Trademark notices

METAfont™ is a trademark of Addison-Wesley Publishing Company.
TeX™, \TeX\textsuperscript{\textregistered}, and \LaTeX\textsuperscript{\textregistered} are trademarks of the American Mathematical Society.
Lucida™ is a trademark of Bigelow & Holmes.
Microsoft®, MS-DOS®, Windows®, Internet Explorer® are registered trademarks of Microsoft Corporation.
PostScript®, Acrobat Reader®, Acrobat logo® are registered trademarks and PDF™ a trademark of Adobe Systems Incorporated.
UNIX® is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company, Limited.
VAX™ and VMS™ are trademarks of Digital Equipment Corporation.
IBM® is a registered trademark and tech\textregistered\textsuperscript{\textcopyright} a trademark of International Business Machines Corporation.
Netscape™ and Netscape Navigator™ are trademarks of Netscape Communications Corporation.
TrueType™ is a trademark and Apple® and Macintosh® are registered trademarks of Apple Computer Inc.
Preface

A new edition to *A Guide to **LaTeX*** begs the fundamental question: Has **LaTeX** changed so much since the appearance of the third edition in 1999 that a new release of this manual is justified?

The simple answer to that question is ‘Well . . .’ In 1994, the **LaTeX** world was in upheaval with the issue of the new version **LaTeX** \(2\varepsilon\), and the second edition of the *Guide* came out just then to act as the bridge between the old and new versions. By 1998, the initial teething problems had been worked out and corrected through semi-annual releases, and the third edition could describe an established, working system. However, homage was still paid to the older 2.09 version since many users still employed its familiar syntax, although they were most likely to be using it in a **LaTeX** \(2\varepsilon\) environment. **LaTeX** has now reached a degree of stability that since 2000 the regular updates have been reduced to annual events, which often appear months after the nominal date, something that does not worry anyone. The old version 2.09 is obsolete and should no longer play any role in such a manual. In this fourth edition, it is reduced to an appendix just to document its syntax and usage.

But if **LaTeX** itself has not changed substantially since 1999, many of its peripherals have. The rise of programs like pdf**TeX** and dvipdfm for PDF output adds new possibilities, which are realized, not in **LaTeX** directly, but by means of more modern packages to extend the basic features. The distribution of **TeX**/**LaTeX** installations has changed, such that most users are given a complete, ready-to-run setup, with all the ‘extras’ that one used to have to obtain oneself. Those extras include user-contributed packages, many of which are now considered indispensable. Today ‘the **LaTeX** system’ includes much more than the basic kernel by Leslie Lamport, encompassing the contributions of hundreds of other people. This edition reflects this increase in breadth.

The changes to the fourth edition are mainly those of emphasis.

1. The material has been reorganized into ‘Basics’ and ‘Beyond the Basics’ (‘advanced’ sounds too intimidating) while the appendices contain topics that really can be skipped by most everyday users. Exception: Appendix H is an alphabetized command summary that many people find extremely useful (including ourselves).

This reorganizing is meant to stress certain aspects over others. For
example, the section on graphics inclusion and color was originally treated as an exotic freak, relegated to an appendix on extensions; in the third edition, it moved up to be included in a front chapter along with the picture environment and floats; now it dominates Chapter 6 all on its own, the floats come in the following Chapter 7, and picture is banished to the later Chapter 13. This is not to say that the picture features are no good, but only that they are very specialized. We add descriptions of additional drawing possibilities there too.

2. It is stressed as much as possible that \LaTeX{} is a \textit{markup} language, with separation of content and form. Typographical settings should be placed in the preamble, while the body contains only logical markup. This is in keeping with the modern ideas of XML, where form and content are radically segregated.

3. Throughout this edition, contributed packages are explained at that point in the text where they are most relevant. The \texttt{fancyhdr} package comes in the section on page styles, \texttt{natbib} where literature citations are explained. This stresses that these ‘extensions’ are part of the \LaTeX{} system as a whole. However, to remind the user that they must still be explicitly loaded, a marginal note is placed at the start of their descriptions.

4. PDF output is taken for granted throughout the book, in addition to the classical DVI format. This means that the added possibilities of pdf\LaTeX{} and \texttt{dvipdfm} are explained where they are relevant. A separate Chapter 10 on PostScript and PDF is still necessary, and the best interface to PDF output, the \texttt{hyperref} package by Sebastian Rahtz, is explained in detail. PDF is also included in Chapter 15 on presentation material. On the other hand, the other Web output formats, HTML and XML, are only dealt with briefly in Appendix E, since these are large topics treated in other books, most noticeably the \LaTeX{} Web Companion.

5. This book is being distributed with the \TeX{}Live CD, with the kind permission of Sebastian Rahtz who maintains it for the \TeX{} Users Group. It contains a full \TeX{} and \LaTeX{} installation for Windows, Macintosh, and Linux, plus many of the myriad extensions that exist.

We once again express our hope that this \textit{Guide} will prove more than useful to all those who wish to find their way through the intricate world of \LaTeX{}. And with the addition of the \TeX{}Live CD, that world is brought even closer to their doorsteps.

\textit{Helmut Kopka and Patrick W. Daly}

\textit{June, 2003}
# Contents

## Preface

### Basics

1. Basics  

   1 Introduction  
   1.1 Just what is \LaTeX{}?  
   1.2 Markup Languages  
   1.3 \TeX{} and its offspring  
   1.4 How to use this book  
   1.5 Basics of a \LaTeX{} file  
   1.6 \TeX{} processing procedure

2. Text, Symbols, and Commands  

   2.1 Command names and arguments  
   2.2 Environments  
   2.3 Declarations  
   2.4 Lengths  
   2.5 Special characters  
   2.6 Exercises  
   2.7 Fine-tuning text  
   2.8 Word division

3. Document Layout and Organization  

   3.1 Document class  
   3.2 Page style  
   3.3 Parts of the document  
   3.4 Table of contents

4. Displayed Text  

   4.1 Changing font  
   4.2 Centering and indenting  
   4.3 Lists  
   4.4 Generalized lists  
   4.5 Theorem-like declarations
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.6 Tabulator stops</td>
<td>81</td>
</tr>
<tr>
<td>4.7 Boxes</td>
<td>85</td>
</tr>
<tr>
<td>4.8 Tables</td>
<td>95</td>
</tr>
<tr>
<td>4.9 Printing literal text</td>
<td>110</td>
</tr>
<tr>
<td>4.10 Footnotes and marginal notes</td>
<td>112</td>
</tr>
<tr>
<td>4.11 Comments within text</td>
<td>118</td>
</tr>
<tr>
<td><strong>5 Mathematical Formulas</strong></td>
<td>119</td>
</tr>
<tr>
<td>5.1 Mathematical environments</td>
<td>119</td>
</tr>
<tr>
<td>5.2 Main elements of math mode</td>
<td>120</td>
</tr>
<tr>
<td>5.3 Mathematical symbols</td>
<td>124</td>
</tr>
<tr>
<td>5.4 Additional elements</td>
<td>130</td>
</tr>
<tr>
<td>5.5 Fine-tuning mathematics</td>
<td>145</td>
</tr>
<tr>
<td>5.6 Beyond standard \LaTeX\</td>
<td>151</td>
</tr>
<tr>
<td><strong>6 Graphics Inclusion and Color</strong></td>
<td>153</td>
</tr>
<tr>
<td>6.1 The \texttt{graphics} packages</td>
<td>153</td>
</tr>
<tr>
<td>6.2 Adding color</td>
<td>166</td>
</tr>
<tr>
<td><strong>7 Floating tables and figures</strong></td>
<td>169</td>
</tr>
<tr>
<td>7.1 Float placement</td>
<td>169</td>
</tr>
<tr>
<td>7.2 Postponing floats</td>
<td>171</td>
</tr>
<tr>
<td>7.3 Style parameters for floats</td>
<td>171</td>
</tr>
<tr>
<td>7.4 Float captions</td>
<td>173</td>
</tr>
<tr>
<td>7.5 Float examples</td>
<td>174</td>
</tr>
<tr>
<td>7.6 References to figures and tables in text</td>
<td>177</td>
</tr>
<tr>
<td>7.7 Some float packages</td>
<td>178</td>
</tr>
<tr>
<td><strong>8 User Customizations</strong></td>
<td>181</td>
</tr>
<tr>
<td>8.1 Counters</td>
<td>181</td>
</tr>
<tr>
<td>8.2 Lengths</td>
<td>184</td>
</tr>
<tr>
<td>8.3 User-defined commands</td>
<td>185</td>
</tr>
<tr>
<td>8.4 User-defined environments</td>
<td>195</td>
</tr>
<tr>
<td>8.5 Some comments on user-defined structures</td>
<td>200</td>
</tr>
<tr>
<td><strong>II Beyond the Basics</strong></td>
<td>205</td>
</tr>
<tr>
<td><strong>9 Document Management</strong></td>
<td>207</td>
</tr>
<tr>
<td>9.1 Processing parts of a document</td>
<td>207</td>
</tr>
<tr>
<td>9.2 In-text references</td>
<td>213</td>
</tr>
<tr>
<td>9.3 Bibliographies</td>
<td>216</td>
</tr>
<tr>
<td>9.4 Keyword index</td>
<td>225</td>
</tr>
</tbody>
</table>
10 PostScript and PDF
10.1 \LaTeX{} and PostScript ........................................ 231
10.2 Portable Document Format ....................................... 236

11 Multilingual \LaTeX{}
11.1 The babel system ............................................... 252
11.2 Contents of the language.dat file ........................... 256

12 Math Extensions with $\mathcal{A}_M$-$\LaTeX$
12.1 Invoking $\mathcal{A}_M$-$\LaTeX$ .................................. 258
12.2 Standard features of $\mathcal{A}_M$-$\LaTeX$ .................... 258
12.3 Further $\mathcal{A}_M$-$\LaTeX$ packages ......................... 280
12.4 The $\mathcal{A}_M$ fonts ............................................. 283

13 Drawing with \LaTeX{}
13.1 The picture environment ..................................... 287
13.2 Extended pictures .............................................. 302
13.3 Other drawing packages ....................................... 307

