1 example can now be written as:

```
\begin{table}
\centering
\mycaption{Redesigned table caption style} \label{tab:style}
\begin{tabular}{lr} \hline
 ...
\end{table}
```

Note that in the code for `\mycaption` I have added two `\hfill` commands and `\centerlastline` compared with the original specification. It turned out that the original definitions worked for a single line caption but not for a multiline caption. The additional commands makes it work in both cases, forcing the name to be centered as well as the last line of a multiline title, thus giving a balanced appearence.

Continuation captions and legends

```
\contcaption{⟨text⟩}
```

The `\contcaption` command can be used to put a ‘continuation’ 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 11.2 illustrates the use of the `\contcaption` command. The table was produced from the following code.

Table 11.2: A multi-part table

just a single line	1

Table 11.2: Continued

just a single line	2

Table 11.2: Concluded

just a single line	3

```
\begin{table}
\centering
\caption{A multi-part table} \label{tab:m}
\begin{tabular}{lc} \hline
 just a single line & 1 \\ \hline
\end{tabular}
\end{table}

\begin{table}
\centering
\contcaption{Continued}
\begin{tabular}{lc} \hline
 just a single line & 2 \\ \hline
\end{tabular}
\end{table}

\begin{table}
\centering
\contcaption{Concluded}
\begin{tabular}{lc} \hline
 just a single line & 3 \\ \hline
\end{tabular}
\end{table}
```

`\legend{⟨text⟩}`

The `\legend` command is intended to be used to put an anonymous caption into a float environment, but may be used anywhere.

For example, the following code was used to produce the two-line Table 11.3. The `\legend` command can be used within a float independently of any `\caption` command.

```
\begin{table}
\centering
\caption{Another table} \label{tab:legend}
\begin{tabular}{lc} \hline
 A legendary table & 5 \\
```

Table 11.3: Another table

A legendary table	5
with two lines	6

The legend

Legendary table

An anonymous table	5
with two lines	6

```
 with two lines    & 6 \\ \hline
\end{tabular}
\legend{The legend}
\end{table}
```

Title legend

This is a marginal note with a legend.

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{Title legend}
           This is a marginal note with a legend.}
```

If you want the legend text to be included in the 'List of...' use the `\addcontentsline` command in conjunction with the `\legend`. For example:

```
\addcontentsline{lot}{table}{Titling text} % left justified
\addcontentsline{lot}{table}{\protect\numberline{}Titling text} % indented
```

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} \hline
 An anonymous table & 5 \\
 with two lines     & 6 \\ \hline
\end{tabular}
\end{table}
```

Look at the List of Tables to see how the two forms of `\addcontentsline` are typeset.

As with the `\caption` command, the spacing before and after a legend is controlled by the `\abovecaptionskip` and `\belowcaptionskip` lengths.

`\namedlegend[`⟨*short-title*⟩`]{`⟨*long-title*⟩`}`

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} \hline
seven & VII \\
eight & VIII \\ \hline
\end{tabular}
\end{table}
```

```
\flegtype{⟨name⟩}
\flegtoctype{⟨title⟩}
```

The macro `\flegtype`, where `type` is the name of a float environment (e.g., `table`) 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, namely:

```
\newcommand{\flegtable}{\tablename}
\newcommand{\flegfigure}{\figurename}
```

which may be altered via `\renewcommand` if desired. The macro `\flegtoctype`, where again `type` 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. . . '. By default it is defined to do nothing, and can be changed via `\renewcommand`. For instance, it could be changed for tables as:

```
\renewcommand{\flegtoctable}[1]{
  \addcontentsline{lot}{table}{#1}}
```

The `\legend` command produces a plain, unnumbered heading. It can also be useful sometimes to have named and numbered captions outside a floating environment, perhaps in a `minipage` if you want the table or picture to appear at a precise location in your document.

```
\newfixedcaption[⟨capcommand⟩]{⟨command⟩}{⟨float⟩}
\renewfixedcaption[⟨capcommand⟩]{⟨command⟩}{⟨float⟩}
\providefixedcaption[⟨capcommand⟩]{⟨command⟩}{⟨float⟩}
```

The `\newfixedcaption` command, and its friends, can be used to create a new 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*⟩. The `\renewfixedcaption` and `\providefixedcaption` commands take the same arguments as `\newfixedcaption`; the three commands are analagous to those in the `\newcommand` family.

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 another example, 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 `\newfixedcaption` 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 required, 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 might need adjusting to meet your particular requirements. The `\cleartoevenpage` command ensures that you get to the next even-numbered page (the `\cleardoublepage` gets you to the next odd-numbered page and `\clearpage` gets you to the next page which may be odd or even).

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

EXAMPLE FIGURE WITH BITWONUMCAPTION

Figure 11.1: Long `\bitwonumcaption`

Bild 11.1: Lang `\bitwonumcaption`

EXAMPLE FIGURE WITH BIONENUMCAPTION

Figure 11.2: Long English `\bionenumcaption`

Bild 11.2: Lang Deutsch `\bionenumcaption`

left as an exercise for the document author.

```
\bitwonumcaption[⟨label⟩]{⟨short1⟩}{⟨long1⟩}{⟨NAME⟩}{⟨short2⟩}{⟨long2⟩}
\bionenumcaption[⟨label⟩]{⟨short1⟩}{⟨long1⟩}{⟨NAME⟩}{⟨short2⟩}{⟨long2⟩}
```

Bilingual captions can be typeset by the `\bitwonumcaption` command which has six arguments. The first, optional argument ⟨*label*⟩, is the name of a label, if required. ⟨*short1*⟩ and ⟨*long1*⟩ are the short (i.e., equivalent to the optional argument to the `\caption` command) and long caption texts for the main language of the document. The value of the ⟨*NAME*⟩ argument is used as the caption name for the second language caption, while ⟨*short2*⟩ and ⟨*long2*⟩ are the short and long caption texts for the second language. For example, if the main and secondary languages are English and German and a figure is being captioned:
`\bitwonumcaption{Short}{Long}{Bild}{Kurz}{Lang}`
If the short title text(s) is not required, then leave the appropriate argument(s) either empty or as one or more spaces, like:
`\bitwonumcaption[fig:bi1]{}{Long}{Bild}{  }{Lang}`
Both language texts are entered into the appropriate 'List of. . . ', and both texts are numbered.

Figure 11.1 is an example of using the above code.

The `\bionenumcaption` command takes the same arguments as `\bitwonumcaption`. The difference between the two commands is that `\bionenumcaption` does not number the second language text in the 'List of. . . '. Figure 11.2 is an example of using `\bionenumcaption`.

```
\bicaption[⟨label⟩]{⟨short1⟩}{⟨long1⟩}{⟨NAME⟩}{⟨long2⟩}
```

When bilingual captions are typeset via the `\bicaption` command the second language text is not put into the 'List of. . . '. The command takes 5 arguments. The optional ⟨*label*⟩ is for a label if required. ⟨*short1*⟩ and ⟨*long1*⟩ are the short and long caption texts for the main language of the document. The value of the ⟨*NAME*⟩ argument is used as the caption name for the second language caption. The last argument, ⟨*long2*⟩, is the caption text for the second language (which is not put into the 'List of. . . '). For example, if the main and secondary languages are English and German:
`\bicaption{Short}{Long}{Bild}{Langlauf}`
If the short title text is not required, then leave the appropriate argument either empty or as one or more spaces.

EXAMPLE FIGURE WITH A RULED BICAPTION

Figure 11.3: Longingly

Bild 11.3: Langlauf

Figure 11.3 is an example of using `\bicaption` and was produced by the following code:

```
\begin{figure}
\centering
EXAMPLE FIGURE WITH A RULED BICAPTION
\precaption{\rule{\linewidth}{0.4pt}\par}
\midbicaption{\precaption{}\postcaption{\rule{\linewidth}{0.4pt}}}
\bicaption[fig:bi2]{Short English \cs{bicaption}}{Longingly}{Bild}{Langlauf}
\end{figure}
```

```
\bicontcaption{⟨long1⟩}{⟨NAME⟩}{⟨long2⟩}
```

Bilingual continuation captions can be typeset via the `\bicontcaption` command. In this case, neither language text is put into the 'List of...'. The command takes 3 arguments. ⟨*long1*⟩ is the caption text for the main language of the document. The value of the ⟨*NAME*⟩ argument is used as the caption name for the second language caption. The last argument, ⟨*long2*⟩, is the caption text for the second language. For example, if the main and secondary languages are again English and German:
`\bicontcaption{Continued}{Bild}{Fortgefahren}`

```
\midbicaption{⟨text⟩}
```

The bilingual captions are implemented by calling `\caption` twice, once for each language. The command `\midbicaption`, which is similar to the `\precaption` and `\postcaption` commands, is executed just before calling the second `\caption`. Among other things, this can be used to modify the style of the second caption with respect to the first. For example, if there is a line above and below normal captions, it is probably undesirable to have a double line in the middle of a bilingual caption. So, for bilingual captions the following may be done within the float before the caption:

```
\precaption{\rule{\linewidth}{0.4pt}\par}
\postcaption{}
\midbicaption{\precaption{}\postcaption{\rule{\linewidth}{0.4pt}}}
```

This sets a line before the first of the two captions, then the `\midbicaption`... nulls the pre-caption line and adds a post-caption line for the second caption. The class initially specifies `\midbicaption{}`.

Subcaptions

The subfigure package [Coc02] 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[1] has been specified.

```
\subcaption[⟨list-entry⟩]{⟨subcaption⟩}
```

The `\subcaption` command is similar to the `caption` command. It typesets an identified ⟨*subcaption*⟩, 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 ⟨*subcaption*⟩ is added to the listing. The `\subcaption` macro should only be used in a fixed width box of some sort, for example a `minipage` environment, as the ⟨*subcaption*⟩ will be typeset using a box which is the width of the environment.

For example:

```
\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}
```

If a figure that includes sub-figures is itself continued then it may be desireable to continue the captioning of the sub-figures. For example, if Figure 3 has three sub-figures, say A, B and C, and Figure 3 is continued then the sub-figures in the continuation should be D, E, etc.

```
\contsubcaption[⟨list-entry⟩]{⟨subcaption⟩}
```

The `\contsubcaption` command is the continued version of `\subcaption`.

```
\label(⟨bookmark⟩){⟨key⟩}
\subcaptionref{⟨key⟩}
```

A `\label` command may be included in the ⟨*subcaption*⟩ argument of `\subcaption` or `\contsubcaption`. Using `\ref` to refer to the label will typeset the number of the float (obtained from a labelled `\caption`) and the subcaption identifier. On the other hand, using `\subcaptionref` will typeset just the subcaption identifier.

In case the hyperref package is used, the `\label` command when used inside a ⟨*subcaption*⟩ argument can take an optional argument, *enclosed in parentheses* not *square brackets*, ⟨*bookmark*⟩ which will create a bookmark field of the form ‘Subfigure 3.2(c)’.

[1]See §11.3 on page 147.

```
\tightsubcaptions \loosesubcations
```

The `\tightsubcaptions` declaration will put little vertical space around the subcaption, while `\loosesubcaptions` will add more whitespace. The default is `\tightsubcaptions`.

```
\subcaptionsize{⟨font-size⟩}
\subcaptionlabelfont{⟨fontspec⟩}
\subcaptionfont{⟨fontspec⟩}
```

The size of the font used for subcaptions is set by `\subcaptionsize`. The default is:

```
\subcaptionsize{\footnotesize}
```

The fonts used for the identifier and the subcaption are set by `\subcaptionlabelfont` and `\subcaptionfont`, respectively, where ⟨*fontspec*⟩ is one or more font style declarations (e.g., `\bfseries\slshape` for a bold, slanted font). The defaults are:

```
\subcaptionlabelfont{\normalfont}
\subcaptionfont{\normalfont}
```

```
\subcaptionstyle{⟨style⟩}
```

By default the identifier and title of a subcaption are typeset as a block (non-indented) paragraph. `\subcaptionstyle` 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 `\subcaptionstyle{}` to give the normal block paragraph style.

```
\hangsubcaption
\shortsubcaption
\normalsubcaption
```

The `\hangsubcaption` declaration causes subcaptions to be typeset with the identifier above the title. With the `\shortsubcaption` declaration subcaptions that are less than one line in length are typeset left justified instead of centered. The `\normalsubcaption` command undoes any previous `\hangsubcaption` or `\shortsubcaption` command. The class initially specifies `\normalcaption`.

```
\subtop[⟨list-entry⟩][⟨subcaption⟩]{⟨text⟩}
\subbottom[⟨list-entry⟩][⟨subcaption⟩]{⟨text⟩}
```

The command `\subtop` puts a subcaption number on top of the ⟨*text*⟩. If both optional arguments are present, ⟨*list-entry*⟩ will be added to the appropriate listing, and an alphabetic identifier and ⟨*subcaption*⟩ will be placed above ⟨*text*⟩. If only one optional argument is present, this is treated as ⟨*subcaption*⟩; the identifier and ⟨*subcaption*⟩ are placed above ⟨*text*⟩ and ⟨*subcaption*⟩ is added to the listing. In all cases, the identifier is placed above ⟨*text*⟩ and added to the listing.

The `\subbottom` command is similar, except that the identifier and any ⟨*subcaption*⟩ is placed below the ⟨*text*⟩.

SUBFIGURE ONE
(a) Subfigure 1

SUBFIGURE TWO
(b) Subfigure 2

Figure 11.4: Figure with two subfigures

The main caption can be at either the top or the bottom of the float.

For example:

```
\begin{figure}
\subbottom{...} %  captioned as (a) below
\subtop{...}    %  captioned as (b) above
\caption{...}
\end{figure}
```

As another example, Figure 11.4 and the next paragraph was produced by the code below.

Figure 11.4 has two subfigures, namely 11.4(a) and (b).

```
    Figure \ref{subfig:sf} has two subfigures, namely \ref{sf:1}
and \subcaptionref{sf:2}.
\begin{figure}
\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{subfig:sf}
\end{figure}
```

The major difference between the `\subcaption` command and the `\subtop` and `\subbottom` commands, apart from the ⟨*text*⟩ argument, is that `\subcaption` must be used in a fixed width environment while `\subtop` uses the width of ⟨*text*⟩ for the box in which to typeset the ⟨*subcaption*⟩.

`\contsubtop[`⟨*list-entry*⟩`][`⟨*subcaption*⟩`]{`⟨*text*⟩`}`
`\contsubbottom[`⟨*list-entry*⟩`][`⟨*subcaption*⟩`]{`⟨*text*⟩`}`

The command `\contsubtop` will continue the sub-caption numbering scheme across (continued) floats, putting the ⟨*subcaption*⟩ at the top of the ⟨*text*⟩. The `\contsubbottom` command is similar but puts the ⟨*subcaption*⟩ at the bottom of the ⟨*text*⟩. In either case, the main caption can be at the top or bottom of the float.

For example:

```
\begin{table}
\caption{...}
\subtop{...}  % captioned as (a) above
\subtop{...}  % captioned as (b) above
\end{table}
...
\begin{table}
\contcaption{Concluded}
```

```
\contsubtop{...} % captioned as (c) above
\contsubtop{...} % captioned as (d) above
\end{table}
```

As with the `\subcaption` command, a `\label` command may be used in either the ⟨*subcaption*⟩ or the ⟨*text*⟩ arguments to `\subtop` and friends.

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 without having to use any package. 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.

The LaTeX kernel provides tools to help in the definition of captions, but it is the particular class that decides on their format.

`\caption[`⟨*short*⟩`]{`⟨*long*⟩`}`

The kernel (in `ltfloat.dtx`) defines the caption command via

```
\def\caption{\refstepcounter\@captype \@dblarg{\@caption\@captype}}
```

`\@captype`

`\@captype` is defined by the code that creates a new float environment and is set to the environment's name (see the code for `\@xfloat` in `ltfloat.dtx`). For a `figure` environment, there is an equivalent to

```
\def\@captype{figure}
```

`\@caption{`⟨*type*⟩`}[`⟨*short*⟩`]{`⟨*long*⟩`}`

The kernel also provides the `\@caption` macro as:

```
\long\def\@caption#1[#2]#3{
  \par
  \addcontentsline{\csname ext@#1\endcsname}{#1}    <--
    {\protect\numberline{\csname the#1\endcsname}{\ignorespaces #2}}
  \begingroup
     \@parboxrestore
     \if@minipage
       \@setminipage
     \fi
     \normalsize
     \@makecaption{\csname fnum@#1\endcsname}{\ignorespaces #3}\par  <--
  \endgroup
```

where ⟨*type*⟩ is the name of the environment in which the caption will be used. Putting these three commands together results in the user's view of the caption command as `\caption[`⟨*short*⟩`]{`⟨*long*⟩`}`.

It is the responsibilty of the class (or package) which defines floats to provide definitions for `\ext@type`, `\fnum@type` and `\@makecaption` which appear in the definition of `\@caption` (in the lines marked `<--` above).

`\ext@type`

This macro holds the name of the extension for a 'List of. . . ' file. For example for the `figure` float environment there is the definition equivalent to

```
\newcommand{\ext@figure}{lof}
```

`\fnum@type`

This macro is responsible for typesetting the caption number. For example, for the `figure` environment there is the definition equivalent to

```
\newcommand{\fnum@figure}{\figurename~\thefigure}
```

`\@makecaption{`⟨*number*⟩`}{`⟨*text*⟩`}`

The `\@makecaption` macro, where ⟨*number*⟩ is a string such as 'Table 5.3' and ⟨*text*⟩ is the caption text, performs the typesetting of the caption, and is defined in the standard classes (in `classes.dtx`) as the equivalent of:

```
\newcommand{\@makecaption}[2]{
  \vskip\abovecaptionskip         <- 1
  \sbox\@tempboxa{#1: #2}        <- 2
  \ifdim \wd\@tempboxa >\hsize
    #1: #2\par                    <- 3
  \else
    \global \@minipagefalse
    \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}
  \fi
  \vskip\belowcaptionskip}        <- 4
```

`\abovecaptionskip \belowcaptionskip`

Vertical space is added before and after a caption (lines marked 1 and 4 in the code for `\@makecaption` above) and the amount of space is given by the lengths `\abovecaptionskip` and `\belowcaptionskip`. The standard classes set these to 10pt and 0pt respectively. If you want to change the space before or after a caption, use `\setlength` to change the values. In figures, the caption is usually placed below the illustration. The actual space between the bottom of the illustration and the baseline of the first line of the caption is the `\abovecaptionskip` plus the `\parskip` plus the `\baselineskip`. If the illustration is in a `center` environment then additional space will be added by the `\end{center}`; it is usually better to use the `\centering` command rather than the `center` environment.

The actual typesetting of a caption is effectively performed by the code in lines marked 2 and 3 in the code for `\@makecaption`; note that these are where the colon that is typeset after the number is specified. If you want to make complex changes to the default captioning

A THOUSAND WORDS…

Figure 11.5: A picture is worth a thousand words

ANOTHER THOUSAND WORDS…

Figure 11.6 — A different kind of figure caption

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. 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 11.5 was created by the following code:

```
\makeatletter
\renewcommand{\fnum@figure}{\textsc{\figurename~\thefigure}}
\makeatother
\begin{figure}
\centering
A THOUSAND WORDS\ldots
\caption{A picture is worth a thousand words}\label{fig:sc}
\end{figure}
```

As another example, suppose that you needed to typeset the `\figurename` and its number in a bold font, replace the colon that normally appears after the number by a long dash, and typeset the actual title text in a sans-serif font, as is illustrated by the caption for Figure 11.6. 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.

Captions with footnotes

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 LoF/lot as appropriate. If the argument is not present, then the text of the required argument is put into the LoF. 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 `\protect`ed (i.e., `\protect\footnote{}`) if it is used in a moving argument. If you don't want the footnote to appear in the LoF, 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.

11.3 Floats

New float environments

```
\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 ⟨*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}
\renewcommmand{\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{X}{Z}{capname}
```

so `X` is the name of the float environment and also the name of the counter for the caption, and `Z` is the file extension. The following float environment and related commands are then created.