14 Bibliographic Databases and Bib$\LaTeX$
14.1 The Bib$\LaTeX$ program ...................................... 309
14.2 Creating a bibliographic database ......................... 311
14.3 Customizing bibliography styles ........................... 321

15 Presentation Material
15.1 Slide production with Sl$\LaTeX$ ............................... 324
15.2 Slide production with seminar ................................ 330
15.3 Electronic documents for screen viewing ................... 340
15.4 Special effects with PDF .................................... 343

16 Letters
16.1 The $\LaTeX$ letter class ..................................... 351
16.2 A house letter style .......................................... 356
16.3 A model letter customization ............................... 359

Appendices

A The New Font Selection Scheme (NFSS)
A.1 Font attributes under NFSS ..................................... 368
A.2 Simplified font selection ...................................... 370
A.3 Installing fonts with NFSS .................................... 372
# CONTENTS

## B The \LaTeX{} Clockwork

B.1 Installing \LaTeX{} ........................................... 381  
B.2 Obtaining the Adobe euro fonts ............................... 387  
B.3 \TeX{} directory structure .................................. 387  
B.4 The CTAN server ............................................. 389  
B.5 Additional standard files .................................... 391  
B.6 The various \LaTeX{} files ................................... 396

## C Error Messages

C.1 Basic structure of error messages ............................. 401  
C.2 Some sample errors .......................................... 409  
C.3 List of \LaTeX{} error messages ............................... 415  
C.4 \TeX{} error messages ...................................... 424  
C.5 Warnings ......................................................... 429  
C.6 Search for subtle errors ...................................... 435

## D \LaTeX{} Programming

D.1 Class and package files ....................................... 437  
D.2 \LaTeX{} programming commands ............................... 440  
D.3 Sample packages .............................................. 451  
D.4 Changing preprogrammed text ................................ 459  
D.5 Direct typing of special letters ............................. 461  
D.6 Alternatives for special symbols ............................. 462  
D.7 Managing code and documentation ............................ 462

## E \LaTeX{} and World Wide Web

E.1 Converting to HTML ............................................ 475  
E.2 The Extensible Markup Language: XML ......................... 478  
E.3 The techexplorer Hypermedia Browser ......................... 481

## F Obsolete \LaTeX{}

F.1 The 2.09 preamble .............................................. 483  
F.2 Font selection .................................................. 484  
F.3 Obsolete means obsolete ...................................... 485

## G \TeX{} Fonts

G.1 Font metrics and bitmaps ...................................... 487  
G.2 Computer Modern fonts ........................................ 488  
G.3 The \Metafont{} program ...................................... 497  
G.4 Extended Computer fonts ..................................... 498  
G.5 PostScript fonts ............................................... 503  
G.6 Computer Modern as PostScript fonts ......................... 505
## H Command Summary

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H.1</td>
<td>Brief description of the \LaTeX commands</td>
</tr>
<tr>
<td>H.2</td>
<td>Summary tables and figures</td>
</tr>
</tbody>
</table>

## Bibliography

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>605</td>
</tr>
</tbody>
</table>

## Index

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>607</td>
</tr>
</tbody>
</table>

## List of Tables

10.1 The psnfss packages and their fonts | 234 |
10.2 Acrobat menu actions | 248 |

A.1 The NFSS encoding schemes | 368 |
A.2 The NFSS \textit{series} attributes | 369 |
A.3 Attributes of the Computer Modern fonts | 370 |

D.1 Input coding schemes for \texttt{inputenc} package | 462 |
D.2 Alternative commands for special symbols | 463 |

G.1 Computer Modern text fonts | 491 |
G.2 Root names of the 35 standard PostScript fonts | 504 |
G.3 Encoding suffixes | 505 |

H.1 Font attribute commands | 595 |
H.2 Math alphabet commands | 595 |
H.3 Font sizes | 595 |
H.4 \LaTeX 2.09 font declarations | 595 |
H.5 Dimensions | 596 |
H.6 Accents | 596 |
H.7 Special letters from other languages | 596 |
H.8 Special symbols | 596 |
H.9 Command symbols | 596 |
H.10 Greek letters | 596 |
H.11 Binary operation symbols | 597 |
H.12 Relational symbols | 597 |
H.13 Negated relational symbols | 597 |
H.14 Brackets | 597 |
H.15 Arrows | 598 |
H.16 Miscellaneous symbols | 598 |
H.17 Mathematical symbols in two sizes | 598 |
H.18 Function names | 598 |
H.19 Math accents | 599 |
H.20 $\mathcal{M}$ arrows | 599 |
H.21 $\mathcal{M}$ binary operation symbols | 599 |
List of Figures

1.1 Sample display with the WinEdt editor . . . . . . . . . . . . 16
3.1 Page layout parameters . . . . . . . . . . . . . . . . . . . . . 48
3.2 Sample title page . . . . . . . . . . . . . . . . . . . . . . . . . 53
4.1 The list parameters . . . . . . . . . . . . . . . . . . . . . . . . 76
6.1 An embellished image file . . . . . . . . . . . . . . . . . . . . 160
10.1 Output produced by pdf\TeX with the hyperref package . . 240
13.1 Comparison of eepic with eepicemu . . . . . . . . . . . . . . 306
15.2 Title page of a pdfscreen document . . . . . . . . . . . . . . 341
B.1 The \TeXLive welcome . . . . . . . . . . . . . . . . . . . . . . 383
B.2 The \TeXLive documentation browser . . . . . . . . . . . . . 384
B.3 The TDS directory tree . . . . . . . . . . . . . . . . . . . . . . 388
B.4 Partial directory tree of CTAN servers . . . . . . . . . . . . . 390
E.1 Example of \TeX4ht and techexplorer output . . . . . . . . . 482
H.1 Single column page format . . . . . . . . . . . . . . . . . . . . 602
H.2 Double column page format . . . . . . . . . . . . . . . . . . . 603
H.3 Format of the list environment . . . . . . . . . . . . . . . . . 604
Part I

Basics
1

Introduction

1.1 Just what is \LaTeX?  

To summarize very briefly:

- \LaTeX{} is a comprehensive set of markup commands used with the powerful typesetting program \TeX{} for the preparation of a wide variety of documents, from scientific articles, reports, to complex books.

- \LaTeX{} is an open software system, available free of charge. Its core is maintained by the \LaTeX{}3 Project Group but it also benefits from extensions written by hundreds of user/contributors, with all the advantages and disadvantages of such a democracy.

- A \LaTeX{} document consists of one or more source files containing plain text characters, the actual textual content plus markup commands. These include instructions which can insert graphical material produced by other programs.

- It is processed by the \TeX{} program to produce a binary file in DVI (device independent) format, containing precise directions for the typesetting of each character. This in turn can be viewed on a monitor, or converted into printer instructions, or some other electronic form such as PostScript, HTML, XML, or PDF.

- A variant on the \TeX{} program called pdf\TeX{} produces PDF output directly from the source file without going through the DVI intermediary. With this, \LaTeX{} can automatically include internal links and bookmarks with little or no extra effort, plus PDF buttons and external links, in addition to graphics in a wide range of common formats.

- \TeX{} activities are coordinated by the \TeX{} Users Group, TUG (www.tug.org) who distribute a set of CDs, called \TeX{}Live, annually to its
members, containing a $\LaTeX$ installation for various computer types.

The rest of this book attempts to fill in the gaps in the above summary. With the help of the included $\TeX$Live CD for Windows, Macintosh, and Linux, which also contains a directory specific to this book (\books\Kopka_and_Daly\), we hope that the user will have additional pleasure in learning the joys of $\LaTeX$.

### 1.2 Markup Languages

#### 1.2.1 Typographical markup

In the days before computers, an author would prepare a *manuscript* either by hand or by typewriter, which he or she would submit to a publisher. Once accepted for publication (and after several rounds of corrections and modifications, each requiring a rewrite of the paper manuscript), it would be sent to a copy editor, a human being who would decorate the manuscript with *markup*, marginal notes that inform the typesetter (another human being) which fonts and spacings and other typographical features should be used to convert it to the final printed form that one expects of books and articles.

Electronic processing of text today follows a similar procedure, except that the humans have been replaced by computer programs. (So far the author has avoided this fate, but they are working on it.) The markup is normally included directly in the manuscript in such a way that it is converted immediately to its output form and displayed on the computer monitor. This is known as WYSIWYG, or ‘what you see is what you get’.

However, what you see is not always what you’ve got. An alternative that is used more and more by major publishers is *markup languages*, in which the raw text is interspersed with indicators ‘for the typesetter.’ The result as seen on the monitor is much the same as a typewritten manuscript, except that the markup is no longer abbreviated marginal notes, but cryptic code within the actual text. This *source text*, which can be prepared by a simple, dumb *text editor* program, is converted into typographically set output by a separate program.

For example, to code the line

He took a **bold step** forward.

with HTML, the classical markup language of the World Wide Web, one enters in the source text:

He took a `<b>bold step</b>` forward.

In Plain $\TeX$, the same sentence would be coded as:
He took a \{\bf bold step\} forward.

The first example is to be processed (displayed) by a Web browser program that decides to set everything between \(<b>\) and \(</b>\) as bold face. The second example is intended for the \TeX{} program (Section 1.3). The markup in these two examples follow different rules, different syntax, but the functionality is the same.

1.2.2 Logical markup

The above examples illustrate typographical markup, where the inserted commands or tags give direct instructions to alter the appearance of the output, here a change of font. An alternative is to indicate the purpose of the text. For example, HTML recognizes several levels of headings; to place a title into the highest level one enters:

\begin{verbatim}
<h1>Logical Markup</h1>
\end{verbatim}

The equivalent \LaTeX{} entry would be:

\begin{verbatim}
\section{Logical Markup}
\end{verbatim}

With this logical markup, the author concentrates entirely on the content and leaves the typographical considerations to the experts. One merely marks the structure of the document, and has no means of controlling how the logical elements, like section titles, are to be rendered typographically. This information is put into HTML style sheets or \LaTeX{} classes and packages, which are external to the actual source file. This means that the entire layout of a document can be overhauled with only minimal or even no alterations to the source file.