```
\begin{X} float material \end{X}
\begin{X*} float material \end{X*}
```

The new float environment is called `X`, and can be used as either `\begin{X}` or `\begin{X*}`, with the matching `\end{X}` or `\end{X*}`. It is given the standard default position specification of `tbp`.

```
Zdepth
```

The `Zdepth` 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 `Zdepth`. The default definition is `\setcounter{Zdepth}{1}`. To have a subfloat of `Z` appear in the listing do `\setcounter{Zdepth}{2}`.

As a fuller 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}
\newfixedcaption{\fdiagcaption}{diagram}
\newlistentry{diagram}{dgm}{0}
\begin{document}
...
\listoffigures
\listfofdiagrams
...
\begin{diagram}
\caption{A diagram} \label{diag1}
...
\end{diagram}
As diagram~\ref{diag1} shows ...
\begin{minipage}{.9\textwidth}
\fdiagcaption{Another diagram} \label{diag2}
...
\end{minipage}

In contrast to diagram~\ref{diag1}, diagram~\ref{diag2} provides ...
```

As a word of warning, if you mix both floats and fixed environments with the same kind of caption you have to ensure that they get printed in the correct order in the final document. If you do not do this, then the 'List of. . . ' captions will come out in the wrong order (the lists are ordered according the page number in the typeset document, *not* your source input order).

New subfloats

The subfigure package defines the `subfigure` and `subtable` subfloats. The class does not define any subfloats, if you need them you have to specify them yourself.

```
\newsubfloat{⟨fenv⟩}
```

The `\newsubfloat` command enables the subcaptioning commands (`\subcaption`, `\subtop`, etc.) to be used within the float environment ⟨*fenv*⟩ previously defined via `\newfloat`.

Calling `\newsubfloat{fenv}` will, among other things, create a new counter called `subfenv` and the command `\thesubfenv` for typesetting the counter. The default definition of `\thesubfenv` is equivalent to:

```
\newcommand{\thesubfenv}{(\alph{subfenv})}
```

which typesets a lowercase letter enclosed in parentheses. This is the identifier for subcaptions, and may be changed via `\renewcommand`.

If you are not using the subfigure package and want, say, subfigures then in the preamble call:

ILLUSTRATION 1 ILLUSTRATION 2

Figure 11.7 — Float with two illustrations

```
\newsubfloat{figure}
```

and if you want the subcaptions to appear in the List of Figures, put:

```
\setcounter{lofdepth}{2}
```

between the `\begin{document}` and `\listoffigures` commands.

Further, if you had some subfigures that were originally planned for use with the subfigure package and wanted to use these but without the package, you could:

```
\let\subfigure\subbottom
```

which will make `\subfigure` an alias for `\subbottom`.

Multiple floats

As far as LaTeX is concerned, a float is a box with certain restrictions as to where it can be placed. 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 put 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.

The subfigure package is designed for the last of these cases; the others do not require a package.

Figure 11.7 is an example of multiple illustrations in a single float with a single caption. This figure was produced by the following code.

```
\begin{figure}
\centering
\hspace*{\fill}
  {ILLUSTRATION 1} \hfill {ILLUSTRATION 2}
\hspace*{\fill}
\caption{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 11.8 and 11.9 which are examples of two separately captioned illustrations in one float.

```
\begin{figure}
\centering
\begin{minipage}{0.4\textwidth}
  \centering
  ILLUSTRATION 3
  \caption{Illustration 3} \label{fig:mult2}
\end{minipage}
```

ILLUSTRATION 3

Figure 11.8 — Illustration 3

ILLUSTRATION 4

Figure 11.9 — Illustration 4

```
\hfill
\begin{minipage}{0.4\textwidth}
  \centering
  ILLUSTRATION 4
  \caption{Illustration 4} \label{fig:mult3}
\end{minipage}
\end{figure}
```

In this case the illustrations (or graphics or tabulars) are put into separate `minipage` environments within the float, and the captions are also put within the `minipage`s. Note that any required `\label` must also be inside the `minipage`. If you wished, you could add yet another caption after the end of the two `minipage`s.

Keith Reckdahl [Rec97] provides more examples of this kind of thing.

Where LaTeX puts floats

The general format for a float environment is:
`\begin{float}[`⟨*loc*⟩`] ... \end{float}` or for double column floats:
`\begin{float*}[`⟨*loc*⟩`] ... \end{float*}`
where the optional argument ⟨*loc*⟩, consisting of one or more characters, specifies a location where the float may be placed. Note that the multicol package [Mit98] only supports the starred floats and it will not let you have a single column float. The possible ⟨*loc*⟩ values are one or more of the following:

- `b` *bottom*: at the bottom of a page. This does not apply to double column floats as they may only be placed at the top of a page.
- `h` *here*: if possible exactly where the float environment is defined. It does not apply to double column floats.
- `p` *page*: on a separate page containing only floats (no text).
- `t` *top*: at the top of a page.
- `!` make an extra effort to place the float at the earliest place specified by the rest of the argument.

The default for ⟨*loc*⟩ is `tbp`, so the float may be placed at the top, or bottom, or on a float-only 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*[2]. A float is never put on an earlier page than its definition but may be put on the same or later page of its definition. If a float cannot be placed, all suceeding floats will be held up, and LaTeX can store no more than 16 held up floats. A float cannot be placed if it would cause an overfull page, or it otherwise cannot be fitted according the the float parameters. A `\clearpage` or `\cleardoublepage` or `\end{document}` flushes out all unprocessed floats, irrespective of the ⟨*loc*⟩ and float parameters, putting them on float-only pages.

[2]This little quirk is fixed by the fixltx2e package, at least for tables and figures. The package is part of a normal LaTeX distribution.

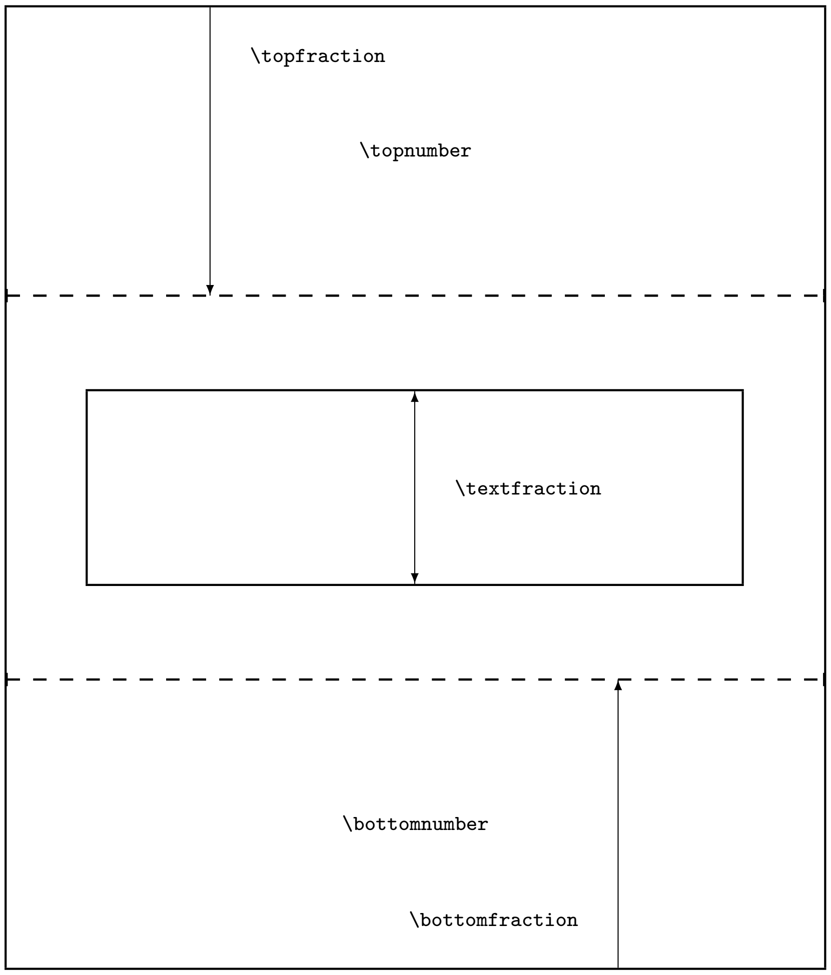


Figure 11.10 — Float and text page parameters

```
\suppressfloats[⟨pos⟩]
```

You can use the command `\suppressfloats` to suppress floats at a given ⟨*pos*⟩ on the current page. `\suppressfloats[t]` prevents any floats at the top of the page and `\suppressfloats[b]` prevents any floats at the bottom of the page. The simple `\suppressfloats` prevents both top and bottom floats.

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 11.10 and 11.11 illustrate the many float parameters and Table 11.4 lists the float parameters and the typical standard default values. All the lengths are rubber lengths,

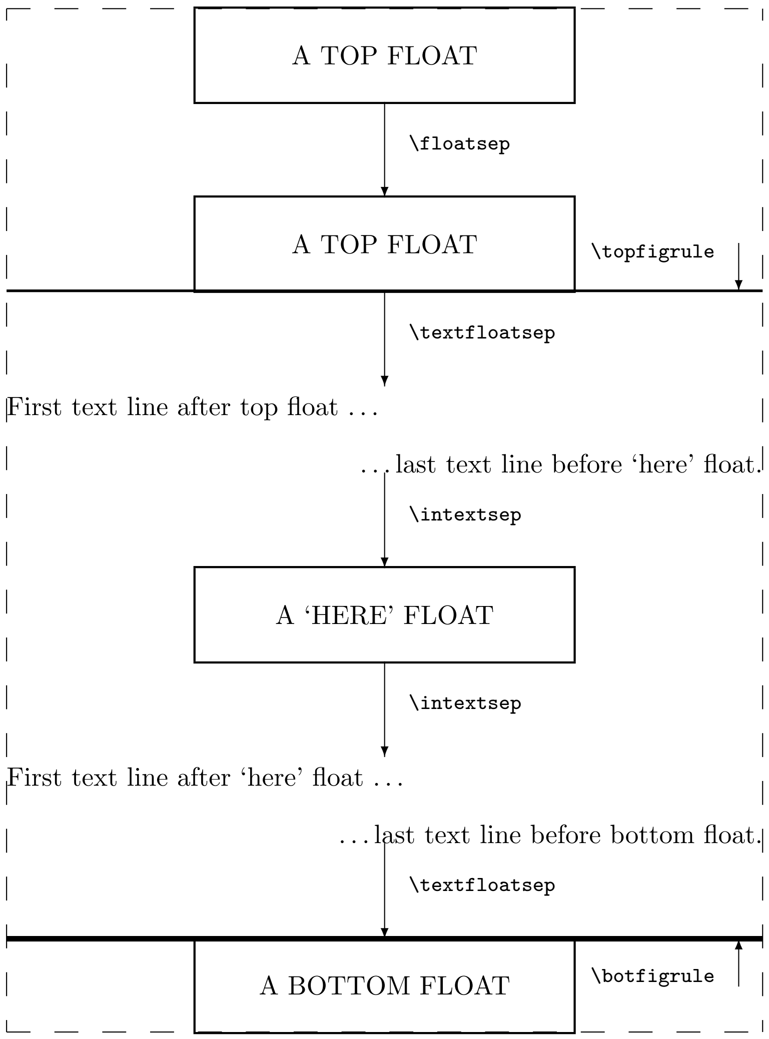

A TOP FLOAT
\floatsep
A TOP FLOAT
\topfigrule
\textfloatsep
First text line after top float ...
... last text line before 'here' float.
\intextsep
A 'HERE' FLOAT
\intextsep
First text line after 'here' float ...
... last text line before bottom float.
\textfloatsep
\botfigrule
A BOTTOM FLOAT

Figure 11.11 — Float parameters

Table 11.4: Float placement parameters

Parameter	Controls	Default
	Counters — change with `\setcounter`	
`topnumber`	max number of floats at top of a page	2
`bottomnumber`	max number of floats at bottom of a page	1
`totalnumber`	max number of floats on a text page	3
`dbltopnumber`	like `topnumber` for double column floats	2
	Commands — change with `\renewcommand`	
`\topfraction`	max fraction of page reserved for top floats	0.7
`\bottomfraction`	max fraction of page reserved for bottom floats	0.3
`\textfraction`	min fraction of page that must have text	0.2
`\dbltopfraction`	like `\topfraction` for double column floats	0.7
`\floatpagefraction`	min fraction of a float page that must have float(s)	0.5
`\dblfloatpagefraction`	like `\floatpagefraction` for double column floats	0.5
	Text page lengths — change with `\setlength`	
`\floatsep`	vertical space between floats	12pt
`\textfloatsep`	vertical space between a top (bottom) float and suceeding (preceding) text	20pt
`\intextsep`	vertical space above and below an `h` float	12pt
`\dblfloatsep`	like `\floatsep` for double column floats	12pt
`\dbltextfloatsep`	like `\textfloatsep` for double column floats	20pt
	Float page lengths — change with `\setlength`	
`\@fptop`	space at the top of the page	`0pt plus 1fil`
`\@fpsep`	space between floats	`8pt plus 2fil`
`\@fpbot`	space at the bottom of the page	`0pt plus 1fil`
`\@dblfptop`	like `\@fptop` for double column floats	`0pt plus 1fil`
`\@dblfpsep`	like `\@fpsep` for double column floats	`8pt plus 2fil`
`\@dblfpbot`	like `\@fpbot` for double column floats	`0pt plus 1fil`

and the actual defaults depend on both the class and its size option.

Given the displayed defaults, the height of a top float must be less than 70% of the textheight and there can be no more than 2 top floats on a text page. Similarly, the height of a bottom float must not exceed 30% of the textheight and there can be no more than 1 bottom float on a text page. There can be no more than 3 floats (top, bottom and here) on the page. At least 20% of a text page with floats must be text. On a float page (one that has no text, only floats) the sum of the heights of the floats must be at least 50% of the textheight. The floats on a float page should be vertically centered.

It can be seen that with the defaults LaTeX might have trouble finding a place for a float. Consider what will happen if a float is a bottom float whose 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 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 can 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` to `0pt`, `\@fpsep` to `8pt` and `\@fpbot` to `0pt plus 1fil` will force floats on a float page to start at the top 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, don't bother 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.

12 Rows and columns

12.1 Introduction

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 [Car99a], and booktabs [Fea03] packages. Much of the material in this chapter strongly reflects the documentation of these packages.

However, new kinds of tabular environments are also provided.

12.2 General

```
\[ \begin{array}[⟨pos⟩]{⟨preamble⟩} rows \end{array} \]
\begin{tabular}[⟨pos⟩]{⟨preamble⟩} rows \end{tabular}
\begin{tabular*}{⟨width⟩}[⟨pos⟩]{⟨preamble⟩} rows \end{tabular*}
\begin{tabularx}{⟨width⟩}[⟨pos⟩]{⟨preamble⟩} rows \end{tabularx}
```

The `array` and `tabular` environments are traditional and the others are extensions. The `array` is for typesetting math and has to be within a math environment of some kind. The `tabular` series are for typesetting ordinary text.

The optional ⟨*pos*⟩ argument can be one of `t`, `c`, or `b` (the default is `c`), and controls the vertical position of the array or tabular; either the top or the center, or the bottom is aligned with the baseline. Each row consists of elements separated by &, and finished with `\\`. There may be as many rows as desired. The number and style of the columns is specified by the ⟨*preamble*⟩. The width of each column is wide enough to contain its longest entry and the overall width of the `array` or `tabular` is sufficient to contain all the columns. However, the `tabular*` and `tabularx` provide more control over the width through their ⟨*width*⟩ argument.

12.3 The preamble

You use the ⟨*preamble*⟩ argument to the array and tabular environments to specify the number of columns and how you want column entries to appear. The preamble consists of a sequence of options, which are listed in Table 12.1.

Examples of the options include:

- A flush left column with bold font can be specified with `>{\bfseries}l`.

```
\begin{center}
\begin{tabular}{>{\large}c >{\large\bfseries}l >{\large\itshape}c} \toprule
```

Table 12.1: The array and tabular preamble options.

`l`	Left adjusted column.
`c`	Centered adjusted column.
`r`	Right adjusted column.
`p{`⟨*width*⟩`}`	Equivalent to `\parbox[t]{`⟨*width*⟩`}`.
`m{`⟨*width*⟩`}`	Defines a column of width ⟨*width*⟩. Every entry will be centered in proportion to the rest of the line. It is somewhat like `\parbox{`⟨*width*⟩`}`.
`b{`⟨*width*⟩`}`	Coincides with `\parbox[b]{`⟨*width*⟩`}`.
`>{`⟨*decl*⟩`}`	Can be used before an `l`, `r`, `c`, `p`, `m` or a `b` option. It inserts ⟨*decl*⟩ directly in front of the entry of the column.
`<{`⟨*decl*⟩`}`	Can be used after an `l`, `r`, `c`, `p{..}`, `m{..}` or a `b{..}` option. It inserts ⟨*decl*⟩ right after the entry of the column.
`|`	Inserts a vertical line. The distance between two columns will be enlarged by the width of the line.
`@{`⟨*decl*⟩`}`	Suppresses inter-column space and inserts ⟨*decl*⟩ instead.
`!{`⟨*decl*⟩`}`	Can be used anywhere and corresponds with the `|` option. The difference is that ⟨*decl*⟩ is inserted instead of a vertical line, so this option doesn't suppress the normally inserted space between columns in contrast to `@{...}`.
`D{`⟨*ssep*⟩`}{`⟨*osep*⟩`}{`⟨*places*⟩`}`	Column entries aligned on a 'decimal point'

```
A   & B  & C \\
100 & 10 & 1 \\ \bottomrule
\end{tabular}
\end{center}
```

A	**B**	*C*
100	**10**	*1*

- In columns which have been generated with `p`, `m` or `b`, the default value of `\parindent` is `0pt`.