Today much effort is being put into XML, the Extensible Markup Language, as the ultimate markup system, since it allows the markup, or tags, to be defined as needed, without any indication of how they are to be implemented. That is left to XSL, the Extensible Stylesheet Language. It must be emphasized that neither XML nor XSL are programs at all; they are specifications for how documents and databases may be marked up, and how the markup tags may be translated into real output. Programs still need to be found to do the actual job.

And that is the fundamental idea behind markup languages: that the source text indicates the logical structure of its contents. Such source files, being written in plain ASCII text, are extremely robust, not being married to any particular software package or computer type.

What does all this have to do with \LaTeX{}? In the next Section we outline the development of \TeX{} and \LaTeX{}, and go on to show that \LaTeX{}, a product of the mid 1980’s, is a programmable markup language that is ideally suited for the modern world of electronic publishing.
1.3 \LaTeX and its offspring

The most powerful formatting program for producing book quality text of scientific and technical works is that of Donald E. Knuth (Knuth, 1986a, 1986b, 1986c, 1986d, 1986e). The program is called \LaTeX, which is a rendering in capitals of the Greek letters \textsc{τεχ}. For this reason the last letter is pronounced not as an \textit{x}, but as the \textit{ch} in Scottish \textit{loch} or German \textit{ach}, or as the Spanish \textit{j} or Russian \textit{kh}. The name is meant to emphasize that the printing of mathematical texts is an integral part of the program and not a cumbersome add-on. In addition to \LaTeX, the same author has developed a further program called \textsc{MetaFont} for the production of character fonts. The standard \LaTeX program package contains 75 fonts in various design sizes, each of which is also available in up to eight magnification steps. All these fonts were produced with the program \textsc{MetaFont}. With additional applications, further character fonts have been created, such as for Cyrillic, Chinese, and Japanese, with which texts in these alphabets can be printed in book quality.

The \LaTeX program is free, and the source code is readily available. Anybody may take it and modify it as they like, provided they call the result something other than \LaTeX. This indeed has occurred, and several \LaTeX variants do exist, including pdf\LaTeX which we deal with later in this Chapter. Only Knuth is allowed to alter \LaTeX itself, which he does only to correct any obvious bugs. Otherwise, he considers \LaTeX to be completed; the current version number is 3.14159, and with his death, the code will be frozen for all time, and the version number will become exactly \pi.

1.3.1 The \LaTeX program

The basic \LaTeX program only understands a set of very primitive commands that are adequate for the simplest of typesetting operations and programming functions. However, it does allow more complex, higher-level commands to be defined in terms of the primitive ones. In this way, a more user-friendly environment can be constructed out of the low-level building blocks.

During a processing run, the program first reads in a so-called \textit{format file} which contains the definitions of the higher-level commands in terms of the primitive ones, and which also contains the hyphenation patterns for word division. Only then does it read in the author's \textit{source file} containing the actual text to be processed, including formatting commands that are predefined in the format file.

Creating new formats is something that should be left to very knowledgeable programmers. The definitions are written to a source file which is then processed with a special version of the \LaTeX program called \texttt{initex}. It stores the new format file in a compact manner so that it can be read in quickly by the regular \LaTeX program.
1.3. \TeX{} and its offspring

Although the normal user will almost never write such a format, he or she may be presented with a new format source file that will need to be installed with \texttt{initex}. For example, this is just what must be done to upgrade \LaTeX{} periodically. How to do this is described in Appendix B.

1.3.2 Plain \TeX{}

Knuth has provided a basic format named \textit{Plain \TeX{}} to interact with \TeX{} at its simplest level. This is such a fundamental part of \TeX{} processing that one tends to forget the distinction between the actual processing program \TeX{} and this particular format. Most people who claim to 'work only with \TeX{}' really mean that they only work with Plain \TeX{}.

Plain \TeX{} is also the basis of every other format, something that only reinforces the impression that \TeX{} and Plain \TeX{} are one and the same.

1.3.3 \LaTeX{}

The emphasis of Plain \TeX{} is still very much at the typesetter's level, rather than the author's. Furthermore, the exploitation of all its potential demands considerable experience with programming techniques. Its application thus remains the exclusive domain of typographic and programming professionals.

For this reason, the American computer scientist Leslie Lamport has developed the \LaTeX{} format (Lamport, 1985), which provides a set of higher-level commands for the production of complex documents. With it, even the user with no knowledge of typesetting or programming is in a position to take extensive advantage of the possibilities offered by \TeX{}, and to be able to produce a variety of text outputs in book quality within a few days, if not hours. This is especially true for the production of complex tables and mathematical formulas.

As pointed out in Section 1.2.2, \LaTeX{} is very much more a \textit{logical} markup language than the original Plain \TeX{}, on which it is based. It contains provisions for automatic running heads, sectioning, tables of contents, cross-referencing, equation numbering, citations, floating tables and figures, without the author having to know just how these are to be formatted. The layout information is stored in additional \textit{class files} which are referred to but not included in the input text. The predefined layouts may be accepted as they are, or replaced by others with minimal changes to the source file.

Since its introduction in the mid-1980s, \LaTeX{} has been periodically updated and revised, like all software products. For many years the version number was fixed at 2.09 and the revisions were only identified by their dates. The last major update occurred on December 1, 1991, with some minor corrections up to March 25, 1992, at which point \LaTeX{} 2.09 became frozen.
1.3.4 \LaTeX 2\varepsilon

The enormous popularity of \LaTeX and its expansion into fields for which it was not originally intended, together with improvements in computer technology, especially dealing with cheap but powerful laser printers, had created a diversity of formats bearing the \LaTeX label. In an effort to re-establish a genuine, improved standard, the \LaTeX3 Project was set up in 1989 by Leslie Lamport, Frank Mittelbach, Chris Rowley, and Rainer Schöpf. Their goal was to construct an optimized and efficient set of basic commands complemented by various packages to add specific functionality as needed.

As the name of the project implies, its aim is to achieve a version 3 for \LaTeX. However, since that is the long-term goal, a first step towards it was the release of \LaTeX2\varepsilon in mid-1994 together with the publication of the second edition of Lamport’s basic manual (Lamport, 1994) and of an additional book (Goossens et al., 1994) describing many of the extension packages available and \LaTeX programming in the new system. Since then, two further books have appeared, Goossens et al. (1997) dealing with the inclusion of graphics and color, and Goossens and Rahtz (1999) explaining how \LaTeX may be used with the World Wide Web. Both these topics are also dealt with in this Guide.

Initially updates to \LaTeX2\varepsilon were issued twice a year, in June and December, but it has now become so stable that since 2000 the changes are released only once a year, nominally in June.

\LaTeX2\varepsilon is now the standard version, and \LaTeX 2.09 is considered obsolete, although source files intended for the older version may still be processed with the newer one. In this book, unless otherwise indicated, ‘\LaTeX’ will always mean \LaTeX2\varepsilon.

1.3.5 \TeX fonts

\TeX initially made use of its own set of fonts, called Computer Modern generated by Knuth's \METAFONT program. The reason for doing this was that printers at that time (and even today) may contain their own preloaded fonts, but they are often slightly different from printer to printer. Furthermore, they lacked the mathematical character sets that are essential to \TeX’s main hallmark, mathematical typesetting. So Knuth created pixel fonts that could be sent to every printer ensuring the same results everywhere.

Today the situation with fonts has changed dramatically. Outline fonts (also known as type 1 fonts) are more compact and versatile than the pixel fonts (type 3). They also have a far superior appearance and are drawn much faster in PDF files. The original Computer Modern fonts have been converted to outline fonts, but there is no reason to stick with them, except possibly for the mathematical symbols. It is \LaTeX2\varepsilon with its New
Font Selection Scheme that freed \TeX from its rigid marriage to Computer Modern.

Fonts are discussed in more detail in Appendix G.

1.3.6 **The \LaTeX bazaar: user contributions**

Like the \TeX program on which it relies, \LaTeX is freeware. There may be a prejudice that what is free in not worth anything, but there are other examples in the computer world to contradict this statement. And since the \LaTeX macros are provided in files containing plain text, there is no problem to exchange, modify, and supplement them. In other words, the user can participate in extending the basic \LaTeX system.

Taking advantage of a mechanism in \LaTeX 2.09 that allowed options to the default layouts to be contained in so-called style option files, many users began writing their own 'options' to provide additional features to the basic \LaTeX. They then made these available to other users via the Internet. Many were intended for very specific problems, but many more proved to be of such general usefulness that they have become part of the standard \LaTeX installation. In this way, the users themselves have built up a system that meets their needs.

With \LaTeX\epsilon, these user contributions acquired official status: they became known as packages, they could be entered directly into the document and not by the back door, guidelines were issued for writing them, and additional commands were introduced to assist package programming. Package files bear the extension .sty from \LaTeX 2.09 days, so that the older style option files may still function as packages today.

Those packages that have established themselves as indispensable for sophisticated \LaTeX processing are described in this book in those sections where they are most relevant. This does not imply that other packages are less worthwhile, but simply that this book does have to make a selection. Many other packages are described fully in *The \LaTeX Companion* (Goossens et al., 1994) and it would go beyond the bounds of this book to reproduce it here.

1.3.7 **\LaTeX and electronic publishing**

The most significant development in computer usage in the last decade is the rise of the World Wide Web (or the hijacking of the Internet by the glitzy society). \LaTeX makes its own contribution here with

- programs to convert \LaTeX files to HTML (Appendix E);
- means of creating PDF output, with hypertext features such as links, bookmarks, active buttons (Chapter 10);
interfacing to XML both by acting as an engine to render XML documents and with programs to convert \LaTeX{} to XML and vice versa (Appendix E).

All these forms of electronic publishing are alternatives to traditional paper output. We do not expect paper to disappear entirely so quickly, but it is rapidly being replaced by electronic forms, which can always reproduce the paper whenever needed.

1.4 How to use this book

This Guide is meant to be a mixture of textbook and reference manual. It explains all the essential elements of the current standard \LaTeX{} 2ε, but compared to Lamport (1985, 1994), it goes into more detail, offers more examples and exercises, and describes many ‘tricks’ based on the authors’ experiences. It explains not only the core \LaTeX{} installation, but also many of the contributed packages that have become essential to modern \LaTeX{} processing, and thus quasi-standard. We necessarily have to be selective, for we cannot go to the same extend as The \LaTeX{} Companion (Goossens et al., 1994), The \LaTeX{} Graphics Companion (Goossens et al., 1997), and The \LaTeX{} Web Companion (Goossens and Rahtz, 1999), which are still valid companions to this book.

The first part of the book is entitled The Basics, and deals with the more fundamental aspects of \LaTeX{}: inputting text and symbols, document organization, lists and tables, entering mathematics, and customizations by the user. The second part is called Beyond the Basics, meaning it presents concepts which may be more advanced but which are still essential to producing complex, sophisticated documents. The distinction is rather arbitrary. Finally, the appendices contain topics that are not directly part of \LaTeX{} itself, but useful for understanding its applications: installation, error messages, creating packages, World Wide Web, fonts. Appendix H is an alphabetized summary of most of the commands and their use, cross-referenced to their locations in the main text.

1.4.1 Some conventions

In the description of command syntax, typewriter type is used to indicate those parts that must be entered exactly as given, while italic is reserved for those parts that are variable or for the text itself. For example, the command to produce tables is presented as follows:

\begin{tabular}{col_form} lines \end{tabular}

The parts in typewriter type are obligatory, while col_form stands for the definition of the column format that must be inserted here. The allowed
values and their combinations are given in the detailed descriptions of the commands. In the above example, _lines_ stands for the line entries in the table and are thus part of the text itself.

Sections describing a package, an extension to basic _LaTeX_, have the name of that package printed as a marginal note, as demonstrated here for this paragraph. In this way, you are reminded that you must include it with \texttt{\usepackage} (Section 3.1.2) in order to obtain the additional features. Without it, you are likely to get an error message about undefined commands.

Sections of text that are printed in a smaller typeface together with the boxed exclamation mark at the left are meant as an extension to the basic description. They may be skipped over on a first reading. This information presents deeper insight into the workings of _LaTeX_ than is necessary for everyday usage, but which is invaluable for creating more refined control over the output.

### 1.5 Basics of a _LaTeX_ file

#### 1.5.1 Text and commands

The _source file_ for _LaTeX_ processing, or simply the _LaTeX_ file, contains the _source text_ that is to be processed to produce the printed output. Splitting the text up into lines of equal width, formatting it into _paragraphs_, and breaking it into _pages_ with page numbers and running heads are all functions of the processing program and not of the input text itself.

For example, words in the source text are strings of letters terminated by some non-letter, such as _punctuation_, _blanks_, or _end-of-lines_ (_hard_ end-of-lines, ones that are really there, not the _soft_ ones that move with the window width); whereas punctuation marks will be transferred to the output, blanks and end-of-lines merely indicate a gap between words. Multiple blanks in the input, or blanks at the beginning of a line, have no effect on the interword spacing in the output.

Similarly, a new paragraph is indicated in the input text by an empty line; multiple empty lines have the same effect as a single one. In the output, the paragraph may be formatted either by indentation of the first line, or by extra interline spacing, but this is not affected in any way by the number of blank lines or extra spaces in the input.

The source file contains more than just text, however; it is also interspersed with markup commands that control the formatting or indicate the structure. It is therefore necessary for the author to be able to recognize what is text and what is a command. Commands consist either of certain single characters that cannot be used as text characters, or of words preceded immediately by a special character, the backslash (\). The syntax of source text is explained in detail in Chapter 2.
Chapter 1. Introduction

1.5.2 Contents of a \texttt{\LaTeX} source file

Every \LaTeX\ file contains a \textit{preamble} and a \textit{body}.

The preamble is a collection of commands that specify the global processing parameters for the following text, such as the paper format, the height and width of the text, the form of the output page with its pagination and automatic page heads and footlines. As a minimum, the preamble must contain the command $\texttt{\documentclass}$ to specify the document’s overall processing type. This is the first command in the preamble.

If there are no other commands in the preamble, \LaTeX\ selects standard values for the line width, margins, paragraph spacing, page height and width, and much more. By default, these specifications are tailored to the American norms. For European requirements, built-in options exist to alter the text height and width to the A4 standard. Furthermore, there are language-specific packages to translate certain headings such as ‘Chapter’ and ‘Abstract’.

The preamble ends with $\texttt{\begin{document}}$. Everything that follows this command is interpreted as \textit{body}. It consists of the actual text mixed with markup commands. In contrast to those in the preamble, these commands have only a local effect, meaning they apply only to a part of the text, such as indentation, equations, temporary change of font, and so on. The body ends with the command $\texttt{\end{document}}$. This is normally the end of the file as well.

The general syntax of a \LaTeX\ file is as follows:

\begin{verbatim}
\documentclass[options]{class}
  Further global commands and specifications
\begin{document}
  Text mixed with additional commands of local effect
\end{document}
\end{verbatim}

The possible \textit{options} and \textit{classes} that may appear in the $\texttt{\documentclass}$ command are presented in Section 3.1.1.

A minimal \LaTeX\ file named \texttt{hi.tex} contains just the following lines:

\begin{verbatim}
\documentclass{article}
\begin{document}
  Hi!
\end{document}
\end{verbatim}

1.5.3 Extending \LaTeX\ with packages

Packages are a very important feature of \LaTeX\. These are extensions to the basic \LaTeX\ commands that are written to files with names that end in \texttt{.sty} and are loaded with the command $\texttt{\usepackage}$ in the preamble. Packages can be classified by their origin:
1.5. Basics of a \LaTeX{} file

\textbf{core} packages are an integral part of the \LaTeX{} basic installation and are therefore fully standard;

\textbf{tools} packages are a set written by members of the \LaTeXX{} Team, and should always be in the installation;

\textbf{graphics} packages are a standardized set for including pictures generated by other programs, and for handling color; they are on the same level as the tools packages;

\textbf{AMS-\LaTeX{}} packages published by the American Mathematical Society, should be in any installation;

\textbf{contributed} packages have been submitted by actual users; certain of these have established themselves as 'essential' to standard \LaTeX{} usage, but all are useful.

Only a limited number of these packages are described in this book, those that we consider indispensable. However, there is nothing to prevent the user from obtaining and incorporating any others that should prove beneficial for his or her purposes.

There are over 1000 contributed packages on the included \TeX{}Live CD. How can one begin to get an overview of what they offer? Graham Williams has compiled a list of brief descriptions which can be found online and on the \TeX{}Live CD at

```
\texttt{\textbackslash texmf\textbackslash doc\textbackslash html\textbackslash catalogue\textbackslash catalogue.html}
```

How to load packages into the \LaTeX{} source file is explained in Section 3.1.2.

Documentation of contributed packages is somewhat haphazard, depending on how much the author has put into it. The preferred method for distributing packages is to integrate the documentation with the code into a single file with extension \texttt{.dtx}. A special program \texttt{DocStrip} (Section D.7.1) is used to extract the actual package file or files, while \LaTeX{}ing the original \texttt{.dtx} file produces the instruction manual. Most ready-to-run installations will already have done all this for the user, with the resulting manuals stored as DVI or PDF files somewhere in \texttt{\textbackslash texmf\textbackslash doc\textbackslash latex\ldots}. However, you might have to generate the documentation output yourself by processing the \texttt{.dtx} file, which should be found in \texttt{\textbackslash texmf\textbackslash source\textbackslash latex\ldots}. (Section B.3 explains the organization of the \TeX{} directory system.)

Some package authors write their manuals as an extra \texttt{.tex} file, the output of which may or may not be prestored in DVI or PDF form. Others provide HTML files. And still others simply add the instructions as comments in the package file itself. (This illustrates some of the joys of an open system.)
1.6 \TeX{} processing procedure

Since \LaTeX{} is a set of definitions for the \TeX{} program, \LaTeX{} processing itself is in fact \TeX{} processing with the \LaTeX{} format. What \TeX{} does with this is the same as for any other of the many formats available (of which \LaTeX{} is perhaps the most popular). All the typesetting work is done by \TeX{}, while \LaTeX{} handles the conversion from the logical markup to the typesetting commands. It also enables cross-referencing, running headlines, table of contents, literature citations and bibliography, indexing, and more. However, the processing of the source file to final output is \TeX{}'s task, regardless of the format being used.

1.6.1 In the good old days

\TeX{} arose over 20 years ago before there were such things as PCs, graphical displays, and before computers were infected with windows or mice. \TeX{} and its support programs were invoked from a command line, not with a mouse click. This may sound very old fashioned, but it did guarantee portability to all computer types.

The processing steps that were taken in those days still exist with today's graphical interfaces, but are now executed more conveniently. One can still open a 'command prompt window' and run them from the command line.

The first step is of course to use a text editor program to write the source file containing the actual text and markup. The rules for entering this source text are explained in Chapter 2. It goes into a text file, or what is often called an 'ASCII' file containing only standard punctuation marks, numbers, unaccented letters, upper and lower case. In other words, the text is that which can be produced from a standard English typewriter.

The name of the source file normally has the extension .tex; it is then processed by \TeX{} to produce a new file with the same base name and the extension .dvi, for device independent file. This is a binary file (all codes possible, not a text file) containing precise instructions for the selection and placement of every symbol, a coded description of the final printed page. The command to invoke \TeX{} with the source file hi.tex is

\texttt{tex \&latex hi}

meaning run the \TeX{} program with the format \latex{}. Usually the installation has defined a shortcut named \latex{} to do this, so

\texttt{latex hi}

should be sufficient. It is only necessary to specify the extension of the source file name if it is something other than .tex.

During the processing, \TeX{} writes information, warnings, error messages to the computer monitor, and to a transcript file with the extension .log. It is well worth inspecting this file when unexpected results appear.
The final step is to produce the printed pages from the DVI file. This requires another program, a *driver*, to generate the instructions specific to the given printer. For example, to produce a PostScript file, one runs

```
dvips hi
```
to obtain `hi.ps` from `hi.dvi`. And then one sends `hi.ps` to the PostScript printer with the regular command for that computer system.