```
\begin{center}
\begin{tabular}{m{1cm}m{1cm}m{1cm}} \toprule
1 1 1 1 1 1 1 1 1 1 1 1 &
2 2 2 2 2 2 2 2         &
3 3 3 3                 \\ \bottomrule
\end{tabular}
\end{center}
```

1 1 1 1
1 1 1 1
1 1 1 1
	2 2 2 2
	2 2 2 2
		3 3 3 3

The `\parindent` 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   1 2 3 4 5 6 7
6 7 8 9 0 1 2   8 9 0 1 2 3 4
3 4 5 6 7 8 9   5 6 7 8 9 0
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.

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{`⟨*ssep*⟩`}{`⟨*osep*⟩`}{`⟨*places*⟩`}`

⟨*ssep*⟩ is the single character which is used as the separator in the source `.tex` file. Thus it will usually be '`.`' or '`,`'.

⟨*osep*⟩ is the separator in the output, this may be the same as the first argument, but may be any math-mode expression, such as `\cdot`. A `D` column always uses math mode for the digits as well as the separator.

⟨*places*⟩ should be the maximum number of decimal places in the column (but see below for more on this). If this is negative, any number of decimal places can be used in the column, and all entries will be centred on (the leading edge of) the separator. Note that this can cause a column to be too wide; for instance, compare the first two columns in the example below.

Here are some example specifications which, for convenience, employ the `\newcolumntype` macro described later.

```
\newcolumntype{d}[1]{D{.}{\cdot}{#1}}
```

This defines `d` to be a column specifier taking a single argument specifying the number of decimal places, and the `.tex` file should use '`.`' as the separator, with `\cdot` ($\cdot$) being used in the output.

```
\newcolumntype{.}{D{.}{.}{-1}}
```

The result of this is that '`.`' specifies a column of entries to be centered on the '.'.

```
\newcolumntype{,}{D{,}{,}{2}}
```

And the result of this is that '`,`' specifies a column of entries with at most two decimal places after a ','.

The following table is typeset from this code:

```
\begin{center}
 \begin{tabular}{|d{-1}|d{2}|.|,|}
 1.2   & 1.2   &1.2    &1,2    \\
 1.23  & 1.23  &12.5   &300,2  \\
 1121.2& 1121.2&861.20 &674,29 \\
 184   & 184   &10     &69     \\
 .4    & .4    &       &,4     \\
       &       &.4     &
 \end{tabular}
 \end{center}
```

1·2	1·2	1.2	1,2
1·23	1·23	12.5	300,2
1121·2	1121·2	861.20	674,29
184	184	10	69
·4	·4		,4
		.4	

Note that the first column, which had a negative ⟨*places*⟩ argument is wider than the second column, so that the decimal point appears in the middle of the column.

The third ⟨*places*⟩ argument may specify *both* the number of digits to the left and to the right of the decimal place. The third column in the next table below is set with `D{.}{.}{5.1}` and in the second table, `D{.}{.}{1.1}`, to specify 'five places to the left and one to the right' and 'one place to the left and one to the right' respectively. (You may use ',' or other characters, not necessarily '.' in this argument.) The column of figures is then positioned such that a number with the specified numbers of digits is centred in the column.

If you have table headings (inserted with `\multicolumn{1}{c}{...}` to over-ride the `D` column type) then it may be that neither of the above 'centred' or 'right aligned' forms is quite what you want.

```
 \begin{center}\small
 \begin{tabular}[t]{|D..{-1}|D..{1}|D..{5.1}|}
\multicolumn{1}{|c|}{head}&
\multicolumn{1}{c|}{head}&
\multicolumn{1}{c|}{head}\\[3pt]
 1.2  & 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  & 1.2  &1.2 \\
 .4    & .4    &.4
 \end{tabular}
 \end{center}
```

head	head	head
1.2	1.2	1.2
11212.2	11212.2	11212.2
.4	.4	.4

wide heading	wide heading	wide heading
1.2	1.2	1.2
.4	.4	.4

In both of these tables the first column is set with `D{.}{.}{-1}` to produce a column centered on the '`.`', and the second column is set with `D{.}{.}{1}` to produce a right aligned column.

The centered (first) column produces columns that are wider than necessary to fit in the numbers under a heading as it has to ensure that the decimal point is centered. The right aligned (second) column does not have this drawback, but under a wide heading a column of small right aligned figures is somewhat disconcerting.

The notation for the ⟨*places*⟩ argument also enables columns that are centred on the mid-point of the separator, rather than its leading edge; for example `D{+}{\,\pm\,}{3,3}` will give a symmetric layout of up to three digits on either side of a ± sign.

Defining new column specifiers

You can easily type

`>{`⟨*some declarations*⟩`}{c}<{`⟨*some more declarations*⟩`}`

when you have a one-off column in a table, but it gets tedious if you often use columns of this form. The `\newcolumntype` lets you define a new column option like, say

`\newcolumntype{x}{>{`⟨*some declarations*⟩`}{c}<{`⟨*some more declarations*⟩`}}`

and you can then use the `x` column specifier in the preamble wherever you want a column of this kind.

```
\newcolumntype{⟨char⟩}[⟨nargs⟩]{⟨spec⟩}
```

The ⟨*char*⟩ argument is the character that identifies the option and ⟨*spec*⟩ is its specification in terms of the regular preamble options. The `\newcolumntype` command is similar to `\newcommand` — ⟨*spec*⟩ itself can take arguments with the optional ⟨*nargs*⟩ argument declaring their number.

For example, it is commonly required to have both math-mode and text columns in the same alignment. Defining:

```
\newcolumntype{C}{>{$}c<{$}}
\newcolumntype{L}{>{$}l<{$}}
\newcolumntype{R}{>{$}r<{$}}
```

Then `C` can be used to get centred text in an `array`, or centred math-mode in a `tabular`. Similarly `L` and `R` are for left- and right-aligned columns.

The ⟨*spec*⟩ in a `\newcolumntype` command may refer to any of the primitive column specifiers (see table 12.1 on page 156), 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.

Notes

- A preamble of the form `{wx*{0}{abc}yz}` is treated as `{wxyz}`
- An incorrect positional argument, such as `[Q]`, is treated as `[t]`.
- A preamble such as `{cc*{2}}` with an error in a $*$-form will generate an error message that is not particularly helpful.
- Error messages generated when parsing the column specification refer to the preamble argument *after* it has been re-written by the `\newcolumntype` system, not to the preamble entered by the user.
- Repeated `<` or `>` constructions are allowed. `>{`⟨*decs1*⟩`}>{`⟨*decs2*⟩`}` is treated the same as `>{`⟨*decs2*⟩`}{`⟨*decs1*⟩`}`.
  The treatment of multiple `<` or `>` declarations may seem strange. Using the obvious ordering of `>{`⟨*decs1*⟩`}{`⟨*decs2*⟩`}` has the disadvantage of not allowing the settings of a `\newcolumntype` defined using these declarations to be overriden.
- The `\extracolsep` command may be used in `@`-expressions as in standard LaTeX, and also in `!`-expressions.
  The use of `\extracolsep` is subject to the following two restrictions. There must be at most one `\extracolsep` command per `@`, or `!` expression and the command must be directly entered into the `@` expression, not as part of a macro definition. Thus
  ```
    \newcommand{\ef}{\extracolsep{\fill}} ... @{\ef}
  ```
  does not work. However you can use something like
  ```
    \newcolumntype{e}{@{\extracolsep{\fill}}
  ```
  instead.
- As noted by the LaTeX book, a `\multicolumn`, with the exception of the first column, consists of the entry and the *following* inter-column material. This means that in a tabular with the preamble `|l|l|l|l|` input such as `\multicolumn{2}{|c|}` in anything other than the first column is incorrect.
  In the standard array/tabular implementation this error is not noticeable as a `|` takes no horizontal space. But in the class the vertical lines take up their natural width and you will see two lines if two are specified — another reason to avoid using `|`.

12.4 The array environment

Mathematical arrays are usually produced using the `array` environment.

```
\[ \begin{array}[⟨pos⟩]{⟨preamble⟩} rows \end{array} \]
\[ \begin{array}[⟨pos⟩]⟨left⟩{⟨preamble⟩}⟨right⟩ rows \end{array} \]
```

Math formula are usually centered in the columns, but a column of numbers often looks best flush right, or aligned on some distinctive feature. In the latter case the `D` column scheme is very handy.

```
\[ \begin{array}{lcr}
   a +b +c & d - e - f & 123 \\
   g-h     &  j k      & 45 \\
    l      &   m       & 6
  \end{array} \]
```

$$\begin{array}{lcr} a+b+c & d-e-f & 123 \\ g-h & jk & 45 \\ l & m & 6 \end{array}$$

Arrays are often enclosed in brackets or vertical lines or brackets or other symbols to denote math constructs like matrices. The delimeters are often large and have to be indicated using `\left` and `\right` commands.

```
\[ \left[ \begin{array}{cc}
         x_{1} & x_{2} \\
         x_{3} & x_{4}
         \end{array} \right] \]
```

$$\left[\begin{array}{cc} x_{1} & x_{2} \\ x_{3} & x_{4} \end{array} \right]$$

The `array` environment is an extension of the standard environment in that it provides a a system of implicit `\left \right` pairs. If you want an array surrounded by parentheses, you can enter:

```
 \[  \begin{array}({cc})a&b\\c&d\end{array}   \]
```

$$\left(\begin{array}{cc} a & b \\ c & d \end{array} \right)$$

Or, a litle more complex

```
\[ \begin{array}({c})
     \begin{array}|{cc}|
       x_{1} & x_{2} \\
       x_{3} & x_{4}
     \end{array} \\
         y \\
         z
   \end{array} \]
```

$$\left(\begin{array}{c} \left| \begin{array}{cc} x_{1} & x_{2} \\ x_{3} & x_{4} \end{array} \right| \\ y \\ z \end{array} \right)$$

Similarly an environment equivalent to plain TeX's `\cases` could be defined by:

```
\[  f(x)=\begin{array}\{{lL}.
          0        &if $x=0$\\
          \sin(x)/x&otherwise
          \end{array}  \]
```

$$f(x) = \left\{ \begin{array}{ll} 0 & \text{if } x = 0 \\ \sin(x)/x & \text{otherwise} \end{array} \right.$$

Here L denotes a column of left aligned L-R text, as described earlier. Note that as the delimiters must always be used in pairs, the '`.`' must be used to denote a 'null delimiter'.

This feature is especially useful if the `[t]` or `[b]` arguments are also used. In these cases the result is not equivalent to surrounding the environment by `\left`...`\right`, as can be seen from the following example:

```
\begin{array}[t]({c}) 1\\2\\3 \end{array}
\begin{array}[c]({c}) 1\\2\\3 \end{array}
\begin{array}[b]({c}) 1\\2\\3 \end{array}
\quad\mbox{not}\quad
\left(\begin{array}[t]{c} 1\\2\\3 \end{array}\right)
\left(\begin{array}[c]{c} 1\\2\\3 \end{array}\right)
\left(\begin{array}[b]{c} 1\\2\\3 \end{array}\right)
```

$$\begin{array}[t]({c}) 1\\2\\3 \end{array} \begin{array}[c]({c}) 1\\2\\3 \end{array} \begin{array}[b]({c}) 1\\2\\3 \end{array} \quad\text{not}\quad \left(\begin{array}[t]{c} 1\\2\\3 \end{array}\right) \left(\begin{array}[c]{c} 1\\2\\3 \end{array}\right) \left(\begin{array}[b]{c} 1\\2\\3 \end{array}\right)$$

12.5 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 often 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 the columns belonging 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.

No mention has been made of any 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

Table 12.2: Two views of one table

(a) Before

gnats	gram	$13.65
	each	.01
gnu	stuffed	92.50
emu		33.33
armadillo	frozen	8.99

(b) After

Item		
Animal	Description	Price ($)
Gnat	per gram	13.65
	each	0.01
Gnu	stuffed	92.50
Emu	stuffed	33.33
Armadillo	frozen	8.99

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's that the visual clutter should be eliminated.

For example, in Table 12.2, Table 12.2(a) is from the LaTeX book and Table 12.2(b) is how Simon Fear [Fea03] suggests it should be cleaned up.

```
\begin{table}
\centering
\caption{Two views of one table} \label{tab:2views}
\subtop[Before]{\label{tab:before}%
 \begin{tabular}{||l|lr||} \hline
 gnats     & gram      & \$13.65 \\ \cline{2-3}
           & each      & .01 \\ \hline
 gnu       & stuffed   & 92.50 \\ \cline{1-1} \cline{3-3}
 emu       &           & 33.33 \\ \hline
 armadillo & frozen    & 8.99 \\ \hline
 \end{tabular}}
\hfill
\subtop[After]{\label{tab:after}%
 \begin{tabular}{@{}llr@{}} \toprule
 \multicolumn{2}{c}{Item} \\ \cmidrule(r){1-2}
 Animal & Description & Price (\$)\\ \midrule
 Gnat  & per gram  & 13.65 \\
       & each      & 0.01 \\
 Gnu   & stuffed   & 92.50 \\
 Emu   & stuffed   & 33.33 \\
 Armadillo & frozen & 8.99 \\ \bottomrule
 \end{tabular}
}
\end{table}
```

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.

In a simple case a table begins with a `\toprule`, has a single row of column headings, then a dividing rule called here a `\midrule`; after the columns of data the table is finished off with a `\bottomrule`. Most book publishers set the `\toprule` and `\bottomrule` heavier (i.e., thicker, or darker) than the intermediate `\midrule`.

```
\toprule[⟨wd⟩] \bottomrule[⟨wd⟩] \heavyrulewidth
\midrule[⟨wd⟩] \lightrulewidth
```

The rule commands here all take a default width (thickness) which may be reset within the document. For the top and bottom rules this is `\heavyrulewidth` and for midrules it is `\lightrulewidth` (fully described below). In very rare cases where you need to have a special width, you may use the optional argument ⟨*wd*⟩ length argument.

The rule commands go immediately after the closing `\\` of the preceding row (except of course `\toprule`, which comes right after the `\tabular{}` command); in other words, exactly where plain LaTeX allows `\hline` or `\cline`.

```
\cmidrule[⟨wd⟩](⟨trim⟩){⟨a–b⟩} \cmidrulewidth
```

Similar to the normal `\cline` macro, `\cmidrule` draws a rule across the columns (numbered) ⟨*a–b*⟩. Generally, this rule should not come to the full width of the end columns, and this is especially the case when we need to begin a `\cmidrule` straight after the end of another one. The optional 'trimming' commands, which are `(r)`, `(l)` and `(rl)` or `(lr)`, indicate whether the right and/or left ends of the rule should be trimmed. Note the exceptional use of parentheses instead of braces or brackets for this optional argument. For example, the code

```
\cmidrule(r){1-2}
```

is used for Table 12.2(b).

The default width is `\cmidrulewidth` but the optional ⟨*wd*⟩ argument can be used to override this.

An example of the commands in use is given by the code used to produce the 'after' example in Table 12.2(b) above.

Occasionally extra space is required between certain rows of a table; for example, before the last row when it is a total of numbers above. This is simply a matter of inserting `\addlinespace` after the `\\` alignment marker.

```
\addlinespace[⟨wd⟩] \defaultaddspace
```

Think of `\addlinespace` as being a white rule of width ⟨*wd*⟩. The default space is `\defaultaddspace` which gives rather less than a whole line space.

```
\specialrule{⟨wd⟩}{⟨abovespace⟩}{⟨belowspace⟩}
```

You should not use double rules in a table; use rules with different thicknesses instead. The `specialrule` can be used for drawing rules with special thickness and spacing. However, the rules already described should be sufficient without having to resort to `\specialrule`.

```
\morecmidrules
```

Nevertheless, if you insist on having double `\cmidrule`s you will need the extra command `\morecmidrules` to do so properly, because two `\cmidrule`s in a row calls for two rules on the same 'rule row'. Thus in

```
\cmidrule{1-2}\cmidrule{1-2}
```

the second command writes a rule that just overwrites the first one. Use

```
\cmidrule{1-2}\morecmidrules\cmidrule{1-2}
```

which gives you a double rule between columns one and two, separated by `\cmidrulesep`. Finish off a whole row of rules before giving the `\morecmidrules` command. Note that `\morecmidrules` has no effect whatsoever if it does not immediately follow a `\cmidrule` (i.e., it is not a general space-generating command).

The new rule commands are not guaranteed to work with `\hline` or `\cline`. More significantly the rules generated by the new commands are pretty well guaranteed not to connect with verticals generated by `{|}` characters in the preamble. This is a feature — you should not use vertical rules in tables.

Tabular environments

```
\begin{tabular}[⟨pos⟩]{⟨preamble⟩} rows \end{tabular}
\begin{tabular*}{⟨width⟩}[⟨pos⟩]{⟨preamble⟩} rows \end{tabular*}
\begin{tabularx}{⟨width⟩}[⟨pos⟩]{⟨preamble⟩} rows \end{tabularx}
```

A table created using the `tabular` environment comes out as wide as it has to be to accomodate the entries. On the other hand, both the `tabular*` and `tabularx` environments let you specify the overall width of the table via the additional ⟨*width*⟩ atrgument.

The `tabular*` environment makes any necessary adjustment by altering the intercolumn spaces while the `tabularx` environment alters the column widths. Those columns that can be adjusted are noted by using the letter `X` as the column specifier in the ⟨*preamble*⟩. Once the correct column widths have been calculated the `X` columns are converted to `p` columns.

The following code is used for a regular `tabular`.

```
\begin{center}
\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{center}
```

Multicolumn entry!		THREE	FOUR
one	The width of this column is fixed (5.5pc).	three	Column four will act in the same way as column two, with the same width.

The following examples use virtually the same contents, the major differences are the specifications of the environment.

The following `tabularx` is set with

`\begin{tabularx}{250pt}{|c|X|c|X|}`

and the result is:

Multicolumn entry!		THREE	FOUR
one	The width of this column depends on the width of the table.[1]	three	Column four will act in the same way as column two, with the same width.

The following `tabular*` is set with

`\begin{tabular*}{250pt}{|c|p{5.5pc}|c|p{5.5pc}|}`

Notice that there is no `X` column; the total width required for the tabular is less than the specfied width and hence the horizontal lines spill over the right hand end of the apparent tabular width.

Multicolumn entry!		THREE	FOUR	
one	The width of this column is fixed (5.5pc).	three	Column four will act in the same way as column two, with the same width.	

The next `tabular*` table is set with

`\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.

The following `tabularx` is set with

[1]You can use footnotes in a `tabularx`!

```
\begin{tabularx}{300pt}{|c|X|c|X|}
```

the result is:

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.

The main differences between the `tabularx` and `tabular*` environments are:

- `tabularx` modifies the widths of the *columns*, whereas `tabular*` modifies the widths of the inter-column *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.[2]
- `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 ragged right, 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` preamble may include

```
>{\raggedright\arraybackslash}X
```

These preamble 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` preamble argument.

[2]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.

```
\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` preamble specification that you want to correspond to `X`. The argument will be replaced by the calculated width of a column.

The default definition is

```
\newcommand{\tabularxcolumn}[1]{p{#1}}
```

This may be changed, for instance

```
\renewcommand{\tabularxcolumn}[1]{>{\small}m{#1}}
```

so that `X` columns will be typeset as `m` columns using the `\small` font.