Previewing the DVI file on a computer monitor before printing was a later development, requiring high quality graphics displays. These programs are essentially special drivers that send the output directly to the monitor rather than to a printer or printer file. One very popular previewer is called with

```
xdvi hi
```
to view `hi.dvi` before committing it to paper.

### 1.6.2 And today

The various steps for LaTeX processing described above are still necessary today, and one can open up a command prompt window and carry them out just as before. However, there now exist intelligent editors with LaTeX-savvy that not only assist writing the source text, but also will call the various programs, TeX, previewer, printer driver, BIBTeX, MakeIndex (these are explained later) with a mouse click.

One such editor for Windows, available on the enclosed TeXLive CD in the `support` directory, is called *WinShell*, written by Ingo H. de Boer ([www.winshell.de](http://www.winshell.de)). Although free of charge, its author appreciates donations to offset his expenses.

Another such editor and LaTeX interface is *WinEdit* by Aleksander Simonic ([www.winedt.com](http://www.winedt.com)). A sample window with the opening text of this chapter is shown in Figure 1.1. This program is available for a 30-day trial period, after which one must pay a nominal fee to obtain a licence. It is the editor that we ourselves use and we can highly recommend it.

An alternative is LyX, a free, open source software for document processing in near WYSIWYG, acting as a front-end to LaTeX, where the user need not know anything about LaTeX. See its home page at [www.lyx.org](http://www.lyx.org).

It must be stressed that all the above are *interfaces* to an existing LaTeX installation. On the other hand, there are also commercial packages which include both the TeX/LaTeX installation and a graphics interface. These are listed in Section B.1.1.

### 1.6.3 Alternative to TeX: pdfTeX

As we mentioned earlier, it is permitted to use the TeX source code to generate something else, as long as it bears another name. One such
Figure 1.1: Sample display with the WinEdt editor for interfacing to \LaTeX.

modification is called pdf\TeX, created by H\'an Th\’anh. This program
does everything \TeX\ does, but it optionally writes its output directly to a
PDF file, bypassing the DVI output of regular \TeX. It therefore combines
the \TeX\ program with a DVI-to-PDF driver program. Normally this option
is also the default.

There are many advantages to producing PDF output directly this way,
 apart from saving a step. The PDF file is generated in exactly the same way
as the DVI file with \TeX, and can be viewed immediately with the Acrobat
Reader or other PDF viewer. The results can be sent directly to a printer
without going through the DVI-to-Printer program. It is also much easier
to include the hypertext features of a true active PDF file, as we explain in
Section 10.2.4.

Adding the \LaTeX \ macros to pdf\TeX produces something one could call
pdf\LaTeX. This distinction is only meaningful for invoking the program-
plus-format to process the \LaTeX \ source file. Except for some things that
we note in Section 10.2.3, \LaTeX \ commands are identical whether used with
\TeX \ or with pdf\TeX. This makes the conversion extremely easy.

The rest of this book deals essentially with \LaTeX itself, regardless of
what the end product is to be: paper, HTML, XML, or PDF.
Text, Symbols, and Commands

The text that is to be the input to a \LaTeX processing run is written to a source file with a name ending in .tex, the file name extension. This file is prepared with a text editor, either one that handles straightforward plain text, or one that is configured to assist the writing and processing of \LaTeX files. In either case, the contents of this file are plain ASCII characters only, with no special symbols, no accented letters, preferably displayed in a fixed width typewriter font, with no frills like bold or italics, all in one size. All these aspects of true typesetting are produced afterwards by the \TeX processing program with the help of markup commands inserted visibly into the actual text. It is therefore vital to know how commands are distinguished from text that is to be printed, and, of course, how they function.

(However, for languages other than English, native keyboard input may indeed be used, as shown in Section 2.5.9.)

2.1 Command names and arguments

A command is an instruction to \LaTeX to do something special, like print some symbol or text not available to the restricted character set used in the input file, or to change the current typeface or other formatting properties. There are three types of command names:

- the single characters # $ & \_ \^ % \{ \} all have special meanings that are explained later in this chapter;

- the backslash character \ plus a single non-letter character; for example \$ to print the $ sign; all the special characters listed above have a corresponding two-character command to print them literally;

- the backslash character \ plus a sequence of letters, ending with the first non-letter; for example, \large to switch to a larger typeface.
Command names are case sensitive, so `\large`, `\Large` and `\LARGE` are distinct commands.

Many commands operate on some short piece of text, which then appears as an argument in curly braces following the command name. For example, `\emph{stress}` is given to print the word `stress` in an emphasized typeface (here italic) as `stress`. Such arguments are said to be mandatory because they must always be given.

Some commands take optional arguments, which are normally employed to modify the effects of the command somehow. The optional arguments appear in square braces.

In this book we present the general syntax of commands as

```latex
\textit{name}[\textit{optional}]\{\textit{mandatory}\}
```

where typewriter characters must be typed exactly as illustrated and italic text indicates something that must be substituted for. Optional arguments are put into square brackets `[ ]` and the mandatory ones into curly braces `{ }`. A command may have several optional arguments, each one in its set of brackets in the specified sequence. If none of the optional arguments is used, the square brackets may be omitted. Any number of blanks, or even a single new line, may appear between the command name and the arguments, to improve legibility.

Some commands have several mandatory arguments. Each one must be put into a `{ }` pair and their sequence must be maintained as given in the command description. For example,

```latex
\texttt{\textbackslash rule}[\textit{lift}]\{\textit{width}\}\{\textit{height}\}
```

produces a black rectangle of size `width` and `height`, raised by an amount `lift` above the current baseline. A rectangle of width 10 mm and height 3 mm is made with `\rule{10mm}{3mm}`. Since the optional argument `lift` is omitted, the rectangle is set on the baseline with no lifting, as `rectangle`. The arguments must appear in the order specified by the syntax and may not be interchanged.

Some commands have a so-called *-form in addition to their normal appearance. A * is added to their name to modify their functionality somehow. For example, the `\section` command has a *-form `\section*` which, unlike the regular form, does not print an automatic section number. For each such command, the difference between the normal and *-form will be explained in the description of the individual commands.

Command names consist only of letters, with the first non-letter indicating the end of the name. If there are optional or mandatory arguments following the command name, then it ends before the [ or { bracket, since these characters are not letters. Many commands, however, possess no arguments and are composed of only a name, such as the command `\LaTeX` to produce the \LaTeX\ logo. If such a command is followed by
a punctuation mark, such as comma or period, it is obvious where the command ends. *If it is followed by a normal word, the blank between the command name and the next word is interpreted as the command terminator:* The \LaTeX{} logo results in ‘The \LaTeX{} logo’, that is, the blank was seen only as the end of the command and not as spacing between two words. This is a result of the special rules for blanks, described in Section 2.5.1.

In order to insert a space after a command that consists only of a name, either an empty structure {} or a space command (\ and blank) must be placed after the command. The proper way to produce ‘The \LaTeX{} logo’ is to type either The \LaTeX{} logo or The \LaTeX\ logo. Alternatively, the command itself may be put into curly braces, as The \{\TeX\} logo, which also yields the right output with the inserted blank: ‘The \TeX{} logo’. Incidentally, the \LaTeX{}2e logo is produced with \LaTeX{}e. Can you see why this logo command cannot be named \LaTeX{}2e?

### 2.2 Environments

An *environment* is initiated with the command \texttt{\begin{name}} and is terminated by \texttt{\end{name}}.

An environment has the effect that the text within it is treated differently according to the environment parameters. It is possible to alter (temporarily) certain processing features, such as indentation, line width, typeface, and much more. The changes apply only within the environment. For example, with the \texttt{quote} environment,

\begin{quote}
\texttt{previous text}
\texttt{\begin{quote}}
\texttt{text1 \small text2 \bfseries text3}
\texttt{\end{quote}}
\texttt{following text}
\end{quote}

the left and right margins are increased relative to those of the previous and following texts. In the example, this applies to the three texts \texttt{text1}, \texttt{text2}, and \texttt{text3}. After \texttt{text1} comes the command \texttt{\small}, which has the effect of setting the next text in a smaller typeface. After \texttt{text2}, there is an additional command \texttt{\bfseries} to switch to bold face type. Both these commands only remain in effect up to the \texttt{\end{quote}}.

The three texts within the \texttt{quote} environment are indented on both sides relative to the previous and following texts. The \texttt{text1} appears in the normal typeface, the same one as outside the environment. The \texttt{text2} and \texttt{text3} appear in a smaller typeface, and \texttt{text3} furthermore appears in bold face.

After the end of the \texttt{quote} environment, the subsequent text appears in the same typeface that was in effect beforehand.
Note that if the names of the environment in the \begin{...} \end{...} pair do not match, an error message will be issued on processing.

Most declaration command names (see next section) may also be used as environment names. In this case the command name is used without the preceding \ character. For example, the command \texttt{\texttt{\em}} switches to an emphatic typeface, usually italic, and the corresponding environment \begin{em} will set all the text in italic until \end{em} is reached.

A nameless environment can be simulated by a {...} pair. The effect of any command within it ends with the closing curly brace.

The user can even create his or her own environments, as described in Section 8.4.

2.3 Declarations

A declaration is a command that changes the values or meanings of certain parameters or commands without printing any text. The effect of the declaration begins immediately and ends when another declaration of the same type is encountered. However, if the declaration occurs within an environment or a {...} pair, its scope extends only to the corresponding \end command, or to the closing brace \}. The commands \texttt{\bfseries} and \texttt{\small} mentioned in the previous section are examples of such non-printing declarations that alter the current typeface.

Some declarations have associated arguments, such as the command \texttt{\texttt{\setlength}} which assigns a value to a length parameter (see Sections 2.4 and 8.2).

Examples:

\begin{itemize}
  \item \texttt{\bfseries This text appears in bold face} The \texttt{\bfseries} declaration changes the typeface: \textit{This text appears in bold face}. The effect of this declaration ends with the closing brace \}.
  \item \texttt{\setlength{\parindent}{0.5cm}} The paragraph indentation is set to 0.5 cm. The effect of this declaration ends with the next encounter of the command \texttt{\setlength{\parindent}}, or at the latest with the \texttt{\end} command that terminates the current environment.
  \item \texttt{\pagernumbering{roman}} The page numbering is to be printed in Roman numerals.
\end{itemize}

Some declarations, such as the last example, are global, that is, their effects are not limited to the current environment. The following declarations are of this nature, the meanings of which are given later:

\begin{itemize}
  \item \texttt{\newcounter}
  \item \texttt{\setcounter}
  \item \texttt{\addtocounter}
  \item \texttt{\pagernumbering}
  \item \texttt{\newlength}
  \item \texttt{\thispagestyle}
  \item \texttt{\newsavebox}
\end{itemize}
Declarations made with these commands are effective right away and remain so until they are overridden by a new declaration of the same type. In the last example above, page numbering will be done in Roman numerals until countermanded by a new \pagenumbering{arabic} command.

2.4 Lengths

2.4.1 Fixed lengths

*Lengths* consist of a decimal number with a possible sign in front (+ or -) followed by a mandatory dimensional unit. Permissible units and their abbreviated names are:

- cm centimeter,
- mm millimeter,
- in inch (1 in = 2.54 cm),
- pt point (1 in = 72.27 pt),
- bp big point (1 in = 72 bp),
- pc pica (1 pc = 12 pt),
- dd didot point (1 157 dd = 1 238 pt),
- cc cicero (1 cc = 12 dd),
- em a font-specific size, the width of the capital M,
- ex another font-related size, the height of the letter x.

Decimal numbers in TeX and LaTeX may be written in either the English or European manner, with a *period* or a *comma*: both 12.5cm and 12,5cm are permitted.

Note that 0 is not a legitimate length since the unit specification is missing. To give a zero length it is necessary to add some unit, such as 0pt or 0cm.

Values are assigned to a length parameter by means of the LaTeX command \setlength, which is described in Section 8.2 along with other commands for dealing with lengths. Its syntax is:

\setlength{\length_name}{\length_spec}

For example, the width of a line of text is specified by the parameter \textwidth, which is normally set to a default value depending on the class, paper type, and font size. To change the line width to be 12.5 cm, one would give:

\setlength{\textwidth}{12.5cm}

2.4.2 Rubber lengths

Some parameters expect a *rubber* length. These are lengths that can be stretched or shrunk by a certain amount. The syntax for a rubber length is:
nominal_value plus stretch_value minus shrink_value

where the nominal_value, stretch_value, and shrink_value are each a length. For example,

\setlength{\parskip}{1ex plus0.5ex minus0.2ex}

means: the extra line spacing between paragraphs, called \parskip, is to be the height of the x in the current font, but it may be increased to 1.5 or reduced to 0.8 times that size.

One special rubber length is \fill. This has the natural length of zero but can be stretched to any size.

2.5 Special characters

2.5.1 Spaces

The space or blank character has some properties different from those of normal characters, some of which have already been mentioned in Section 2.1. During processing, blanks in the input text are replaced by rubber lengths (Section 2.4.2) in order to allow the line to fill up to the full line width. As a result, some peculiar effects can occur if one is not aware of the following rules:

• one blank is the same as a thousand, only the first one counts;
• blanks at the beginning of an input line are ignored;
• blanks terminating a command name are removed;
• the end of a line is treated as a blank.

Some of the consequences of these rules are that there may be as many blanks as desired between words or at the beginning of a line (to make the input text more legible) and that a word may come right at the end of a line without the spacing between it and the next word disappearing. To force a space to appear where it would otherwise be ignored, one must give the command \_ (a \ followed by a space character, made visible here by the symbol \).

To ensure that certain words remain together on the same line, a protected space is inserted between them with the ~ character (Section 2.7.1, page 28). Multiple protected spaces are all printed out, in contrast to normal spaces.

Sometimes it is necessary to suppress the space that appears because of the new line. In this case, the last character in the line must be the comment character % (Section 4.11).

Paragraphs are separated in the source text by blank lines. As for blank characters, one blank line is the same as a thousand.
Instead of a blank line, the command \par may also be used to indicate the end of a paragraph.

2.5.2 Quotation marks

The quotation marks found on the typewriter " are not used in book printing. Instead different characters are used at the beginning and end, such as ‘single quotes’ and “double quotes”. Single quotes are produced with ‘ and ’, while double quotes are made by typing the respective characters twice: ‘‘ for “ and ”’ for ”. Furthermore the typewriter character " will also generate the double closing quote ”. However, it should be avoided since it can lead to confusion.

2.5.3 Hyphens and dashes

In book printing, the character that appears on the typewriter as – comes in various lengths: -, –, —. The smallest of these, the hyphen, is used for compound words such as father-in-law and for word division at the end of a line; the middle-sized one, the en dash, is used in ranges of numbers, for example, pages 33–36; and the largest, the em dash, is used as punctuation—what is normally called the dash. These are generated by typing the hyphen character one, two, or three times, so that - yields -, while -- makes –, and --- produces —. A fourth type of dash is the minus sign –, which is entered in math mode as $-$ (Chapter 5).

2.5.4 Printing command characters

As mentioned in Section 2.1, the characters # $ ˜ _ ˆ % { } are inter-

preted as commands. To print them as text, one must give a command consisting of \ plus that character.

\$ = \$ & = \& % = \% # = \# _ = \_ \{} = \{ \}

2.5.5 The special characters § † ‡ ¶ © and £

These special characters do not exist on the computer keyboard. They can however be generated by special commands as follows:

§ = \$ † = \dag ‡ = \ddag ¶ = \P © = \copyright £ = \pounds

The production of Greek letters and other mathematical symbols is described in Chapter 5.
2.5.6 Non-English letters

Special letters that exist in languages other than English can also be generated with \TeX. These are:

\oe = \{\oe\} \OE = \{\OE\} \ae = \{\ae\} \AE = \{\AE\} \= \{\aa\} \AA = \{\AA\} i = \{\i\} o = \{\o\} O = \{\O\} l = \{\l\} L = \{\L\} \= \{\ss\} SS = \{\SS\} i = \{\i\} Ångström may be written as \{\AA\}ngstr\{\o\}m while Karlstraße can be input as Karlstra\{\ss\}e. The ‘letter’ \SS is the upper case equivalent of \ss, used for automatic conversion between upper and lower case.

However, see Section 2.5.9 for the possibility of entering such characters directly.

2.5.7 Accents

In non-English languages, there is a multiplicity of diacritical marks or accents, most of which can be printed with \TeX:

\= o = \{"o\} \= o = \{"o\} \= o = \{"o\} \= o = \{"o\} \= o = \{"o\} Ø = \{\O\} l = \{\l\} L = \{\L\} \= \{\ss\} SS = \{\SS\} i = \{\i\} o = \{\o\} O = \{\O\} l = \{\l\} L = \{\L\} \= \{\ss\} SS = \{\SS\} i = \{\i\}

The accent commands consisting of a non-letter may also be given without the curly braces:

\ = o \ = o \ = o \ = o \ = o \ = o \ = o

The letter accent commands should always be used with the curly braces.

2.5.8 The euro symbol

The euro symbol € (or \e) is too new to be part of the original \LaTeX, but it can be produced with the help of some additional fonts and contributed packages. Just which package you may use depends on your installation, and whether you have access to these additional fonts.

The Text Companion fonts, described in Section G.4.4, do contain a euro symbol. Since these fonts should be part of every modern \LaTeX installation, you should be able to use their euro symbol if all else fails.

\usepackage{textcomp}

Package: textcomp

The package textcomp must be loaded in the preamble with
which defines many commands including \texteuro to print the symbol €. Since the European Commission originally dictated that it should only be printed in a sans serif font, it is better to issue \textsf{\texteuro} to produce €. (The font selection commands are described in Section 4.1.4.) If you are going to use this very frequently, you might want to define a shortcut named \euro with

\newcommand{\euro}{\textsf{\texteuro}}

as described in Section 8.3 on defining commands.

A better solution is presented by the eurosym package by Henrik Theiling and the associated fonts that come with it, which bear the names feymr10, feybr10, and so on. This package defines the \euro command to print €, which changes automatically to bold € and slanted € as needed.

The europs package by Joern Clausen interfaces to the type 1 (PostScript) euro fonts published by Adobe. For licensing reasons, these fonts may only be obtained from Adobe directly, even though free of charge (see Section B.2). This package provides the command \EUR for a symbol that varies with font family (Roman €25, sans serif €25, and typewriter €25) as well as for bold €25 and slanted €25. There is also a command \EURofc for the invariable symbol €.

Finally, the package eurosans by Walter Schmidt also addresses the Adobe euro fonts, again with the command \euro, with the same behavior as that of eurosym: always sans serif family, but changes with the other font attributes.

The table below summarizes the above packages:

<table>
<thead>
<tr>
<th>Package</th>
<th>Command</th>
<th>Fonts</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>textcomp</td>
<td>\texteuro</td>
<td>Text Companion</td>
<td>Non standard symbol</td>
</tr>
<tr>
<td>eurosym</td>
<td>\euro</td>
<td>Eurosym</td>
<td>Sans serif, variable</td>
</tr>
<tr>
<td>europs</td>
<td>\EUR</td>
<td>PostScript</td>
<td>Varies with font family</td>
</tr>
<tr>
<td></td>
<td>\EURofc</td>
<td></td>
<td>Invariable, official</td>
</tr>
<tr>
<td>eurosans</td>
<td>\euro</td>
<td>PostScript</td>
<td>Sans serif, variable</td>
</tr>
</tbody>
</table>

So which package should one use? That really depends on the fonts available. Since the Adobe fonts can never be distributed with a \TeX installation, they must be actively fetched and installed. However, it is worth doing so, because the European Commission has revised its initial directive and now allows the euro symbol to be typographically matched to the text, which is also standard practice in Europe today. This strengthens the case for the europs package and the \EUR command for €, at least for Roman fonts.
26 Chapter 2. Text, Symbols, and Commands

2.5.9 Typing special symbols directly

The commands for producing the special characters and accented letters in the previous sections may be suitable for typing isolated 'foreign' words, but become quite tedious for inputting large amounts of text making regular use of such characters. Most computer systems provide non-English keyboards with appropriate fonts for typing these national variants directly. Unfortunately, the coding of such extra symbols is by no means standard, depending very much on the computer system.