Normally all `X` columns in a single table are set to the same width, however it is possible to make `tabularx` set them to different widths. A preamble argument of

```
{>{\hsize=.5\hsize}X>{\hsize=1.5\hsize}X}
```

specifies two columns, the second will be three times as wide as the first. However if you want to play games like this you should follow the following two rules.

1. 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.)
2. Do not use `\multicolumn` entries which cross any `X` column.

As with most rules, these may be broken if you know what you are doing.

It may be that the widths of the 'normal' columns of the table already total more than the requested total width. `tabularx` refuses to set the `X` columns to a negative width, so in this case you get a warning "X Columns too narrow (table too wide)".

The `X` columns will in this case 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
```

`\TX@verb` is the 'poor man's `\verb`' and is based on page 382 of the TeXBook. As explained there, doing verbatim this way means that spaces are not treated correctly, and so a `\verb*` version may well be useless. The mechanism is quite general, and any macro which wants to allow a form of `\verb` to be used within its argument may

```
\let\verb=\TX@verb
```

It must ensure that the real definition is restored afterwards.

This version of `\verb` is subject to the following restictions:

1. Spaces in the argument are not read verbatim, but may be skipped according to TeX's usual rules.
2. Spaces will be added to the output after control words, even if they were not present in the input.
3. Unless the argument is a single space, any trailing space, whether in the original argument, or added as in (2), will be omitted.

4. The argument must not end with `\`, so `\verb|\|` is not allowed, however, because of (3), `\verb|\ |` produces `\`.
5. The argument must be balanced with respect to `{` and `}`. So `\verb|{|` is not allowed.
6. A comment character like `%` will not appear verbatim. It will act as usual, commenting out the rest of the input line!
7. The combinations `?‘` and `!‘` will appear as `¿` and `¡` if the `cmtt` font is being used.

12.6 Free tabulars

All the tabular environments described so far put the table into a box, which LaTeX treats like a large, even though 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 on a caption at the top of each page, repeating the column heads, and so forth.

Continuous tabulars

```
\begin{ctabular}[⟨pos⟩]{⟨preamble⟩} rows \end{ctabular}
\begin{ctabular*}[⟨pos⟩]{⟨width⟩}{⟨preamble⟩} rows \end{ctabular*}
```

The `ctabular` environment is similar to `tabular`, but with a few differences, the main one being that the table will merrily continue across page breaks. The ⟨*preamble*⟩ argument is the same as for the previous `array` and `tabular` environments, but the optional ⟨*pos*⟩ argument controls the horizontal position of the table, not the vertical. The possible argument values are `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

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.

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 table below was set by `\autorows` using:

```
\freetabcaption{Example automatic row ordered table}
\autorows{c}{5}{c}{one, two, three, four, five,
                   six, seven, eight, nine, ten,
                   eleven, twelve, thirteen, fourteen }
```

Table 12.3: Example automatic row ordered table

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

The ⟨*pos*⟩ (`l`, `c`, or `r`) argument controls the horizontal position of the tabular and ⟨*num*⟩ is the number of columns. The ⟨*style*⟩ (`l`, `c`, or `r`) argument specifies the location of the entries in the columns; each column is treated identically. 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.

Each column is normaly the same width which is large enough to accomodate the widest entry in the list. A positive ⟨*width*⟩ (e.g., `[0.8\textwidth]`), defines the overall width of the table, and the column width is calculated by dividing ⟨*width*⟩ by the number of columns. Any negative value for the ⟨*width*⟩ width lets each column be wide enough for the widest entry in that column; the column width is no longer a constant.

The next examples illustrate the effect of the ⟨*width*⟩ argument (the default value is 0pt).

```
\freetabcaption{Changing the width of a row ordered table}
\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, four, five, ... }
\autorows[0.9\textwidth]{c}{5}{c}{one, two, three, four, five, ... }
```

Table 12.4: Changing the width of a row ordered table

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

one	two	three	four	five
six	seven	eight	nine	ten

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

```
\autocols[⟨width⟩]{⟨pos⟩}{⟨num⟩}{⟨style⟩}{⟨entries⟩}
```

The `\autocols` macro lists its ⟨*entries*⟩ in columns, proceeding top to bottom and left to right. The arguments, except for ⟨*width*⟩, are the same as for `\autorows`. The column width is always constant throughout the table and is normally sufficient for the widest entry. A positive ⟨*width*⟩ has the same effect as for `\autorows` but a negative value is ignored.

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 following examples illustrate these points.

```
\freetabcaption{Changing the width of a column ordered table}
\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, four, five,
                   six, seven, eight, nine, ten,
                   eleven, twelve, thirteen, fourteen }
```

Table 12.5: Changing the width of a column ordered table

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

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

12.7 Spacing

Sometimes tabular rows appear vertically challenged.

```
\extrarowheight
```

The length called `\extrarowheight` can be set to modify the normal height of `tabular` (and `array`) rows. If it is a positive length, the value is added to the normal height of every row of the 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, `\setlength{\extrarowheight}{1pt}` was used in the definition of Table 12.1.

Special variations of `\hline`

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     versus
        hline
        commands
        used
tables ┌─────────┐ used.
       │with some│
       │hline    │
       │commands │
       └─────────┘
```

```
Tables
\begin{tabular}[t]{l}
 with no\\ hline \\ commands \\ used
\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     versus
        line
        commands
        used
       ┌─────────┐
tables │with some│ used.
       │line     │
       │commands │
       └─────────┘
```

```
Tables
\begin{tabular}[t]{l}
  with no\\ line \\ commands \\ used
\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.

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.

This 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 preamble from `lll` to `l|l|l` does not affect the document other than adding rules to the table. In contrast, with this class a table that just fitted the `\textwidth` might now produce an overfull box. (But you shouldn't have vertical rules in the first place.)

13 Margin and foot notes

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

13.1 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⟩}
\footnotemark[⟨num⟩] \footnotetext[⟨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.

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}}
```

```
\footref{⟨label⟩}
```

On occasions it may be desireable to make more than one reference to the text of a footnote. This can be done by putting a `\label` in the footnote and then using `\footref` to refer to the label; this prints the footnote label. 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 footnote[3] on page 13 ...

The parameters provided by the standard classes for controlling footnotes are illustrated in Figure 13.1.

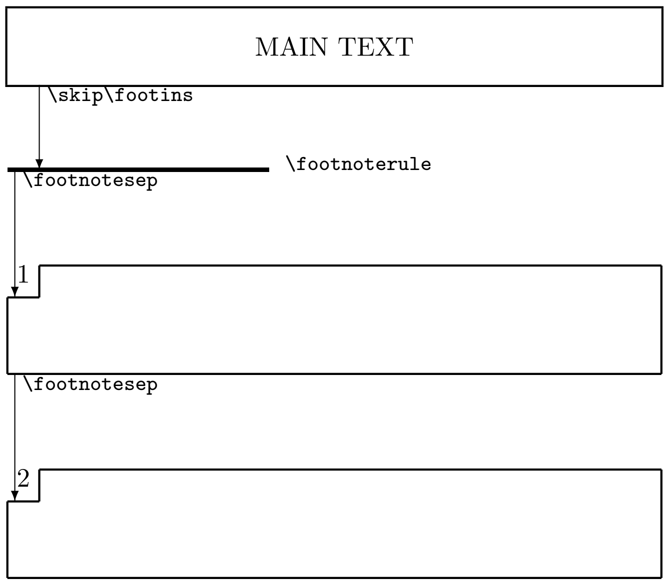


Figure 13.1 — Footnote layout parameters for the standard classes

```
\footnotesep
\skip\footins
```

The length `\footnotesep` controls the vertical spacing between footnotes (and thanks notes), and is initialised by the class to give no extra spacing for a `\footnotesize` font. You can change the spacing by

```
\addtolength{\footnotesep}{...}
```

The length `\skip\footins` defines the vertical spacing between the bottom of the main text and the top of the first line of the first footnote. Likewise this can be changed by

```
\addtolength{\skip\footins}{...}
```

The total vertical distance between the bottom of the main text and the baseline of the first line of the first footnote is `\footnotesep + \skip\footins`.

```
\footnoterule
```

The `\footnoterule` macro is defined in the LaTeX kernel and redefined in the standard classes. An '`@`less' definition, which is slightly less efficient than that in the classes, is:

```
\renewcommand{\footnoterule}{%
  \kern -3pt                   % call this kerna
  \hrule height 0.4pt width 0.4\columnwidth
  \kern 2.6pt                  % call this kernb
}
```

This produces a horizontal rule from the left margin to 40% of the `\columnwidth` with thickness 0.4pt. The rule must not take up any vertical space, which means that `kerna + kernb` must equal the thickness of the rule. The rule is located a distance `\skip\footins + kerna` below the bottom of the main text. So, to move the rule upwards, decrease the value of `kerna` and increase that of `kernb`, and the reverse to move the rule downwards.

This class provides some additional parameters but for a wider variety of footnote styles you may like to use the footmisc package [Fai00]. The additional class parameters have been chosen so as not to clash with the footmisc package. Similar concepts apply to the `\thanks` command described in §7.2 starting on page 71.

The `\footnote` macro essentially does three things:

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

```
\@makefnmark
```

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

```
\def\@makefnmark{\hbox{\textsuperscript{\@thefnmark}}}
```

To get, say, the marker to be typeset on the baseline in the normal font and enclosed in brackets —

```
\renewcommand*{\@makefnmark}{ [\@thefnmark]}
```

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

The class macro for typesetting the marker at the foot of the page is `\footfootmark`. The mark is set in a box of width `\footmarkwidth`. If this is negative, the mark is outdented into the margin, if zero the mark is flush left, and when positive the mark is indented. The appearance of the mark is contolled by `\footmarkstyle`. The default specification is

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

where the `#1` indicates the position of `\@thefnmark` in the style. The default results in the mark being set as a superscript. For example, to have the marker set on the baseline and followed by a right parenthesis, do

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

```
\footmarksep \footparindent
```

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

```
\foottextfont
```

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

Altogether, the class specifies

```
\footmarkstyle{\textsuperscript{#1}}
\setlength{\footmarkwidth}{1.8em}
\setlength{\footmarksep}{-1.8em}
\setlength{\footparindent}{1em}
\newcommand{\foottextfont}{\footnotesize}
```

to replicate the standard footnote layout.

You might like to try the following combinations of `\footmarkwidth` and `\footmarksep` to see if you prefer the layout produced by one of them to the standard layout (which is produced by the first pairing in the list below):

```
\setlength{\footmarkwidth}{1.8em}  \setlength{\footmarksep}{-1.8em}
\setlength{\footmarkwidth}{1.8em}  \setlength{\footmarksep}{0em}
\setlength{\footmarkwidth}{0em}    \setlength{\footmarksep}{0em}
\setlength{\footmarkwidth}{-1.8em} \setlength{\footmarksep}{1.8em}
\setlength{\footmarkwidth}{0em} \setlength{\footmarksep}{1.8em} \footmarkstyle{#1)\hfill}
```

```
\makefootmarkhook
```

Just before a footnote is typeset the macro `\makefootmarkhook` is called, which by default does nothing, but you can renew it to do something. For example:

```
\renewcommand{\makefootmarkhook}{\raggedright}
```

will cause footnotes to be typeset raggedright.

Any footnotes after this point will be set according to:

```
\setlength{\footmarkwidth}{-1.0em}
\setlength{\footmarksep}{-\footmarkwidth}
\footmarkstyle{#1}
```

```
\multfootsep
```

In the standard classes if two or more footnotes are applied sequentially[1,2] then the markers in the text are just run together. The class, like the footmisc [Fai00] and ledmac [Wil03] packages, inserts a separator between the marks. In the class the macro `\multfootsep` is used as the separator. Its default definition is:

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

```
\feetabovefloat
\feetbelowfloat
```

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

1 One footnote
2 Immediately followed by another

```
\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
...
```

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

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

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

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

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

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

```
\renewcommand{\foottextfont}{\itshape\footnotesize}
```

This can then be used like:

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

In a minipage a different counter (`mpfootn...`) is used for a footnote than in the main text (`footn...`). By default its value is a letter. To change it to, say, a numeral for the ‘P’ series, do:

```
\renewcommand{\thempfootnoteP}{\arabic{mpfootnoteP}}
```

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.

```
\footfudgefiddle
```

For paragraphed footnotes TeX has to estimate the amount of space they will take. Unfortunately this *is* an estimate and if it is an underestimate then the footnotes will flow too low on the page, for example below the page number. Increasing `\footfudgefiddle` from its default vaue of 64, causes TeX to allot more space. For instance

```
\renewcommand{\footfufgefiddle}{70}
```

will at least go some way to curing the problem. You will probably have to do some experimentation to get a good value.

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

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

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

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

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

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

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

13.2 Verbatim footnotes

The macro `\verbfootnote` can typeset a footnote that includes some verbatim text.

```
\verbfootnote[⟨num⟩]{⟨text⟩}
```

The next two paragraph are typeset by this code:

```
    Below, footnote\footref{fn1} is a \cmd{\footnote} while
footnote~\ref{fn2} is a \cmd{\verbfootnote}.

    The \cmd{\verbfootnote} command should
appear\footnote{There may be some problems if color is
                used.\label{fn1}}
to give identical results as the normal \cmd{\footnote},
but it can include some verbatim
text\verbfootnote{The \verb?\footnote? macro, like all others
                  except for \verb?\verbfootnote?, can not
                  contain verbatim text in its argument.\label{fn2}}
in the \meta{text} argument.
```

Below, footnote[3] is a `\footnote` while footnote 4 is a `\verbfootnote`.

The `\verbfootnote` command should appear[3] to give identical results as the normal `\footnote`, but it can include some verbatim text[4] in the ⟨*text*⟩ argument.

13.3 Sidebars

It appears that the `\sidebar` command, which was originally noted as being experimental, has been used successfully for a variety of tasks.

```
\sidebar{⟨text⟩}
\sidebarfont
```

The `\sidebar` command is like the `\marginpar` command in that it typesets its ⟨*text*⟩ argument, using the font specified by `\sidebarfont`, in the margin. 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.

The default definition of `\sidebarfont` is

```
\newcommand{\sidebarfont}{\normalfont}
```

```
\sidebarform
```

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

3 There may be some problems if color is used.

4 The `\footnote` macro, like all others except for `\verbfootnote`, can not contain verbatim text in its argument.

```
\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 are likely to 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 (e.g., around a `\section`). Also, the contents of sidebars may not be typeset if they are too near to the end of the document.

```
\sidebarwidth
\sidebarhsep
\sidebarvsep
```

The ⟨*text*⟩ of a `\sidebar` is typeset in a column of width `\sidebarwidth` and there will be a horizontal gap of length `\sidebarhsep` between the typeblock and the ⟨*text*⟩. The length `\sidebarvsep` is the vertical gap between sidebars that fall on the same page.

```
\setsidebarheight{⟨length⟩}
```

The macro `\setsidebarheight` can be used to specify the height of a sidebar. The ⟨*length*⟩ argument should be equivalent to an integral number of lines. For example:

```
\setsidebarheight{15\onelineskip}
```

Currently the alignment of the text in a sidebar with the main text is not particularly good and you may want to do some experimentation (possibly through a combination of `sidebarvsep` and `\setsidebarheight`) to make it better.[5]

13.4 Side notes

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

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

The `\sidepar` macro is similar to `\marginpar` except that it produces side notes that do not float — they may overlap. The same spacing is used for both `\marginpar` and `\sidepar`, namely the lengths `\marginparsep` and `\marginparwidth`. The length `\sideparvshift` can be used to make vertical adjustments to the position of `\sidepar` notes. By default this is set to a value of -2.08ex which seems to be the magical length that aligns the top of the note with the text line.

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

5 If you do improve the alignment, please let me know how you did it.

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

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

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

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
         \strictpagechecktrue%
         \checkoddpage%
         \setlength{\draftnotewidth}{\foremargin}%
         \addtolength{\draftnotewidth}{\trimedge}%
         \addtolength{\draftnotewidth}{-3\marginparsep}%
         \ifoddpage
           \setlength{\marginparwidth}{\draftnotewidth}%
           \marginpar{\flushleft\textbf{\textit{\HUGE !\ }}\small #1}%
         \else
           \settowidth{\draftnotesignwidth}{\textbf{\textit{\HUGE\ !}}}%
```

```
              \addtolength{\draftnotewidth}{-\draftnotesignwidth}%
              \marginpar{\raggedleft\makebox[0pt][r]{%% hack around
                  \parbox[t]{\draftnotewidth}{%%%%%%%%% funny behaviour
                    \raggedleft\small\hspace{0pt}#1%
                  }}\textbf{\textit{\HUGE\ !}}%
              }%
            \fi
          }\@esphack}
      \else
        \newcommand{\draftnote}[1]{\@bsphack\@esphack}
      \fi
   \makeatother
```

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.

CHAPTER 14

Signposts and decoration

My soul, seek not the life of immortals; but enjoy to the full the resources that are within your reach

Pythian Odes
Pindar

This chapter is typeset using the *companion* chapterstyle and the *companion* pagestyle.

14.1 Introduction

By now we have covered most aspects of typesetting. As far as the class is concerned this chapter describes the remaining major element, namely headers and footers, and the slightly more fun task of typesetting epigraphs.

14.2 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 there are a total of 12 elements that have to be specified for a pagestyle. You may find, though, that one of the built in pagestyles meets your needs so you don't have to worry about all these specifications.

```
\pagestyle{⟨style⟩}
\thispagestyle{⟨style⟩}
```

`\pagestyle` sets the current pagestyle to ⟨*style*⟩. On a particular page `\thispagestyle` can be used to override the cuurent pagestyle for the one page.

Some of the class' commands automatically call `\thispagestyle`:

- `\part` calls `\thispagestyle{part}`;
- `\chapter` calls `\thispagestyle{chapter}`;
- if `\cleardoublepage` will result in an empty verso page it calls `\thispagestyle{cleared}` for the empty page.

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 and the header contains the the folio at the outer side of the page, and either the chapter or section title. Chapter 5 uses this style.
myheadings	This is like the *headings* style except you have to specify the titles, if any, to go into the header.
ruled	The footer contains the folio at the outside. The header contains either the chapter or section titles, and a line is drawn underneath the header. This style is used in Chapter 6.
Ruled	This is like the *ruled* style except that the headers and footers extend into the foredge margin. Chapter 15 uses this style.
companion	This is like the pagestyle in the *Companion* series (e.g., see [GMS94]). For an example, see Chapter 14.
part	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.
title	This is the same as the *plain* pagestyle.
titlingpage	This is the same as the *empty* pagestyle.

For the *headings* pagestyle above, you have to define your own titles to go into the header. Each sectioning command, say `\sec`, calls a macro called `\secmark`. The pagestyles usually define this command so that it picks up the title, and perhaps the number, of the `\sec`.

```
\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*⟩.

```
\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.

So, 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.

14.3 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.

```
\makepagestyle{⟨style⟩}
\aliaspagestyle{⟨alias⟩}{⟨original⟩}
\copypagestyle{⟨new⟩}{⟨original⟩}
```

The command `\makepagestyle` specifies a pagestyle ⟨*style*⟩ which is initially equivalent to the *empty* pagestyle. The `\aliaspagestyle` macro defines the ⟨*alias*⟩ pagestyle to be the same as the ⟨*original*⟩ pagestyle. As an example of the latter, the class includes the code

```
\aliaspagestyle{part}{plain}
\aliaspagestyle{chapter}{plain}
\aliaspagestyle{cleared}{empty}
```

The macro `\copypagestyle` creates a ⟨*new*⟩ pagestyle with the definitions being copies of those for the ⟨*original*⟩ pagestyle.

```
\makeevenhead{⟨style⟩}{⟨left⟩}{⟨center⟩}{⟨right⟩}
\makeoddhead{⟨style⟩}{⟨left⟩}{⟨center⟩}{⟨right⟩}
\makeevenfoot{⟨style⟩}{⟨left⟩}{⟨center⟩}{⟨right⟩}
\makeoddfoot{⟨style⟩}{⟨left⟩}{⟨center⟩}{⟨right⟩}
```

The macro `\makeevenhead` defines the ⟨*left*⟩, ⟨*center*⟩, and ⟨*right*⟩ portions of the ⟨*style*⟩ pagestyle header for even numbered (verso) pages. Similarly `\makeoddhead`, `\makeevenfoot`, and `\makeoddfoot` define the ⟨*left*⟩, ⟨*center*⟩ and ⟨*right*⟩ portions of the ⟨*style*⟩ header for odd numbered (recto) pages, and the footers for verso and recto pages. These commmands 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` sets 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 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 the 14pt or 17pt class options.

```
\footruleheight
\footruleskip
```

The macro `\footruleheight` is the height of a normal rule above a footer (default zero). `\footruleskip` is a distance sufficient to ensure that a foot rule will be placed between the bottom of the typeblock and the footer. Despite appearing to be lengths, if you really need to change the values use `\renewcommand`, not `\setlength`.

```
\makeheadposition{⟨style⟩}
{⟨eheadpos⟩}{⟨oheadpos⟩}{⟨efootpos⟩}{⟨ofootpos⟩}
```

The `\makeheadposition` macro specifies the horizontal positioning of the even and odd headers and footers, respectively, for the ⟨*style*⟩ pagestyle. Each of the ⟨*...pos*⟩ arguments may be `flushleft`, `center`, or `flushright`, with the obvious meanings. An empty, or unrecognised, argument is equivalent to `center`. This macro is really only of use if the header/footer width is not the same as the `\textwidth`.

```
\makepsmarks{⟨style⟩}{⟨code⟩}
```

The last thing that the `\pagestyle{`⟨*style*⟩`}` does is call the ⟨*code*⟩ argument of the `\makepsmarks` 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.

Example pagestyles

This first example demonstrates most of the page styling commands. In the *LaTeX Companion* series of books [GMS94, GRM97, GR99] the header is wider than the typeblock, sticking out into the foredge 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 spine 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 foredge 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, and this is where the `\makepsmarks` macro comes into play.

```
\makepsmarks{companion}{%
  \let\@mkboth\markboth
  \def\chaptermark##1{\markboth{##1}{##1}}% % left & right marks
  \def\sectionmark##1{\markright{%         % right mark
    \ifnum \c@secnumdepth>\z@
      \thesection. \%                      % section number
    \fi
    ##1}}
}
```

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 definition is actually part of the class. If you want to alter it, or use it as the basis for a style of your own, remember the caveats about macros with `@` in their names.

Special 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 manual `\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 a little fragile on twocolumn pages, but the standard fixltx2e package [MC00] 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 foredge margin.

I used Pehong Chen's *MakeIndex* program [CH88] for converting the raw index data in the `.idx` file into the form expected by LaTeX in the `.ind` file. *MakeIndex* can be configured via an `.ist` file, and the `memman.ist` file that I created for this manual includes the mysterious lines

```
% output main entry <entry> as: \item \idxmark{<entry>},
item_0  "\n\\item \\idxmark{"
delim_0 "},"
```

Look up either Chen's paper [CH88] or Chapter 12 in [GMS94] for an explanation of this cryptic code.

`\ifonlyfloats{⟨yes⟩}{⟨no⟩}`

There are occasions when it is desireable 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 having to retype 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}
```

14.4 Epigraphs

The whole is more than the sum of the parts.

Metaphysica
Aristotle

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 eipgraphs 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.

The `epigraph` command

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*⟩.

The `epigraphs` environment

```
\begin{epigraphs}
\qitem{⟨text⟩}{⟨source⟩}
...
\end{epigraphs}
```

The `epigraphs` environment typesets a list of epigraphs, and by default places them at the right hand side of the typeblock. Each epigraph in an `epigraphs` environment is specified by a `\qitem` (analagous to the `\item` command in ordinary list environments). By default, the ⟨*source*⟩ is typeset at the bottom right of the ⟨*text*⟩.

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
\epigraphtextposition{⟨flush⟩}
```

The epigraphs are typeset in a minipage of width `\epigraphwidth`. The default value for this can be changed using the `\setlength` command. Typically, epigraphs are typset in a measure much less than the width of the typeblock. In order to avoid bad line breaks, the ⟨*text*⟩ is normally typeset raggedright.

The ⟨*flush*⟩ argument to the `\epigraphtextposition` declaration controls the ⟨*text*⟩ typesetting style. By default this is `flushleft` (which produces raggedright text). The sensible values are `center` for centered text, `flushright` for raggedleft text, and `flushleftright` for normal justified text.

If by any chance you want the ⟨*text*⟩ to be typeset in some other layout style, the easiest way to do this is by defining a new environment which sets the paragraphing parameters to your desired values. For example, as the ⟨*text*⟩ is typeset in a minipage, there is no paragraph indentation. If you want the paragraphs to be indented and justified then define a new environment like:

```
\newenvironment{myparastyle}{\setlength{\parindent}{1em}}{}
```

and use it as:

```
\epigraphtextposition{myparastyle}
```

```
\epigraphposition{⟨flush⟩}
```

As noted, the default position of epigraphs is at the right hand side of the typeblock. The positioning is controlled by the ⟨*flush*⟩ argument to the `\epigraphposition` declaration. The default value is `flushright`. This can be changed to `flushleft` for positioning at the left hand side or to `center` for positioning at the center of the typeblock.

```
\epigraphsourceposition{⟨flush⟩}
```

The ⟨*flush*⟩ argument to the `\epigraphsourceposition` declaration controls the position of the ⟨*source*⟩. The default value is `flushright`. It can be changed to `flushleft`, `center` or `flushleftright`.

For example, to have epigraphs centered with the ⟨*source*⟩ at the left, add the following to your document.

```
 \epigraphposition{center}
 \epigraphsourceposition{flushleft}
```

```
\epigraphfontsize{⟨fontsize⟩}
```

Epigraphs are often typeset in a smaller font than the main text. The ⟨*fontsize*⟩ argument to the `\epigraphfontsize` declaration sets the font size to be used. If you don't like the default value (`\small`), you can easily change it to, say `\footnotesize` by:

```
\epigraphfontsize{\footnotesize}
```

```
\epigraphrule
```

By default, a rule is drawn between the ⟨*text*⟩ and ⟨*source*⟩, with the rule thickness being given by the value of `\epigraphrule`. The value can be changed by using `\setlength`. A value of `0pt` will eliminate the rule. Personally, I dislike the rule in the list environments.

```
\beforeepigraphskip
\afterepigraphskip
```

The two `...skip` commands specify the amount of vertical space inserted before and after typeset epigraphs. Again, these can be changed by `\setlength`. It is desireable that the sum of their values should be an integer multiple of the `\baselineskip`.

Note that you can use normal LaTeX commands in the ⟨*text*⟩ and ⟨*source*⟩ arguments. You may wish to use different fonts for the ⟨*text*⟩ (say roman) and the ⟨*source*⟩ (say italic).

The epigraph at the start of this section was specified as:

```
\epigraph{Example is the school of mankind,
          and they will learn at no other.}
 {\textit{Letters on a Regicide Peace}\\ \textsc{Edmund Burke}}
```

Epigraphs before chapter headings

The `\epigraph` command and the `epigraphs` environment typeset an epigraph at the point in the text where they are placed. The first thing that a `\chapter` command does is to start off a new page, so another mechanism is provided for placing an epigraph just before a chapter heading.

`\epigraphhead[`⟨*distance*⟩`]{`⟨*text*⟩`}`

The `\epigraphhead` macro stores ⟨*text*⟩ for printing at ⟨*distance*⟩ below the header on a page. ⟨*text*⟩ can be ordinary text or, more likely, can be either an `\epigraph` command or an `epigraphs` environment. By default, the epigraph will be typeset at the righthand margin. If the command is immediately preceded by a `\chapter` or `\chapter*` command, the epigraph is typeset on the chapter title page.

The default value for the optional ⟨*distance*⟩ argument is set so that an `\epigraph` consisting of a single line of quotation and a single line denoting the source is aligned with the bottom of the 'Chapter X' line produced by the `\chapter` command. In other cases you will have to experiment with the ⟨*distance*⟩ value. The value for ⟨*distance*⟩ can be either a integer or a real number. The units are in terms of the current value for `\unitlength`. A typical value for ⟨*distance*⟩ for a single line quotation and source for a `\chapter*` might be about 70 (points). A positive value of ⟨*distance*⟩ places the epigraph below the page heading and a negative value will raise it above the page heading.

Here's some example code:

```
\chapter*{Celestial navigation}
\epigraphhead[70]{\epigraph{Star crossed lovers.}{\textit{The Bard}}}
```

The ⟨*text*⟩ argument is put into a minipage of width `\epigraphwidth`. If you use something other than `\epigraph` or `epigraphs` for the ⟨*text*⟩ argument, you may have to so some positioning of the text yourself so that it is properly located in the minipage. For example

```
\chapter{Short}
\renewcommand{\epigraphflush}{center}
\epigraphhead{\centerline{Short quote}}
```

The `\epigraphhead` command changes the page style for the page on which it is specified, so there should be no text between the `\chapter` and the `\epigraphhead` commands. The page style is identical to the *plain* page style except for the inclusion of the epigraph. If you want a more fancy style for epigraphed chapters you will have to do some work yourself.

```
\epigraphforheader[⟨distance⟩]{⟨text⟩}
\epigraphpicture
```

The `\epigraphforheader` macro takes the same arguments as `\epigraphhead` but puts ⟨*text*⟩ into a zero-sized picture at the coordinate position `(0,-<distance>)`; the macro `\epigraphpicture` holds the resulting picture. This can then be used as part of a chapter pagestyle, as in

```
\makepagestyle{mychapterpagestyle}
...
\makeoddhead{mychapterpagestyle}{}{}{\epigraphpicture}
```

```
\dropchapter{⟨length⟩}
\undodrop
```

If a long epigraph is placed before a chapter title it is possible that the bottom of the epigraph may interfere with the chapter title. The command `\dropchapter` will lower any subsequent chapter titles by ⟨*length*⟩; a negative ⟨*length*⟩ will raise the titles. The command `\undodrop` restores subsequent chapter titles to their default positions. For example:

```
\dropchapter{2in}
\chapter{Title}
\epigraphhead{long epigraph}
\undodrop
```

```
\cleartoevenpage[⟨text⟩]
```

On occasions it may be desirable to put something (e.g., an epigraph, a map, a picture) on the page facing the start of a chapter, where the something belongs to the chapter that is about to start rather than the chapter that has just ended. In order to do this in a document that is going to be printed doublesided, the chapter must start on an odd numbered page and the pre-chapter material put on the immediately preceding even numbered page. The `\cleartoevenpage` command is like `\cleardoublepage` except that the page following the command will be an even numbered page, and the command takes an optional argument which is applied to the skipped page (if any).

Here is an example:

```
... end previous chapter.
\cleartoevenpage
\begin{center}
\begin{picture}... \end{picture}
\end{center}
\chapter{Next chapter}
```

If the style is such that chapter headings are put at the top of the pages, then it would be advisable to include `\thispagestyle{empty}` (or `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}] % skipped page, if any, 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 & initialises figure numbers
\addtocounter{figure}{1}   % to account for pre-chapter figure
```

Epigraphs on part pages

The epigraph package as it stands cannot put an epigraph on the same page as a `\part` or `\part*` title page in a book or report class. This is because the `\part` command internally does some page flipping before and after the title page. However, it is easy enough to put epigraphs on part pages.

- Create a file called, say, `epipart.sty` which looks like this:
  ```
  % epipart.sty
  \let\@epipart\@endpart
  \renewcommand{\@endpart}{\thispagestyle{epigraph}\@epipart}
  \endinput
  ```

- Start your document like:
  ```
  \documentclass[...]{...}
  \usepackage{epigraph}
  \usepackage{epipart}
  ```

- Immediately *before* each `\part` command put an `\epigraphhead` command. For example:
  ```
  \epigraphhead[300]{Epigraph text}
  \part{Part title}
  ```

  The value of the optional argument may need changing to vertically adjust the position of the epigraph. If there is any `\part` that does not have an epigraph then an empty `\epigraphhead` command (i.e., `\epigraphhead{}`) must be placed immediately before the `\part` command.

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.

Fifteen

Typesetting verse

This chapter uses the *demo* chapterstyle and the *Ruled* pagestyle.

15.1 Introduction

The typesetting of a poem should be really be dependent on the particular poem. Individual problems do not usually admit of a general solution, so this manual and code should be used more as a guide towards some solutions rather than providing a ready made solution for any particular piece of verse.

The doggerel used as illustrative material has been taken from [Wil??].

Note that for the examples in this section I have made no attempt to do other than use the minimal (La)TeX capabilities; in particular I have made no attempt to do any special page breaking so some stanzas may cross onto the next page — most undesireable for publication.

The standard LaTeX classes provide the `verse` environment which is defined as a particular kind of list. Within the environment you use `\\` to end a line, and a blank line will end a stanza. For example, here is 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) it looks like:

I used to love my garden
But now my love is dead
For I found a bachelor's button
In black-eyed Susan's bed.

Typesetting it within the `verse` environment produces:

> I used to love my garden
> But now my love is dead
> For I found a bachelor's button
> In black-eyed Susan's bed.

The stanza could also be typeset within the `alltt` environment, defined in the standard alltt package [Bra97], using a normal font and no `\\` line endings.

```
\begin{alltt}\normalfont
I used to love my garden
But now my love is dead
For I found a bachelor's button
In black-eyed Susan's bed.
\end{alltt}
```

which produces:

I used to love my garden
But now my love is dead
For I found a bachelor's button
In black-eyed Susan's bed.

The `alltt` environment is like the `verbatim` environment except that you can use LaTeX macros inside it. In the `verse` environment long lines will be wrapped and indented but in the `alltt` environment there is no indentation.

Some stanzas have certain lines indented, often alternate ones. To typeset stanzas like this you have to add your own spacing. For instance:

```
\begin{verse}
There was an old party of Lyme \\
Who married three wives at one time. \\
\hspace{2em} When asked: 'Why the third?' \\
\hspace{2em} He replied: 'One's absurd, \\
And bigamy, sir, is a crime.'
\end{verse}
```

is typeset as:

> There was an old party of Lyme
> Who married three wives at one time.
>   When asked: 'Why the third?'
>   He replied: 'One's absurd,
> And bigamy, sir, is a crime.'

Using the `alltt` environment you can put in the spacing via ordinary spaces. That is, this:

```
\begin{alltt}\normalfont
There was an old party of Lyme
Who married three wives at one time.
      When asked: 'Why the third?'
      He replied: 'One's absurd,
And bigamy, sir, is a crime.'
\end{alltt}
```

is typeset as

There was an old party of Lyme
Who married three wives at one time.
    When asked: 'Why the third?'
    He replied: 'One's absurd,
And bigamy, sir, is a crime.'

More exotically you could use the TeX `\parshape` command[1]:

```
\parshape = 5 0pt \linewidth 0pt \linewidth
              2em \linewidth 2em \linewidth 0pt \linewidth
\noindent There was an old party of Lyme \\
Who married three wives at one time. \\
When asked: 'Why the third?' \\
He replied: 'One's absurd, \\
And bigamy, sir, is a crime.' \par
```

which will be typeset as:

There was an old party of Lyme
Who married three wives at one time.
    When asked: 'Why the third?'
    He replied: 'One's absurd,
And bigamy, sir, is a crime.'

This is about as much assistance as standard (La)TeX provides, except to note that in the `verse` environment the `\\*` version of `\\` will prevent a following page break. You can also make judicious use of the `\needspace` macro to keep things together.

Some books of poetry, and especially anthologies, have two or more indexes, one, say for the poem titles and another for the first lines. The index [Jon95] and multind [Lon91] packages provide support for multiple indexes in one document.

15.2 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.

1 See the *TeXbook* for how to use this.

```
\begin{verse}[⟨length⟩] ... \end{verse}
\versewidth
\stanzaskip
```

The `verse` environment provided by the class is an extension of the usual LaTeX environment. The environment takes one optional parameter, which is a length; for example `\begin{verse}[4em]`. You may have noticed that the earlier verse examples are all near the left margin, whereas verses usually look better if they are typeset about the center of the page. The length parameter, if given, should be about the length of an average line, and then the entire contents will be typeset with the mid point of the length centered horizontally on the page.

The length `\versewidth` is provided as a convenience. It may be used, for example, to calculate the length of a line of text for use as the optional argument to the `verse` environment:

```
\settowidth{\versewidth}{This is the average line,}
\begin{verse}[\versewidth]
```

The vertical space between stanzas is the length `\stanzaskip`. It can be changed by the usual methods.

```
\\ \\* \\> \\!
```

Each line in the `verse` environment, except possibly for the last line in a stanza, must be ended with either `\\` or `\\*`. The starred version prevents a page break before the following line. The `\\>` command causes a break in the line (see the description of the `\verselinebreak` macro).

The `\\!` macro may be used at the end of a line to signal the end of the stanza. This would normally be followed by a blank line before the next stanza.

Each of the `\\...` macros take an optional length argument. In the case of the `\\`, `\\*` and `\\!` macros it introduces the specified amount of vertical space. For `\\>` it specifies a horizontal space.

```
\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`.

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

```
\vinphantom{⟨text⟩}
```

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

> ...
> Come away with me.

Impossible!

. . .

The above fragment from a poem was typeset by:

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

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

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

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

When a verse line is too long to fit within the typeblock it is wrapped onto the next line with a space, given by the value of the length `\vindent`. The initial value of `\vindent` is twice `\vgap`.

```
\verselinebreak[⟨length⟩]
```

Using the command `\verselinebreak` will cause later text in the line of the verse to be typeset indented on the following line. If the optional length argument is given then its value is added to the normal indentation. The broken line will count as a single line as far as the `altverse`, `patverse`, and `patverse*` environments are concerned. The `\\>` macro is shorthand for `\verselinebreak`.

```
\begin{altverse} ... \end{altverse}
```

Within the `verse` environment stanzas are separated by a blank line in the input. 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`. The first line is not indented, irrespective of the value of d_1 and if the number of lines is greater than the pattern length, the trailing lines are not indented.