For example, the text Gauß meets Ampère entered with an MS-DOS editor (code page 437 or 850) appears in a Windows application as Gauâ meets AmpŠre and on a Macintosh as Gau· meets Ampâre. Since \(\LaTeX\) is intended to run on all systems, it simply ignores all such extra character codes on the grounds that they are not properly defined.

The \texttt{inputenc} package solves this problem. It not only informs \(\LaTeX\) which input coding scheme is being used, it also tells it what to do with the extra characters. One invokes it with

\begin{verbatim}
\usepackage[code]{inputenc}
\end{verbatim}

where \texttt{code} is the name of the coding scheme to be used. The current list of allowed values for \texttt{code} (more are added with each \LaTeX\ update) can be found in Table D.1 on page 462. For most users, the most interesting codes are:

- \texttt{cp437} IBM code page 437 (DOS, North America)
- \texttt{cp850} IBM code page 850 (DOS, Western Europe)
- \texttt{applemac} Macintosh encoding
- \texttt{ansinew} Windows ANSI encoding

In short, you should select \texttt{applemac} for a Macintosh, and \texttt{ansinew} for Windows, and one of the others if you are working with DOS.

Documents making use of this package are fully portable to other computer systems. The source text produced with a DOS editor may still look very strange to a human user reading it on a Macintosh, but when the Macintosh \LaTeX\ processes it, the proper DOS interpretations will be applied so that the end result is what the author intended.

See Section D.5 for more details.

2.5.10 Ligatures

In book printing, certain combinations of letters are not printed as individuals but as a single symbol, a so-called ligature. \(\TeX\) processes the letter combinations \texttt{ff}, \texttt{fi}, \texttt{fl}, \texttt{ffi}, and \texttt{ffl} not as

\begin{verbatim}
ff, fi, fl, ffi, ffl
\end{verbatim}

Ligatures may be broken, that is, forced to be printed as separate letters, by inserting \texttt{\slash} between the letters. This is sometimes desired for such words as \texttt{shelfful} (\texttt{shelf\slash fu1}), which looks rather strange when printed with the normal \texttt{ff} ligature, \texttt{shelfful}. 
2.5.11 The date

The current date can be placed at any point in the text with the command \today. The standard form for the date is the American style of month, day, year (for example, January 15, 2004). The British form (15th January 2004) or the date in other languages can be generated with the help of the \TeX commands \day, \month, and \year, which return the current values of these parameters as numbers. Examples of how such a new \today command may be made are shown on page 461 in Section D.4.2.

It is in fact better to enter the date explicitly, rather than to rely on \today. Reprocessing a two-year-old \LaTeX source file will yield a document with the processing date, not the date when the text was written.

2.6 Exercises

**Exercise 2.1:** This exercise tests the basic operations of running the \LaTeX program with a short piece of text. A few simple commands are also included. Use a text editor to produce the following source text and store it in a file named exer.tex.

```
\documentclass{article}
\begin{document}
Today (\today) the rate of exchange between the British pound and American dollar is £1 = $1.58, an increase of 1\% over yesterday.
\end{document}
```

Process this source file with \LaTeX by clicking the appropriate icon, or by issuing \texttt{latex exer} in a command window. If the processing occurs without any error messages, the .dvi file \texttt{exer.dvi} will have been successfully created and may be viewed by a dvi previewer or sent to a printer. The final printed result should look as follows except that your current date will appear:

Today (January 15, 2004) the rate of exchange between the British pound and American dollar is £1 = $1.58, an increase of 1\% over yesterday.

Note the following points about the commands used:

- no blank is necessary after \today because the ) suffices to terminate it;
- the blank after \pounds is optional and it is not printed in the output;
- the commands \$ and \% do not require blanks to terminate them; if blanks are given, they will be printed.

**Exercise 2.2:** Take some text of about 3/4 of a page long out of a book or journal article and type it into a \LaTeX source file. Pay attention that the paragraphs are separated by blank lines. Use the same set of commands as in Exercise 2.1, that is, put the text between the commands \begin{document}...\end{document} and repeat the procedures for obtaining the output.
Exercise 2.3: If you are likely to need the euro symbol in your work, try redoing Exercise 2.1 as follows:

```latex
\documentclass{article}
\usepackage{eurosym}
\begin{document}
Today (\today) the rate of exchange between the British pound and European euro is \pounds 1 = \euro1.46, an increase of 1\% over yesterday.
\end{document}
```

If this fails, try one of the other packages described in Section 2.5.8, substituting \textsf{\texteuro} or \EUR for \euro as required.

2.7 Fine-tuning text

The subject of the section concerns pure typographical markup, and has nothing to do with the logical markup that we wish to stress in this book. Unfortunately, there are times when the author or editor does have to help the typesetting program to achieve good appearance.

2.7.1 Word and character spacing

The spacing between words and characters is normally set automatically by \TeX, which not only makes use of the natural width of the characters but also takes into account alterations for certain character combinations. For example, an A followed by a V does not appear as AV but rather as AV; that is, they are moved together slightly for a more pleasing appearance. Interword spacing within one line is uniform, and is chosen so that the right and left ends match exactly with the side margins. This is called left and right justification. \TeX also attempts to keep the word spacing for different lines as nearly the same as possible.

Words that end with a punctuation mark are given extra spacing, depending on the character: following a period \textquoteleft or exclamation mark \textquoteleft!, there is more space than after a comma \textquoteleft,\textquoteleft. This corresponds to the rule in English typesetting that there should be extra spacing between sentences. In certain cases, the automatic procedures do not work properly, or it is desirable to override them, as described in the next sections.

Sentence termination and periods

\TeX interprets a period following a lower case letter to be the end of a sentence where additional interword spacing is to be inserted. This leads to confusion with abbreviations such as \textquoteleft i. e.,\textquoteright Prof. Jones, or \textquoteright Phys. Rev., where the normal spacing is required. This can be achieved by using the characters \textquoteleft\textasciitilde or \textbackslash instead of the normal blank. (The character \textbackslash is
simply a symbol for the blank which is otherwise invisible.) Both these methods insert the normal interword spacing; in addition, \textasciitilde{} is a \textit{protected space} that prevents the line from being broken at this point. The above examples should be typed in as i.\textasciitilde{}e., Prof.\textasciitilde{}Jones, and \textit{Phys.}\ Rev., producing \textit{i. e.}, \textit{Prof. Jones}, and \textit{Phys. Rev.} with the correct spacing and forcing the first two to be all on one line. In the third case, there is nothing wrong with putting \textit{Phys.} and \textit{Rev.} on different lines.

A period following an upper case letter is not interpreted as the end of a sentence, but as an abbreviation. If it really is the end of a sentence, then it is necessary to add \texttt{@} before the period in order to achieve the extra spacing. For example, this sentence ends with NASA. It is typed in as \texttt{This sentence ends with NASA@}.

\textbf{French spacing}

The additional interword spacing between sentences can be switched off with the command \texttt{\frenchspacing}, which remains in effect until countermanded with \texttt{\nonfrenchspacing}. In this case, the command \texttt{@} is ignored and may be omitted. This paragraph has been printed with \texttt{\frenchspacing} turned on, so that all word spacings within one line are the same. It corresponds to the normal rule for non-English typesetting.

\textbf{Character combinations “‘ and ’”}

A small spacing is produced with the command \texttt{\,}. This may be used, for example, to separate the double quotes “ and ” from the corresponding single quotes ‘ and ’ when they appear together. For example, the text “‘\texttt{,‘Beginning’} and ‘End’\texttt{,’}” produces “‘Beginning’ and ‘End’”.

\textbf{Inserting arbitrary spacing}

Spacing of any desired size may be inserted into the text with the commands

\begin{verbatim}
\hspace{space}
\hspace*{space}
\end{verbatim}

where \texttt{space} is the length specification for the amount of spacing, for example 1.5cm or 3em. (Recall that one \texttt{em} is the width of the letter M in the current typeface.)

This command puts blank space of width \texttt{space} at that point in the text where it appears. The standard form (without \texttt{*}) has no effect if it should come at the beginning of an output line, just as normal blanks are removed at the beginning of lines. The \texttt{*}-form, on the other hand, inserts the spacing no matter where it occurs.

A blank before or after the command will also be included:
The length specification may be negative, in which case the command works as a backspace for overprinting characters with other ones, or moving them closer together. For example, there is an energy unit in physics called electron volt, abbreviated ‘eV’, which looks much better if the two letters are nearer together, as ‘eV’, with e\hspace{-.12em}V.

The command \hfill is an abbreviation for \hspace{\fill} (see Section 2.4.2). It inserts enough space at that point to force the text on either side to be pushed over to the left and right margins. With \texttt{Left\hfill Right} one produces

\begin{verbatim}
Left \hfill Right
\end{verbatim}

Multiple occurrences of \hfill within one line will each insert the same amount of spacing so that the line becomes left and right justified. For example, the text \texttt{Left\hfill Center\hfill Right} generates

\begin{verbatim}
Left \hfill Center \hfill Right
\end{verbatim}

If \hfill comes at the beginning of a line, the spacing is suppressed in accordance with the behavior of the standard form for \hspace. If a rubber space is really to be added at the beginning or end of a line, \hspace*{\fill} must be used instead. However, \LaTeX{} also offers a number of commands and environments to simplify most such applications (see Section 4.2.2).

A number of other fixed horizontal spacing commands are available:

\begin{verbatim}
\quad \text{and} \quad \qquad
\end{verbatim}

The command \quad inserts a horizontal space equal to the current type size, that is, 10 pt for a 10 pt typeface, whereas \qquad inserts twice as much.