The `patverse*` environment is similar to `patverse` except that the indentation pattern will keep repeating until the end of the environment.

The indentation pattern for a `patverse` environment is specified via the `\indentpattern` command, where ⟨*digits*⟩ is a string of digits (e.g., `3213245281`).

```
\linenumberfrequency{⟨nth⟩}
\thepoemline
\linenumberfont{⟨font-decl⟩}
\vrightskip
```

The lines in a poem may be numbered. The `\linenumberfrequency` declaration specifies that every ⟨*nth*⟩ line is to be numbered. If ⟨*nth*⟩ is less than 1 then line numbering is switched off, if ⟨*nth*⟩ is 1 then every line will be numbered, and if ⟨*nth*⟩ is, say 5, every fifth line will be numbered. The default is `\linenumberfrequency{0}`.

The counter for the lines is `poemline`, so the typeset form of the line number is specified via `\thepoemline` which defaults to arabic numbers. The number is positioned in the right hand margin at a distance `\vrightskip` (default 1em) from the edge of the typeblock. The font used for the line numbers is specified by `\linenumberfont`. The default definition is:
`\linenumberfont{\small\rmfamily}`
to produce small numbers in the roman font.

```
\flagverse{⟨text⟩}
\vleftskip
```

The `\flagverse` macro can be used at the start of a verse line, and it typesets its ⟨*text*⟩ argument at a distance `\vleftskip` (default 3em) to the left of the verse line. This could be used, for example, to number stanzas.

```
\poemtitle[⟨short⟩]{⟨long⟩}
\poemtitle*{⟨long⟩}
\poemtitlefont{⟨font⟩}
```

`\poemtitle` typesets the title of a poem and makes an entry into the ToC. The starred version, `\poemtitle*`, makes no ToC entry. The `\poemtitlefont` macro specifies the font and positioning of the poem title. Its initial definition is:
`\newcommand{\poemtitlefont}{\normalfont\bfseries\large\centering}`
to give a large bold centered title. This can of course be renewed if you want something else.

```
\beforepoemtitleskip
\afterpoemtitleskip
```

These two lengths are the vertical space before and after the `\poemtitle` title text. They are initially defined to give the same spacing as for a `\section` title. They can be changed by `\setlength` or `\addtolength` for different spacings.

```
\poemtitlemark{⟨title⟩}
```

The `\poemtitle` macro, but not `\poemtitle*`, calls the `\poemtitlemark{⟨title⟩}` macro, which is defined to do nothing. This would probably be changed by a pagestyle definition (like `\sectionmark` or `\chaptermark` may be modified).

```
\poemtoc{⟨sec⟩}
```

The kind of entry made in the ToC by the `\poemtitle` command 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 ....

Examples

Here are some sample verses using the class facilities.

First our old Limerick friend, but titled and centered:

```
\renewcommand{\poemtoc}{subsection}
\settocdepth{subsection}
\poemtitle{A Limerick}
\settowidth{\versewidth}{There was an old party of Lyme}
\begin{verse}[\versewidth]
There was an old party of Lyme \\
Who married three wives at one time. \\
\vin When asked: ‘Why the third?’ \\
\vin He replied: ‘One’s absurd, \\
And bigamy, sir, is a crime.’
\end{verse}
```

which gets typeset as below. The `\poemtoc` is redefined to `subsection` so that the `\poemtitle` titles are entered into the ToC as unnumbered subsections. However, the ToC normally only lists sections and above, so I also used `\settocdepth` to change this to subsections and above[2].

A Limerick

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.’

Next is the Garden verse within the `altverse` environment. It is titled and centered.

```
\settowidth{\versewidth}{But now my love is dead}
\poemtitle{Loves Lost}
\begin{verse}[\versewidth]
```

2 It is changed back at the end of this chapter.

```
\begin{altverse}
\garden
\end{altverse}
\end{verse}
```

Note how the alternates lines are automatically indented in the typeset result below.

Loves 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. 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 you only need specify one `\indentpattern` for all of them.

```
\settowidth{\versewidth}{There was a young lady of Ryde}
\indentpattern{00110}
\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}
```

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 that I have known since childhood. The ‘forty-niner’ in line 3 refers to the 1849 gold rush.

```
\settowidth{\versewidth}{Oh my darling, Clementine}
\poemtitle{Clementine}
\begin{verse}[\versewidth]
\linenumberfrequency{3}
\flagverse{1.} In a cavern, in a canyon, \\
Excavating for a mine, \\
Lived a miner, forty-niner, \label{vs:1} \\
And his daughter, Clementine. \\!
\flagverse{\textsc{chorus}} Oh my darling, Oh my darling, \\
Oh my darling, Clementine. \\
Thou art lost and gone for ever. \\
Dreadful sorry, Clementine. \\!
\linenumberfrequency{0}
\end{verse}
```

Clementine

1. In a cavern, in a canyon,
Excavating for a mine,
Lived a miner, forty-niner,
And his daughter, Clementine.

CHORUS Oh my darling, Oh my darling,
Oh my darling, Clementine.
Thou art lost and gone for ever.
Dreadful sorry, Clementine.

The last example is a much more ambitious use of `\indentpattern`. In this case it is defined as:
`\indentpattern{0135554322112346898779775545653222345544456688778899}`
and the result is shown on the next page.

Mouse's Tale

Fury said to
a mouse, That
he met
in the
house,
'Let us
both go
to law:
I will
prosecute
you. —
Come, I'll
take no
denial;
We must
have a
trial:
For
really
this
morning
I've
nothing
to do.'
Said the
mouse to
the cur,
Such a
trial,
dear sir,
With no
jury or
judge,
would be
wasting
our breath.'
'I'll be
judge,
I'll be
jury.'
Said
cunning
old Fury;
'I'll try
the whole
cause
and
condemn
you
to
death.'

Lewis Carrol, *Alice's Adventures in Wonderland*, 1865

Sixteen

Verbatims, boxes, and files

16.1 Introduction

This chapter describes some new facilities for handling verbatim text; new boxes to go around things, including verbatims; and a more friendly interface for writing to and reading from files and in some cases being able to read files verbatim, while putting them into a box with line numbers.

16.2 Verbatims

I have included in the class the code for the shortvrb package [Mit95] and an extended version of the verbatim package [SRR99]. I have also borrowed from the moreverb package [Fai98].

When processing text verbatim LaTeX ignores any special meaning that a character may have. As a special case of this, LaTeX ignores *any* character in a comment.

`\newcomment{⟨comenv⟩}`

The macro `\newcomment` creates a new comment environment called ⟨*comenv*⟩. All text within the ⟨*comenv*⟩ environment will be ignored.

`\begin{comment}` anything `\end{comment}`

The class provides one comment environment, namely the `comment` environment which is specified by

```
\newcomment{comment}
```

This may be useful for 'commenting out' large chunks of a source document.

```
\commentsoff{⟨comenv⟩}
\commentson{⟨comenv⟩}
```

The declaration `\commentsoff` switches off the commenting within the ⟨*comenv*⟩ comment environment, and the declaration `\commentson` switches on commenting within the ⟨*comenv*⟩ comment environment. In either case, ⟨*comenv*⟩ must have been previously defined 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}
...
```

LaTeX provides the `\verb` command for typesetting short pieces of text verbatim; 'short' means less than one line. If you have to type a lot of verbatim bits, like macro names, it becomes very tedious to keep on doing `\verb?\amacro?`. The `\verb*` macro is similar to `\verb` except that it typesets the symbol ␣ in place of a space.

`\MakeShortVerb{`⟨*verbchar*⟩`}`
`\DeleteShortVerb{`⟨*verbchar*⟩`}`

With the `\MakeShortVerb` declaration you can use a single character instead of `\verb`⟨*char*⟩. The ⟨*verbchar*⟩ argument to `\MakeShortVerb` is a backslash followed by a single character, for example `\MakeShortVerb{\?}`. You can then use `?...?` instead of `\verb?...?`. To turn off the use of `?` in this special manner, call `\DeleteShortVerb{\?}`.

`\begin{verbatim}` anything `\end{verbatim}`
`\begin{verbatim*}` anything `\end{verbatim*}`

The `verbatim` and `verbatim*` environments are used for typesetting any length of verbatim text.

`\setverbatimfont{`⟨*font-decl*⟩`}`

The font that is used for *all* verbatim text is specified by `\setverbatimfont`. The default is:
`\setverbatimfont{\normalfont\ttfamily}`
If you wanted verbatims to be set in a smaller size, then this will do the trick:

```
\setverbatimfont{\normalfont\small\ttfamily}
```

`\tabson[`⟨*num*⟩`]`
`\tabsoff`

The standard `verbatim` environment ignores any TABs, or rather treats a sequence of TABs as a single space. In the class, if you use the declaration `\tabson` then TABs will not be ignored in subsequent verbatims. The declaration `\tabsoff` turns off tabbing inside verbatims. The default is `\tabsoff`. By default the class uses four spaces for each tab. The optional ⟨*num*⟩ argument to `\tabson` will use ⟨*num*⟩ spaces for each TAB.

```
\wrappingon
\wrappingoff
\verbatimindent
\verbatimbreakchar
```

Very occasionally it may be useful to wrap long verbatim lines round onto the following line. After the declaration `\wrappingon`, verbatim lines that would extend into the margin will be wrapped (`\wrappingoff` returns the behaviour to normal). The second and later parts of a wrapped line are indented by the length `\verbatimindent` which can be altered in the usual manner.

It may be desireable to indicate that a line is continued on the next line. That is the role of `\verbatimbreakchar`. By default this is defined as:

```
\newcommand*{\verbatimbreakchar}{\char`\%}
```

which typesets % at the end of each line that is wrapped. To have `/` at the end of each line instead, do:

```
\renewcommand*{\verbatimbreakchar}{\char`\/}
```

Note that trying both tabbing and wrapping together does not always work well. It is best to do only one out of the two, and make that tabbing.

There is some more on verbatims in later sections.

16.3 Framed boxes

You can use framed boxes, looking like the syntax boxes here, in the standard LaTeX classes but they cannot break over a page.

The `framed`, `shaded` and `leftbar` environments, which can break over a page, are from the framed package [Ars01b].

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

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

```
\FrameRule \FrameSep \FrameHeightAdjust
shadecolor
```

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

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

The background color in the `shaded` environment is specified by `shadecolor` which you have to specify using the color package [Car98a]. For example:

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

```
\frameasnormaltrue \frameasnormalfalse
```

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

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

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

The environments are a little delicate. You cannot use floats, footnotes or marginpars inside them, and they do not work in two column mode except for the standard LaTeX supplied one.

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

```
\begin{fboxverbatim} anything \end{fboxverbatim}
```

The `fboxverbatim` environment is, except for its name, identical to the `boxedverbatim` environment from the moreverb package [Fai98]. It puts a close fitting rectangular box around its contents, which are typeset verbatim.

```
\begin{boxedverbatim} anything \end{boxedverbatim}
```

The `boxedverbatim` environment has the flavour of both the `framed` and `fboxverbatim` environments, but it adds its own bells and whistles. In its simplest usage the contents of the environment are typeset verbatim in a rectangular box (or boxes as it allows pagebreaks) like a `framed` box.

```
\bvbox \bvsides \bvtopandtail \nobvbox
```

Four different kinds of framing styles are provided for the `boxedverbatim` environment. After a `\bvbox` declaration the default framing style is used, which draws rectangular boxes.

After the `\bvsides` declarations, rules are drawn on each side but there are no top or bottom rules; the converse is the `\bvtopandtail` declaration when only an initial and final horizontal rule is drawn and no side rules. No rules at all are drawn after `\nobvbox`.

```
\bvtopofpage{⟨text⟩}
```

With the default framing style (`\bvbox`) a heading can be put at the top of continuation boxes if there is too much to fit onto a page. The heading is the ⟨*text*⟩ of `\bvtopofpage`. The default specification is `\bvtopofpage{}` but if you wanted, say, to indicate that the verbatim was a continuation from the previous page you could do

```
\bvtopofpage{\begin{center}\normalfont (Continued)\end{center}}
```

```
\linenumberfrequency{⟨nth⟩}
\linenumberfont{⟨font-decl⟩}
\resetbvlinenumber
```

Exactly as in the `verse` environment line numbers are printed if the ⟨*nth*⟩ argument to `\linenumberfrequency` is greater than zero, and the numbers are set in the font defined via `\linenumberfont`. At any point outside the environment the line numbers may be reset to the initial value by `\resetbvlinenumber`.

```
\bvnumbersinside
\bvnumbersoutside
```

If the lines are numbered, the numbers can be put inside the box (`\bvnumbersinside`) or outside (`\bvnumbersoutside`). In either case, numbers are set at the left of the lines. The default is for numbers inside the box.

Verbatim tabbing, but not wrapping, applies to the `boxedverbatim` environment.

16.4 Files

Latex gives you the `\input` command to read a file but is really not much obvious help if you want to write stuff out to a file.

TeX has 16 ‘streams’ that can be open simultaneously for writing out, and there are also 16 input streams. These streams can be attached to files so that data can be read out and read in from external files.

```
\newoutputstream{⟨stream⟩}
\newinputstream{⟨stream⟩}
```

The command `\newoutputstream` creates a new stream called ⟨*stream*⟩ for writing out text and commands. The ⟨*stream*⟩ argument must be just alphabetic characters with no spaces; for example `myout`. Similarly, the command `\newinputstream` creates a new stream for reading from a file. The ⟨*stream*⟩ names must be unique — you cannot use the same name for both an input and an output stream.

If you try and create too many streams, TeX will tell you via an error message.

```
\IfStreamOpen{⟨stream⟩}{⟨true-code⟩}{⟨false-code⟩}
```

You can check if a stream ⟨*stream*⟩ is currently open by `\IfStreamOpen`. If the stream is open then the ⟨*true-code*⟩ argument will be processed, otherwise the stream is closed and the ⟨*false-code*⟩ argument is processed.

```
\openoutputfile{⟨filename⟩}{⟨stream⟩}
\closeoutputstream{⟨stream⟩}
```

The macro `\openoutputfile` opens the file called ⟨*filename*⟩ and the output stream ⟨*stream*⟩. It then attaches the file to the stream for writing. Any pre-existing contents of ⟨*filname*⟩ are deleted.

The macro `\closeoutputstream` closes the output stream ⟨*stream*⟩ and closes whatever file is currently attached to ⟨*stream*⟩. It then detaches the file from the stream.

```
\begin{writeverbatim}{⟨stream⟩} anything \end{writeverbatim}
```

The `writeverbatim` environment takes one argument, the name of an output stream, which must be open. The contents of the environment are written out verbatim to the file currently attached to the stream.

```
\addtostream{⟨stream⟩}{⟨text⟩}
```

The command `\addtostream` writes ⟨*text*⟩ to the file currently attached to the output stream ⟨*stream*⟩, which must be open. Any commands within ⟨*text*⟩ will be processed before being written. To prevent this, put `\protect` before any command that you do not want expanding.

```
\openinputfile{⟨filename⟩}{⟨stream⟩}
\closeinputstream{⟨stream⟩}
```

The macro `\openinputfile` opens the file called ⟨*filename*⟩ and the input stream ⟨*stream*⟩. It then attaches the file to the stream for reading. It is an error if ⟨*filename*⟩ can not be found.

The macro `\closeinputstream`closes the input stream ⟨*stream*⟩ and closes whatever file is currently attached to ⟨*stream*⟩. It then detaches the file from the stream.

```
\readstream{⟨stream⟩}
\readaline{⟨stream⟩}
```

The macro `\readstream` reads the contents of the file that is currently associated with the input stream ⟨*stream*⟩. This provides the same functionality as `\input{`⟨*filename*⟩`}` does.

The macro `\readaline` reads what TeX considers to be one line from the file that is currently associated with the input stream ⟨*stream*⟩. Multiple lines can be read by calling `\readaline` multiple times. A warning is issued if there are no more lines to be read (i.e., the end of the file has been reached).

```
\verbatiminput{⟨filename⟩}
\readverbatim{⟨stream⟩}
\boxedverbatiminput{⟨filename⟩}
\readboxedverbatim{⟨stream⟩}
```

The macro `\verbatiminput` is similar to `\input` except that it inputs the contents of the file ⟨*filename*⟩ as verbatim text. Similarly, `\readverbatim` reads the contents of the file that is currently associated with the input stream ⟨*stream*⟩ as verbatim text. These two commands are equivalent to

```
\begin{verbatim}
  \input{...}
  \readstream{...}
\end{verbatim}
```

except, of course, that you cannot do that.

To round things out, `\boxedverbatiminput` inputs the contents of the ⟨*filename*⟩ file as boxed verbatim text, and `\readboxedverbatim` reads the contents of the file attached to ⟨*stream*⟩ as a boxed verbatim. If you could do it, these commands are equivalent to

```
\begin{boxedverbatim}
  \input{...}
  \readstream{...}
\end{boxedverbatim}
```

Tabbing, wrapping and numbering are just as applicable to verbatim input texts as they are to the corresponding verbatim environments.

Seventeen

Miscellaneous

In which we talk of many things, but not ships or shoes or sealing wax, nor cabbages and kings.

This chapter uses the *demo* chapterstyle and the *Ruled* pagestyle. It started with the `\chapterprecis` command.

17.1 Introduction

The class provides some minor additional facilities, which are described here.

17.2 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
```

17.3 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].

```
\changemarkstrue \changemarksfalse
```

The change marking macros only work properly when the `draft` class option is used. Additionaly, the marks are only printed when the `\changemarkstrue` declaration is in effect. Using `\changemarksfalse` switches off any marking.

```
\added{⟨change-id⟩}
\deleted{⟨change-id⟩}
\changed{⟨change-id⟩}
```

Each of these macros puts a mark 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.

17.4 Trim marks

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

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

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

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

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

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

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

```
\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.

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.

17.5 Sheet numbering

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

```
\thesheetsequence
```

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

```
lastsheet
lastpage
```

The counter `lastsheet` holds the number of sheets processed during the *previous* run of LaTeX. Similarly, the counter `lastpage` holds the number of the last page processed during the previous run. Note that the last page number is not necessarily the same as the last sheet number. For example:
In this document this is sheet 252 of 301 sheets, and page 218 of 273.

The previous sentence was the result of processing the following code

```
\textit{In this document this is
        sheet \thesheetsequence\ of \thelastsheet\ sheets,
        and page \thepage\ of \thelastpage.}
```

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

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

17.6 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 introduction and the first item.

```
\noprelistbreak
```

Putting `\noprelistbreak` immediately before the `\begin{itemize}` should prevent a page break. Ideally, there sould be no blank lines between the end of the introduction and the start of the list.

17.7 Changing counters

This is effectively a bundling of the chngcntr package [Wil01e].

```
\newcounter{⟨ctr⟩}[⟨within⟩]
\thectr
```

In LaTeX a new counter called, say `ctr`, is created by the `\newcounter` command as `\newcounter{ctr}`. If the optional ⟨*within*⟩ argument is given, the counter ⟨*ctr*⟩ is reset to zero each time the counter called `within` is changed; the `within` counter must exist before it is used as the optional argument. The command `\thectr` typesets the value of the counter `ctr`. This is automatically defined for you by the `\newcounter` command to typeset arabic numerals.

```
\counterwithin{⟨ctr⟩}{⟨within⟩}
\counterwithin*{⟨ctr⟩}{⟨within⟩}
```

The `\counterwithin` macro makes a ⟨*ctr*⟩ that has been initially defined by `\newcounter{ctr}` act as though it had been defined by `\newcounter{ctr}[within]`. It also redefines the `\thectr` command to be `\thewithin.\arabic{ctr}`. The starred version of the command does nothing to the original definition of `\thectr`.

```
\counterwithout{⟨ctr⟩}{⟨within⟩}
\counterwithout*{⟨ctr⟩}{⟨within⟩}
```

The `\counterwithout` macro makes a ⟨*ctr*⟩ 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`.

17.8 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 repetoire.

```
\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.

17.9 Changing macros

Commands are provided for extending simple macro definitions. Get the patchcmd package [Dow00] if you need to make other additions to 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]{# is a violinist in spite of being tone deaf}
```

and

```
\newcommand{\mymacro}[1]{#1 is a violinist}
\addtoiargdef{\mymacro}{}{ in spite of being tone deaf}
```

The `\addtoiargdef` (and `\addtodef`) commands can be applied several times to the same ⟨*macro*⟩. Revising the previous example we could have

```
\newcommand{\mymacro}[1]{#1 is a violinist}
\addtoiargdef{\mymacro}{Although somewhat elderly, }%
                       { in spite of being tone deaf}
\addtoiargdef{\mymacro}{}{ and a bagpiper}
```

which is equivalent to

```
\newcommand{\mymacro}[1]{%
  Although somewhat elderly, #1 is a violinist
  in spite of being tone deaf and a bagpiper}
```

The ⟨*prepend*⟩ and ⟨*append*⟩ arguments may include LaTeX code, as shown in this extract from the class code:

```
\addtoiargdef{\date}{}{%
  \begingroup
    \renewcommand{\thanks}[1]{}
    \renewcommand{\thanksmark}[1]{}
    \renewcommand{\thanksgap}[1]{}
    \protected@xdef\thedate{#1}
  \endgroup}
```

Note that in the case of `\addtoiargdef` an argument can also refer to the original argument of the ⟨*macro*⟩.

```
\addtodef*{⟨macro⟩}{⟨prepend⟩}{⟨append⟩}
\addtoiargdef*{⟨macro⟩}{⟨prepend⟩}{⟨append⟩}
```

These starred versions are for use when the original ⟨*macro*⟩ was defined via `\newcommand*`. Using the starred versions is like using `\renewcommand*` and the unstarred versions are like having used `\renewcommand`. It is the version (starred or unstarred) of a sequence of `\addto...` commands that counts when determining whether the equivalent `\renew...` is treated as starred or unstarred.

The `\addto...` macros cannot be used to delete any code from ⟨*macro*⟩ nor to add anything except at the start and end. Also, in general, they cannot be used to change the definition of a macro that takes an optional argument, or that is starred.

17.10 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
  ...
}
```

17.11 Odd/even page checking

It is difficult to check robustly if the current page is odd or even but the class does provide a robust method based on writing out a label and then checking the page reference for the label. This requires at least two LaTeX runs to stabilise. This has been extracted from the original `chngpage` package [Wil01b].

```
\checkoddpage
\ifoddpage
\strictpagechecktrue \strictpagecheckfalse
```

The macro `\checkoddpage` sets `\ifoddpage` to TRUE if the page number is odd, otherwise it sets it to FALSE (the page number is even). The robust checking methos involves writing and reading labels, which is what is done after the command `\strictpagechecktrue` is issued. 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` commmand was issued. The simple, but faster, page checking method is used after the `\strictpagecheckfalse` 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.

17.12 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[1] of `\cleartooddpage` and `\cleartoevenpage`. For example, if you wanted the Table of Contents to start on a verso page, like in *The TeXbook* [Knu84], then do this:

```
\cleartoverso
\tableofcontents
```

17.13 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}} ->` 223rd
- The arabic representation of a counter, e.g., `\ordinal{\thepage} ->` 223rd
  However, if the representation of a counter is not completely in arabic digits, such as `\thesection` which here prints as 17.13, it will produce odd errors or peculiar results if it is used as the argument. For instance:
  `\ordinal{\thesection} ->` .1317th

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}{,}
```

1 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

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]{\textsuperscript{#1}}\fordinal{2} ->` 2^{nd}
`\ordinal{1234567} ->` 1234567^{th}
`This is the \ordinal{\value{chapter}} chapter. ->` This is the 17^{th} 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^{st}, `\iindstring` (default: nd) for ordinals like 22^{nd}, and `\iiirdstring` (default: rd) for ordinals like 23^{rd}.

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
`The minimum number in TeX is \numtoname{-2147483647}`

`(i.e., \fcardinal{-2147483647}) ->`
The minimum number in TeX is minus two billion, one hundred and forty-seven million, four hundred and eighty-three thousand, six hundred and forty-seven (i.e., -2,147,483,647)

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

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

For example:
`This is the \ordinaltoname{\value{chapter}} chapter. ->` This is the seventeenth chapter.

```
\namenumberand \namenumbercomma \tensunitsep
```

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

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

The next example shows how to achieve American typesetting.

```
\renewcommand*{\namenumberand}{ }
\renewcommand*{\namenumbercomma}{ }
The maximum number in TeX is \numtoname{2147483647} (i.e., \cardinal{2147483647}). ->
```

The maximum number in TeX is two billion one hundred forty-seven million four hundred eighty-three thousand six hundred forty-seven (i.e., 2147483647).

```
\minusname \lcminusname \ucminusname
```

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

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

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

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

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

There is a group of macros that hold the names for the numbers. To typeset named numbers in a language other than English these will have to be changed as appropriate.

You can find them in the class and patch code. The actual picking and ordering of the names is done by an internal macro called `\n@me@number`. If the ordering is not appropriate for a particular language, that is the macro to peruse and modify. Be prepared, though, for the default definitions to be changed in a later release in case there is a more efficient way of implementing their functions.

Fractions

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

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

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

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

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

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

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

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

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

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

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

17.14 An array data structure

The class includes some macros for supporting the `patverse` environment which may be more generally useful.

```
\newarray{⟨arrayname⟩}{⟨low⟩}{⟨high⟩}
```

`\newarray` defines the ⟨*arrayname*⟩ array, where ⟨*arrayname*⟩ is a name like `MyArray`. The lowest and highest array indices are set to ⟨*low*⟩ and ⟨*high*⟩ respectively, where both are integer numbers.

```
\setarrayelement{⟨arrayname⟩}{⟨index⟩}{⟨text⟩}
\getarrayelement{⟨arrayname⟩}{⟨index⟩}{⟨result⟩}
```

The `\setarrayelement` macro sets the ⟨*index*⟩ location in the ⟨*arrayname*⟩ array to be ⟨*text*⟩. Conversely, `\getarrayelement` sets the parameterless macro ⟨*result*⟩ to the contents of the ⟨*index*⟩ location in the ⟨*arrayname*⟩ array. For example:

```
\setarrayelement{MyArray}{23}{$2^{23}$}
\getarrayelement{MyArray}{23}{\result}
```

will result in `\result` being defined as `\def\result{$2^{23}$}`.

```
\checkarrayindex{⟨arrayname⟩}{⟨index⟩}
\ifbounderror
```

`\checkarrayindex` checks if ⟨*arrayname*⟩ is an array and if ⟨*index*⟩ is a valid index for the array. If both conditions hold then `\ifbounderror` is set FALSE, but if either ⟨*arrayname*⟩ is not an array or, if it is, ⟨*index*⟩ is out of range then `\ifbounderror` is set TRUE.

```
\stringtoarray{⟨arrayname⟩}{⟨string⟩}
\arraytostring{⟨arrayname⟩}{⟨result⟩}
```

The macro `\stringtoarray` puts each character from ⟨*string*⟩ sequentially into the ⟨*arrayname*⟩ array, starting at index 1. The macro `\arraytostring` assumes that ⟨*arrayname*⟩ is an array of characters, and defines the macro ⟨*result*⟩ to be that sequence of characters. For example:

```
\stringtoarray{MyArray}{Chars}
\arraytostring{MyArray}{\MyString}
```

is equivalent to `\def\MyString{Chars}`.

```
\checkifinteger{⟨num⟩}
\ifinteger
```

The command `\checkifinteger` ckecks if ⟨*num*⟩ is an integer (not less than zero). If it is then `\ifinteger` is set TRUE, otherwise it is set FALSE.

17.15 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.

By default pdfLaTeX produces a `.dvi` file but there may be a configuration file on your system that makes pdfLaTeX default to producing a `.pdf` instead. To ensure a `.pdf` output file you have to put the incantation `\pdfoutput=1` near the start of the preamble. Conversely, to ensure a `.dvi` file put `\pdfoutput=0` instead. Unfortunately this command is unknown to LaTeX so if you used LaTeX it would hiccup.

```
\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 e.g., \pdfoutput=1 or \pdfoutput=0
\else
  code for LaTeX only
\fi
```

If there is no LaTeX specific code then don't write the `\else` part.

17.16 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[2]. 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.

17.17 Minor space adjustment

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

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

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

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

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

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

2 This statement ignores any attempts to stretch the baseline.

17.18 Cross references

LaTeX supplies the `\ref` and `\pageref` commands for cross referencing to a label or a page which has a label on it.

```
\fref{⟨label⟩} \figurerefname
\tref{⟨label⟩} \tablerefname
\pref{⟨label⟩} \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 13.1 on page 174.

```
\Pref{⟨label⟩} \partrefname
\Cref{⟨label⟩} \chapterrefname
\Sref{⟨label⟩} \sectionrefname
```

Also provided are named references to labelled Part (`\Pref`), Chapter (`\Cref`) and sectional (`\Sref`) divisions. These are all defined like

```
\newcommand{\Sref}[1]{\sectionrefname\ref{#1}}
```

with no tie between the name and the `\ref`.

In this document

```
‘In \Cref{chap:misc} there is a section
(\Sref{sec:xrefthis}) about cross references.’
```

is typeset as:

‘In Chapter 17 there is a section (§17.18) about cross references.’

It can be useful to refer to parts of a document by name rather than 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.

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

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

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

Labelling for `\titleref` may be applied to:

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

Item 2 in section 17.18 mentions captions while item Cross references in the same section *Cross references* lists other things.

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

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

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

```
\currenttitle
```

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

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

This sentence in the section titled 'Cross references' is an example of the use of the command `\currenttitle`.

```
\theTitleReference{⟨num⟩}{⟨text⟩}
```

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

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

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

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

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

The commands `\titleref`, `\theTitleReference` and `\currenttitle` are direct equivalents of those in the titleref package [Ars01a].

```
\namerefon \namerefoff
```

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

17.19 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. The following lists the macros and their default values.

`\contentsname`	Contents
`\listfigurename`	List of Figures
`\listtablename`	List of Tables
`\abstractname`	Abstract
`\partname`	Part
`\chaptername`	Chapter
`\appendixname`	Appendix
`\appendixtocname`	Appendices
`\appendixpagename`	Appendices
`\bibname`	Bibliography
`\indexname`	Index
`\figurename`	Figure
`\tablename`	Table
`\figurerefname`	Figure

`\tablerefname`	Table
`\pagename`	page
`\pagerefname`	page
`\partrefname`	Part (defined as `\newcommand{\partrefname}{Part~}`)
`\chapterrefname`	Chapter (defined as `\newcommand{\chapterrefname}{Chapter~}`)
`\sectionrefname`	§(defined as `\newcommand{\sectionrefname}{\S}`)

The above definitions are simple — for example

```
\newcommand{\partname}{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.

17.20 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$^{\text{TM}}$, but the `\textregistered` command produces®. When I wanted to use the registered trademark symbol it needed to be typeset like a superscript instead of on the baseline. The `\textsuperscipt` macro typesets its argument like a superscript, so using

```
\textsuperscript{\textregistered}
```

gave the required result$^{®}$.

AN EXAMPLE DESIGN

18.1 INTRODUCTION

In this chapter I will work through a reasonably complete design exercise. The chapter is typeset using the results of the exercise.

Rather than trying to invent something myself I am taking the design of Bringhurst's *The Elements of Typographic Style* [Bri92] 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.

18.2 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 diagramatically in Figure 2.2 on page 16, but further details need to be described for those not fortunate enough to have a copy of their own.

The trimmed size of a page is 23 by 13.3cm. The foredge is 3.1cm and the top margin is 1.9cm.

As already noted, the font for the main text is Minion, with 12pt leading on a 21pc measure with 42 lines per page. For the purposes of this exercise I will assume that Minion can be replaced by Computer Modern Roman at 10pt (like this manual). The captions to figures and tables are unnamed and unnumbered and typeset in Syntax. The captions give the appearance of being in a smaller font size than the main text, which is often the case. I'll assume that the `\small` `\sfseries` CMR 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 foredge. 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 foredge 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 Table of Contents.

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.

Specifying the page and typeblock

18.3 SPECIFYING THE PAGE AND TYPEBLOCK

The first and second things to do are to specify the sizes of the page after trimmming and the typeblock. The trimmed size is easy as we have the dimensions.

```
\settrimmedsize{23cm}{13.3cm}{*}
```

However, there is a trick to setting the height of the typeblock in terms of lines of text. The height calculation ensures that an integral number of lines can fit in the typeblock, and as well as the specified height for the block some more height is added so that the final height is approximately measured from the base of the bottom line of text to the top of the first line of text. This is a complicated way of saying that if you want N lines of text, only ask for $N-1$. We want 42 lines so we give the height of the typeblock as 41 times the distance between two normal text lines (i.e., times `\onelineskip`)

```
\settypeblocksize{41\onelineskip}{21pc}{*}
```

To make life easier, we'll do no trimming of the top of the stock

```
\setlength{\trimtop}{0pt}
```

but will trim the foredge. The next set of calculations first sets the value of the `\trimedge` to be the `\stockwidth`; subtracting the trimmed `\paperwidth` then results in `\trimedge` being the amount to trim off the foredge.

```
\setlength{\trimedge}{\stockwidth}
\addtolength{\trimedge}{-\paperwidth}
```

The sizes of the trimmed page and the typeblock have now been specified. The typeblock is now positioned on the page. The sideways positioning is easy as we know the foredge 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
```

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 236 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.

18.4 SPECIFYING THE SECTIONAL TITLING STYLES

An example design

The chapter style

Recapping, chapter titles are in small caps, lowercase, in a larger font than for the main text, and a rule is placed between the title and the typeblock. The total vertical space used by a chapter title is three text lines. Chapters are not numbered in the text but are in the Table of Contents. Titles in the ToC are in mixed case.

The definition of the chapterstyle is remarkably simple, as shown below.

```
%% Bringhurst chapter style
\makechapterstyle{bringhurst}{%
  \renewcommand{\chapterheadstart}{}
  \renewcommand{\printchaptername}{}
  \renewcommand{\chapternamenum}{}
  \renewcommand{\printchapternum}{}
  \renewcommand{\afterchapternum}{}
  \renewcommand{\printchaptertitle}[1]{%
    \raggedright\Large\scshape\MakeLowercase{##1}}
  \renewcommand{\afterchaptertitle}{%
    \vskip\onelineskip \hrule\vskip\onelineskip}
}
```

Most of the specification consists of nulling the majority of the normal LaTeX specification, and modifying just two elements.

The chapter title (via `\printchaptertitle`) is typeset raggedright using the `\Large` smallcaps fonts. The `\MakeLowercase` macro is used to ensure that the entire title is lowercase before typesetting it. Titles are input in mixed case.

After the title is typeset the `\afterchaptertitle` macro specifies that one line is skipped, a horizontal rule is drawn and then another line is skipped.

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:

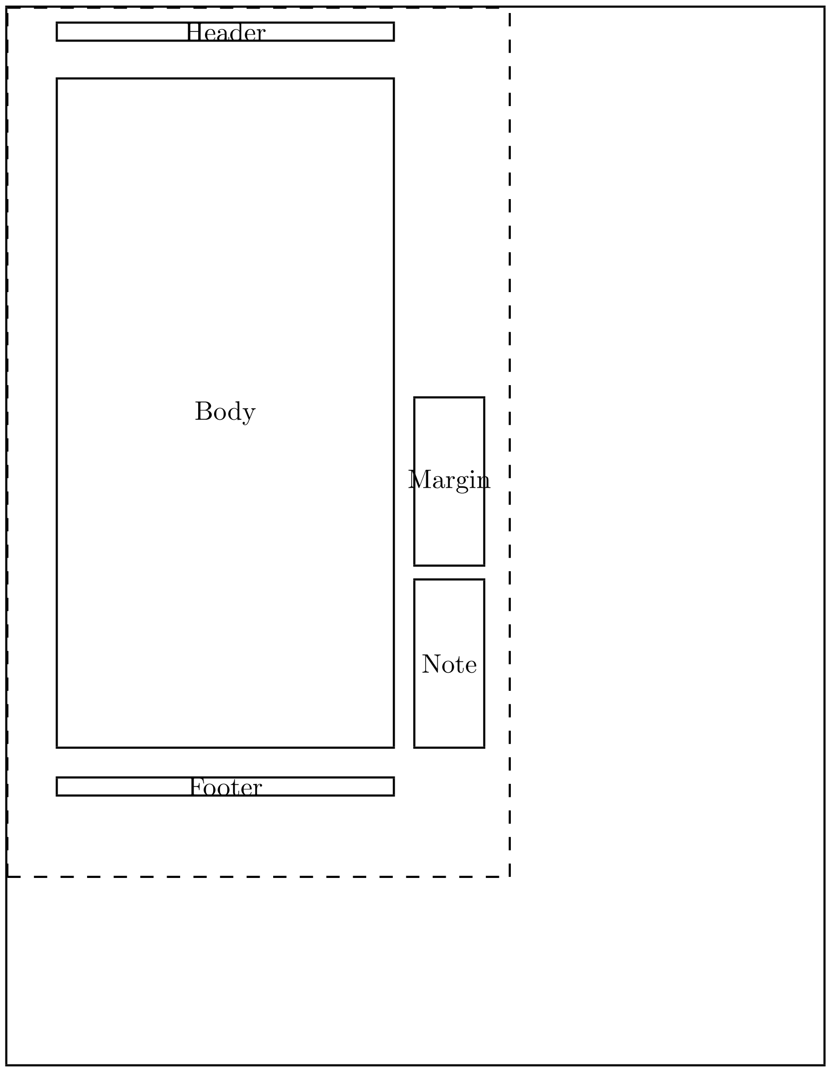
```
\setsecheadstyle{\raggedright\scshape\MakeLowercase}
  \setbeforesecskip{-\onelineskip}
  \setaftersecskip{\onelineskip}
```

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

Specifying the sectional titling styles

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



Lengths are to the nearest pt.

```
\stockheight = 795pt       \stockwidth = 614pt
\pageheight = 654pt        \pagewidth = 378pt
\textheight = 502pt        \textwidth = 252pt
\trimtop = 0pt             \trimedge = 236pt
\uppermargin = 54pt        \spinemargin = 38pt
\headheight = 12pt         \headsep = 30pt
\footskip = 36pt           \marginparsep = 17pt
\marginparpush = 12pt      \columnsep = 10pt
\columnseprule = 0.0pt
```

An illustration of Bringhurst's page layout style when printed on US letter paper stock. Also shown are the values used for the page layout parameters for this design.

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}
```

An example design

As in the redefinition of the `\section` style, there are fixed spaces before and after the title text. The title is typeset (`\setsubsecheadstyle`) raggedright in a normal sized italic font. The macro `\sethangfrom` is used to to redefine the internal `\@hangfrom` macro so that the title and number are typeset as a block paragraph instead of the default hanging paragraph style. Note the use of the double `##` mark for denoting the position of the argument to `\@hangfrom`.

18.5 SPECIFYING THE PAGESTYLE

The pagestyle is perhaps the most interesting aspect of the exercise. Instead of the chapter and section titles being put at the top of the pages they are put in the margin starting about seven lines below the top of the typeblock. The folios are put at the bottom of the page aligned with the outside of the typeblock.

As the folios are easy, we'll deal with those first.

```
%% Bringhurst page style
\makepagestyle{bringhurst}
\makeevenfoot{bringhurst}{\thepage}{}{}
\makeoddfoot{bringhurst}{}{}{\thepage}
```

Putting text at a fixed point on a page is typically done by first putting the text into a zero width picture (which as far as LaTeX is concerned takes up zero space) and then placing the picture at the required point on the page. This can be done by hanging it from the header.

We might as well treat the titles so that they will align with any marginal notes, which are `\marginparsep` (17pt) into the margin and `\marginparwidth` (51pt) wide. Earlier in the manual I defined two lengths called `\pwlayi` and `\pwlayii` which are no longer used. I will use these as scratch lengths in performing some of the necessary calculations.

For the recto page headers the picture will be the ⟨*right*⟩ part of the header and for the verso pages the picture will be the ⟨*left*⟩ part of the header, all other parts being empty.

For the picture on the ⟨*right*⟩ the text must be 17pt to the right of the origin, and some distance below the origin. From some experiments, this distance turns out to be the `\headsep` plus the `\topskip` plus 7.3 lines, which is calculated as follows:

```
\setlength{\pwlayi}{\headsep}
\addtolength{\pwlayi}{\topskip}
\addtolength{\pwlayi}{7.3\onelineskip}
```

There is a nifty internal LaTeX macro called `\strip@pt` which you probably haven't heard about, and I have only recently come across. What it does is strip the 'pt' from a following length, reducing it to a plain real number. Remembering that the default `\unitlength` is 1pt we can do the following, while making sure that the current `\unitlength` *is* 1pt:

```
\makeatletter
\newcommand{\bringpicr}[1]{%
  \setlength{\unitlength}{1pt}
  \begin{picture}(0,0)
    \put(\strip@pt\marginparsep, -\strip@pt\pwlayi){%
      \begin{minipage}[t]{\marginparwidth}
        \raggedright\itshape #1
      \end{minipage}}
  \end{picture}
}
\makeatother
```

Specifying the pagestyle

The new macro `\bringpicr{`⟨*text*⟩`}` puts ⟨*text*⟩ into a `minipage` of width `\marginparwidth`, typeset raggedright in an italic font, and puts the top left of the minipage at the position (`\marginparsep`, `-\pwlayi`) in a zero width picture.

We need a different picture for the ⟨*left*⟩ as the text needs to be typeset raggedleft with the right of the text 17pt from the left of the typeblock. I will use the length `\pwlayii` to calculate the sum of `\marginparsep` and `\marginparwidth`. Hence:

```
\makeatletter
\setlength{\pwlayii}{\marginparsep}
\addtolength{\pwlayii}{\marginparwidth}
\newcommand{\bringpicl}[1]{%
  \setlength{\unitlength}{1pt}
  \begin{picture}(0,0)
    \put(-\strip@pt\pwlayii, -\strip@pt\pwlayi){%
      \begin{minipage}[t]{\marginparwidth}
        \raggedleft\itshape #1
      \end{minipage}}
  \end{picture}
}
\makeatother
```

The new macro `\bringpicl{`⟨*text*⟩`}` puts ⟨*text*⟩ into a `minipage` of width `\marginparwidth`, typeset raggedleft in an italic font, and puts the top left of the minipage at the position (`-(\marginparsep+\marginparwidth)`, `-\pwlayi`) in a zero width picture.

Now we can proceed with the remainder of the pagestyle specification. The next bit puts the chapter and section titles into the `\...mark` macros.

```
\makeatletter
\makepsmarks{bringhurst}{%
  \let\@mkboth\markboth
  \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}}
```

18.6 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.

An example design

```
\let\caption\legend
```

An aside: I initially used the default caption style (block paragraph) for the diagram on page 236, but this looked unbalanced so now it has the last line centered. As a float environment, like any other environment, forms a group, you can make local changes within the float. I actually did it like this:

```
\begin{figure}
\captiontitlefont{\small\sffamily}
\captionstyle{\centerlastline}
...
\legend{...} \label{...}
\end{figure}
```

For fine typesetting you may wish to change the style of particular captions. The default style for a single line caption works well, but for a caption with two or three lines either the centering or centerlastline style might look better. A very long caption is again probably best done in a block paragraph style.

Only chapter titles are included in the ToC. To specify this we use the `\settocdepth` command.

```
\settocdepth{chapter}
```

The ToC is typeset raggedright with no leaders and the page numbers coming immediately after the chapter title. This is specified via:

```
\renewcommand{\cftchapterfont}{\normalfont}
\renewcommand{\cftchapterpagefont}{\normalfont}
\renewcommand{\cftchapterpresnum}{\bfseries}
\renewcommand{\cftchapterleader}{}
\renewcommand{\cftchapterafterpnum}{\cftparfillskip}
```

18.7 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. The code could just as easily be put into a package called, say, bringhurst. That is, by putting all the code, except for the `\makeatletter` and `\makeatother` commands, into a file called `bringhurst.sty`. It is a good idea also to end the code in the file with `\endinput`; LaTeX stops reading the file at that point and will ignore any possible garbage after `\endinput`.

Preamble or package?

You then use the bringhurst package just like any other by putting

```
\usepackage{bringhurst}
```

in your document's preamble.

Appendices

A Packages and macros

This chapter is typeset with the *section* chapterstyle, otherwise it uses the layout defined in Chapter 18.

A.1 INTRODUCTION

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.2 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.

The package that I used and you most likely do not have is layouts [Wil99a]. I used it for all the layout diagrams. For example, Figure 8.1 and Figure 8.2 were drawn simply by:

```
\begin{figure}
\centering
\setlayoutscale{1}
\drawparameterstrue
\drawheading{}
\caption{Displayed sectional headings} \label{fig:displaysechead}
\end{figure}

\begin{figure}
\centering
\setlayoutscale{1}
\drawparameterstrue
```

```
\runinheadtrue
\drawheading{}
\caption{Runin sectional headings} \label{fig:runsechead}
\end{figure}
```

The package also lets you try experimenting with different layout parameters and draw diagrams showing what the results would be in a document.

The version of layouts used for this manual is v2.4 dated 2001/04/30. Earlier versions will fail when attempting to draw some figures (e.g., to draw Figure 6.2).

Macros

A.3 MACROS

The preamble of the manual contains 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.

To get the whole glory you will have to read the preamble, but I show a few of the macros below as they may be of more general interest.

\pstyle{⟨*style*⟩}

The command `\pstyle` prints its argument in the slanted font used for pagestyles and also makes a pagestyle entry in the index. Its definition is

```
\newcommand{\pstyle}[1]{\textsl{#1}\index{#1pages@\textsl{#1} (pagestyle)}}
```

The first part prints the argument in the text and the second adds an entry to the `.idx` file. The fragment `#1pages` is what the MAKEINDEX program will use for sorting entries, and the fragment following the `@` character is what will be put into the index.

\cstyle{⟨*style*⟩}

The command `\cstyle` prints its argument in the slanted font used for chapterstyles and also makes a chapterstyle entry in the index. Its definition is

```
\newcommand{\cstyle}[1]{\textsl{#1}\index{#1chaps@\textsl{#1} (chapterstyle)}}
```

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 `companionchaps` and `companionpages` 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

```
\newenvironment{syntax}{\begin{center}
                        \begin{tabular}{|p{0.9\linewidth}|} \hline}%
                       {\hline
                        \end{tabular}
                        \end{center}}
```

It is implemented in terms of the `tabular` environment, 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.

Packages and macros

`\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. It is defined with the help of the `verbatim` package.

At the bottom the environment is defined in terms of a `list`, although that is not obvious from the code; for details see the `verbatim` package documentation [SRR99]. I wanted the environment to be a tight list so started off by defining some two helper items.

```
% \@zeroseps sets list before/after skips to minimum values
\newcommand{\@zeroseps}{\setlength{\topsep}{\z@}
                        \setlength{\partopsep}{\z@}
                        \setlength{\parskip}{\z@}}
% \gparindent is the \parindent for the body text
\newlength{\gparindent} \setlength{\gparindent}{\parindent}
```

The macro `\@zeroseps` sets the before, after and middle skips in a list to 0pt (`\z@` is shorthand for 0pt). The value of `\parindent` is saved in `\gparindent`, because this will be the line indentation in the environment.

```
% Now we can do the new lcode verbatim environment.
% This has no extra before/after spacing.
\newenvironment{lcode}{\@zeroseps
  \renewcommand{\verbatim@startline}%
    {\verbatim@line{\hskip\gparindent}}
  \small\setlength{\baselineskip}{\onelineskip}\verbatim}%
  {\endverbatim
   \vspace{-\baselineskip}\noindent
}
```

The fragment `{\hskip\gparindent}` puts `\gparindent` space at the start of each line.

The fragment `\small\setlength{\baselineskip}{\onelineskip}` sets the font size to be `\small`, which has a smaller `\baselineskip` than the normal font, but this is corrected for by changing the local `\baselineskip` to the normal skip, `\onelineskip`. At the end of the environment there is a negative space of one line to compensate for a one line space that LaTeX inserts.

The two versals in §2.3 were typeset with macros defined in the preamble. The first and poorer of the two used the `\versal` macro. The second used the `\drop` macro which was written for LaTeX v2.09 by David Cantor and Dominik Wujastyk in 1998. 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.

Macros

Bibliography

CTAN is the Comprehensive TeX Archive Network. Information on how to access CTAN is available at `http://www.tug.org`.

[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 framed package*. July, 2001. (Available from CTAN as `/macros/latex/contrib/misc/framed.sty`)

[Ars01b] Donald Arseneau. *The chapterbib package*. September, 2001. (Available from CTAN as `/macros/latex/contrib/cite/chapterbib.sty`)

[Bar92] Helen Barolini. *Aldus and his Dream Book*. Italica Press (ISBN 0–934977–22–4), 1992.

[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/makeindx.dtx(ins)`)

[Bra97] Johannes Braams. *The alltt environment*. June, 1997. (Available from CTAN as `/macros/latex/base/alltt.dtx(ins)`)

[Bri92] Robert Bringhurst. *The Elements of Typographic Style*. Hartley & Marks (ISBN 0–88179–033–8), 1992.

[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`)

[Car98a] David Carlisle. *The color package.* May, 1998. (Available from CTAN in `/macros/latex/required/tools`)

[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`)

[Car99a] David Carlisle. *The tabularx package.* January, 1999. (Available from CTAN in `/macros/latex/required/tools`)

[Car01] David Carlisle. *The dcolumn package.* May, 2001. (Available from CTAN in `/macros/latex/required/tools`)

[CR99] David Carlisle and Sebastian Rahtz. *The graphicx package.* February, 1999. (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 (ISBN 0–226–10389–7) 1993.

[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 (ISBN 0–387–97993–X), 1996.

[Cra92] James Craig. *Designing with Type: A Basic Course in Typography.* Watson-Guptill, NY, 1992.

[Dal99] Patrick W. Daly. *Natural Sciences Citations and References.* May, 1999. (Available from CTAN in `/macros/latex/contrib/natbib`)

[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 (ISBN 0–88179–119–9), 1996.

[Dow98] Geoffrey Dowding. *An Introduction to the History of Printing Types.* The British Library and Oak Knoll Press (ISBN 0–7123–4563–9 UK, 1–884718–44–2 USA), 1998.

[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/`).

[Fai00] Robin Fairbairns. *footmisc — a portmanteau package for customising footnotes in LaTeX2e*. March, 2000. (Available from CTAN in `/macros/latex/contrib/footmisc`)

[Fai98] Robin Fairbairns. *The moreverb package*. December, 1998. (Available from CTAN in `/macros/latex/contrib/moreverb`)

[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`)

[Fli98] Daniel Flipo. *Typesetting 'lettrines' in LaTeX2e documents*. March, 1998. (Available from CTAN in `/macros/latex/contrib/lettrine`)

[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. (ISBN 1–57912–023–7), 1998.

[Gar66] Martin Gardner. *More Mathematical Puzzles and Diversions*. Penguin Books (ISBN 0–14–020748–1), 1966.

[GMS94] Michel Goossens, Frank Mittelbach and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley Publishing Company (ISBN 0–201–54199–8), 1994.

[GRM97] Michel Goossens, Sebastian Rahtz and Frank Mittelbach. *The LaTeX Graphics Companion: Illustrating Documents with TeX and PostScript*. Addison-Wesley Publishing Company (ISBN 0–201–85469–4), 1997.

[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 Publishing Company (ISBN 0–201–43311–7), 1999.

[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. Maggs (Eds). *Texts on Type: Critical Writings on Typography*. Allworth Press (ISBN 1–58115–082–2), 2001.

[Hoe98] Alan Hoenig. *TeX Unbound: LaTeX and TeX strategies for fonts, graphics, and more*. Oxford University Press (ISBN 0–19–509686–X), 1998.

[HK75] J. K. Hvistendahl and M. R. Kahl. 'Roman vs. sans serif body type: Readability and reader prference'. *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`)

[Knu84] Donald E. Knuth. *The TeXbook.* Addison-Wesley Publishing Company (ISBN 0–201–13448–9), 1984.

[Lam94] Leslie Lamport. *LaTeX: A Document Preparation System.* Addison-Wesley Publishing Company (ISBN 0–201–52983–1), 1994.

[LMB99] Leslie Lamport, Frank Mittelbach and Johannes Braams. *Standard document classes for LaTeX version 2e.* September, 1999. (Available from CTAN as `/macros/latex/base/classes.dtx`)

[Law90] Alexander Lawson. *Anatomy of a Typeface.* David R. Godine (ISBN 0–87923–333–8), 1990.

[Leu92] Mary-Claire van Leunen. *A Handbook for Scholars.* Oxford University Press (ISBN 0–19–506954–4), 1992.

[Lon91] F. W. Long. *multind.* August, 1991. (Available from CTAN as `/macros/latex209/contrib/misc/multind.sty`)

[McD98] Rowland McDonnell. *The sectsty package.* November, 1998. (Available from CTAN in `/macros/latex/contrib/secsty`)

[McL75] Ruari McLean. *Jan Tschichold: Typographer.* David R. Godine (ISBN 0–87923–841–0), 1975.

[McL80] Ruari McLean. *The Thames & Hudson Manual of Typography.* Thames & Hudson (ISBN 0–500–68022–1), 1980.

[McL95] Ruari McLean (Ed). *Typographers on Type.* W. W. Norton & Co. (ISBN 0–393–70201–4), 1995.

[Mit95] Frank Mittelbach. *The doc and shortvrb packages.* May, 1995. (Available from CTAN in `/macros/latex/base`)

[Mit95] Frank Mittelbach. *The doc and shortvrb packages.* May, 1995. (Available from CTAN in `/macros/latex/base`)

[Mit98] Frank Mittelbach. *An environment for multicolumn output.* January, 1998. (Available from CTAN in `/macros/latex/required/tools`)

[MC98] Frank Mittelbach and David Carlisle. *A new implementation of LaTeX's tabular and array environment.* May, 1998. (Available from CTAN in `/macros/latex/required/tools`)

[MC00] Frank Mittelbach and David Carlisle. *The fixltx2e package.* September, 2000. (Available from CTAN in `/macros/latex/base`)

[Mor99] Stanley Morison. *A Tally of Types.* David R. Godine (ISBN 1–56792–004–7), 1999.

[NG98] Rolf Niespraschk and Hubert Gäßlein. *The sidecap package.* June, 1998. (Available from CTAN in `/macros/latex/contrib/sidecap`)

[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`)

[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.

[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. (no ISBN), 1943. (Reissued by Dover, 1979, ISBN 0–486–23817–2)

[Rog49] Bruce Rogers. *Centaur Types*. October House (no ISBN), 1949.

[SW94] Douglas Schenck and Peter Wilson. *Information Modeling the EXPRESS Way*. Oxford University Press (ISBN 0–19–508714–3), 1994.

[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.

[Thi99] Christina Thiele. 'The Treasure Chest: Package tours from CTAN', *TUGboat*, vol. 20, no. 1, pp 53–58, March 1999.

[Tin63] Miles A. Tinker. *Legibility of Print*. Books on Demand (University Microfilms International), 1963.

[Tsc91] Jan Tschichold. *The Form of the Book*. Lund Humphries (ISBN 0–85331–623–6), 1991.

[Ume99] Hideo Umeki. *The geometry package*. November, 1999. (Available from CTAN in `/macros/latex/contrib/geometry`)

[Whe95] Colin Wheildon. *Type & Layout*. Strathmore Press (ISBN 0–9624891–5–8), 1995.

[Wil93] Adrian Wilson. *The Design of Books*. Chronicle Books (ISBN 0–8118–0304–X), 1993.

[Wil99a] Peter Wilson. *The layouts package*. January, 1999. (Available from CTAN in `/macros/latex/contrib/layouts`)

[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`)

[Wil03] Peter Wilson. *ledmac: A presumptuous attaempt to port EDMAC and TABMAC to LaTeX* November, 2003. (Available from CTAN in `/macros/latex/contrib/ledmac`)

[Wil??] Peter Wilson. *A Rumour of Humour: A scientist's commonplace book*. To be published?

[Zac69] B. Zachrissom. *Studies in the Legibility of Printed Text.* Almqvist & Wiksell, Stockholm, 1969.

[Zap00] Hermann Zapf. *The Fine Art of Letters.* The Grolier Club (ISBN 0–910672–35–0), 2000.

Index

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

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 Computer Modern Roman designed by Donald Knuth. Other fonts include Sans, Smallcaps, Italic, Slanted and Typewriter, all from Knuth's Computer Modern family.