**Inserting variable …… and _____ sequences**

Two commands that work exactly the same way as \hfill are

\begin{verbatim}
\dotfill \text{and} \quad \hrulefill
\end{verbatim}

Instead of inserting empty space, these commands fill the gap with dots or a ruled line, as follows:

\begin{verbatim}
Start \dotfill\hfill Finish\\and
Left \hrulefill\hfill Center \hrulefill\hfill Right\\ produce
\end{verbatim}

\begin{verbatim}
Start ......................................................... Finish
Left ........................................... Center ................................... Right
\end{verbatim}
Any combination of \hfill, \dotfill, and \hrulefill may be given on one line. If any of these commands appears more than once at one location, the corresponding filling will be printed that many more times than for a single occurrence.

Departure \dotfill\dotfill\dotfill 8:30 \hfill \hfill
Arrival \hrulefill 11:45\*

Departure \ldots \ldots \ldots \ldots 8:30 \quad \text{Arrival} \phantom{11:45}

2.7.2 Line breaking

Breaking text into lines is done automatically in \TeX{} and \LaTeX{}. However, there are times when a line break must be forced or encouraged, or when a line break is to be suppressed.

The commandourke \\\

A new line with or without additional line spacing can be achieved with the command \\\/. Its syntax is

\[ [\text{space}] \]
\[ ** [\text{space}] \]

The optional argument \textit{space} is a length that specifies how much additional line spacing is to be put between the lines. If it is necessary to start a new page, the additional line spacing is not included and the new page begins with the next line of text. The *-form prevents a new page from occurring between the two lines.

With **[10cm], the current line is ended and a vertical spacing of 10 cm is inserted before the next line, which is forced to be on the same page as the current line. If a page break is necessary, it will be made before the current line, which is then positioned at the top of the new page together with the 10 cm vertical spacing and the next text line.

The command newline is identical to \ without the option \textit{space}. That is, a new line is started with no additional spacing and a page break is possible at that point.

Both commands may be given only within a paragraph, and not between them where they would be meaningless.

Further line-breaking commands

The command \texttt{\linebreak} is used to encourage or force a line break at a certain point in the text. Its form is

\texttt{\linebreak[num]}
where *num* is an optional argument, a number between 0 and 4 that specifies how important a line break is. The command recommends a line break, and the higher the number the stronger the recommendation. A value of 0 allows a break where it otherwise would not occur (like in the middle of a word), whereas 4 compels a line break, as does `\linebreak` without *num*. The difference between this command and `\` or `\newline` is that the current line will be fully justified, that is, interword spacing will be added so that the text fills the line completely. With `\` and `\newline`, however, the line is filled with empty space after the last word and the interword spacing remains normal.

The opposite command

`\nolinebreak[\textit{num}]`

discourages a line break at the given position, with *num* specifying the degree of discouragement. Again, `\nolinebreak` without a *num* argument has the same effect as `\nolinebreak[4]`, that is, a line break is absolutely impossible here.

Another way of forcing text to stay together on one line is with the command `\mbox{\textit{text}}`. This is convenient for expressions such as ‘Voyager-1’ to stop a line break at the hyphen.

### 2.7.3 Vertical spacing

It is possible to add extra vertical spacing of amount *space* between particular paragraphs using the commands

`\vspace{\textit{space}}`

`\vspace*[\textit{space}]`

The *-form will add the extra space even when a new page occurs, or when the command appears at the top of a new page. The standard form ignores the extra vertical spacing in these situations.

If these commands are given within a paragraph, the extra space is inserted after the current line, which is right and left justified as usual.

The *space* parameter may even be negative, in order to move the following text higher up the page than where it would normally be printed.

The command `\vfill` is an abbreviation for `\vspace{\fill}` (see Section 2.4.2). This is the equivalent of `\hfill` for vertical spacing, inserting enough blank vertical space to make the top and bottom of the text match up exactly with the upper and lower margins. The comments on multiple occurrences of `\hfill` also apply to `\vfill`. If this command is given at the beginning of a page, it is ignored, just like the standard form of `\vspace{\fill}`. If a rubber space is to be put at the top of a page, the *-form `\vspace*[\fill]` must be used.

Further commands for increasing the spacing between paragraphs are
which add vertical spacing depending on the font size declared in the document class.

### 2.7.4 Page breaking

Breaking text into pages occurs automatically in \TeX{} and \LaTeX{}, just as for line breaking. Here again, it may be necessary to interfere with the program’s notion of where a break should take place.

**Normal pages**

The commands

\begin{verbatim}
\pagebreak[\textit{num}]
\nopagebreak[\textit{num}]
\end{verbatim}

are the equivalents of \linebreak and \nolinebreak for page breaking. If \pagebreak appears between two paragraphs, a new page will be forced at that point. If it comes within a paragraph, the new page will be implemented after the current line is completed. This line will be right and left justified as usual.

The command \nopagebreak has the opposite effect: between paragraphs, it prevents a page break from occurring there, and within a paragraph, it stops a page break that might take place at the end of the current line.

Optional numbers between 0 and 4 express the degree of encouragement or discouragement for a page break. The analogy with the command \linebreak goes further: just as the line before the break is left and right justified with extra interword spacing, in the same way the page before the break is expanded with interline spacing to make it top and bottom justified.

The proper command to end a page in the middle, fill it with blank spacing, and go on to a new page is

\begin{verbatim}
\newpage
\end{verbatim}

which is equivalent to \newline with regard to page breaking.

**Pages with figures and tables**

If the text contains tables, pictures, or reserved space for figures, these are inserted at the location of the corresponding command, provided that there is enough room for them on the current page. If there is not enough space, the text continues and the figure or table is stored to be put on a following page.

The command
\clearpage

ends the current page like \newpage and in addition outputs all the pending figures and tables on one or more extra pages (Chapter 7).

Two-column pages

If the document class option twocolumn has been chosen, or the command \twocolumn is in effect, then the two commands \pagebreak and \newpage end the current column and begin a new one, treating columns as pages. On the other hand, \clearpage and \cleardoublepage (see below) terminate the current page, inserting an empty right column if necessary.

Two-sided pages

An additional page-breaking command is available when the document class option twoside has been selected:

\cleardoublepage

which functions exactly the same as \clearpage (the current page is terminated, all pending figures and tables are output) but in addition, the next text will be put on to an odd-numbered page. If necessary, an empty page with an even number is printed to achieve this.

Controlling page breaks

\LaTeX provides the possibility of increasing the height of the current page slightly with commands

\enlargethispage{size}
\enlargethispage*{size}

which add the length size to \textheight for this one page only. Sometimes the difference of a few points is all that is necessary to avoid a bad page break. The *-form of the command also shrinks any interline spacing as needed to maximize the amount of text on the page.

2.8 Word division

When a line is to be right and left justified, it often turns out that the break cannot be made between whole words without either shoving the text too close together or inserting huge gaps between the words. It is then necessary to split a word. This fundamental task is performed by \TeX, the underlying basis of \LaTeX, by means of a word-dividing algorithm
that works (almost) perfectly for English text, which is more than can be said for most authors. Nevertheless, even it makes mistakes at times which need to be corrected by human intervention.

If normal \TeX/\LaTeX{} is used for other languages, or if foreign words appear in English text, incorrect hyphenations are very likely to appear. (See Section 2.8.4 and Chapter 11 for more about \LaTeX{} with other languages.) In these cases too, something must be done to override \TeX{}'s hyphenation rules, as described below.

### 2.8.1 Manual hyphenation

The simplest way to correct a wrongly divided word is to include a `-command at the right place within the word. The word manuscript, for example, will not be hyphenated at all, so if it causes problems with breaking a line, write it as man\-u\-script. This tells \TeX{} to divide the word as necessary either as man-uscript or as manu-script, and to ignore its normal rules.

The `- command merely makes hyphenation possible at the indicated locations; it does not force it. If the author absolutely insists in dividing a word at a certain point, say between the u and s in manuscript, he or she can type manu-\linebreak script to achieve this. However, this brute force method is not recommended, because the line break will always occur here even if the text is later changed.

For English text, the spelling of a word remains the same when it is hyphenated, something that is not true in other languages. In traditional German spelling, for example, if ck is split, it becomes k-k. \TeX{} allows such behavior with the general hyphenation command

\[\texttt{discretionary\{before\}\{after\}\{without\}}\]

where \texttt{before} and \texttt{after} are the letters (with hyphen) that come on either side of the break if division takes place, and \texttt{without} is the normal text with no hyphenation. Thus Boris Becker's name should be typed as

\[\texttt{Boris Be\texttt{\linebreak discretionary\{k-\}\{k\}\{ck\}er}}\]

something that one only wants to do in exceptional situations. Incidentally, the `- command is shorthand for \texttt{\discretionary{-}{}{}}.

**Note:** In today's new German spelling, \texttt{ck} is never split. This reformed spelling is still controversial however.

### 2.8.2 Hyphenation list

Words that are incorrectly hyphenated and that appear frequently within the document can be put into a list of exceptions in the preamble, to avoid laboriously inserting `- every time:
The list consists of a set of words, separated by blanks or new lines, with the allowed division points indicated by hyphens. For example,

\hyphenation{man-u-script com-pu-ter gym-na-sium coun-try-man re-sus-ci-tate ... }

The list may contain only words with the normal letters a–z, with no special characters or accents. However, if the \inputenc package is loaded, then the directly typed letters are also be included in the automatic hyphenation.

2.8.3 Suppressing hyphenation

Another means of avoiding bad word divisions is to turn hyphenation off, at least for a paragraph or two. Actually the command

\begin{sloppypar} paragraph text \end{sloppypar}

does not prevent word division, but does permit larger interword spacings without giving a warning message. This means that practically all lines are broken between words. It is also possible to put the command \sloppy in the preamble or in the current environment to reduce the number of word divisions in the whole document or within the environment scope. This is recommended when the line width is rather narrow.

When the command \sloppy is in effect, it is possible to undo it temporarily and to turn hyphenation back on with the command \fussy.

2.8.4 Word division with multilingual text

Multiple hyphenation lists may be included in the \TeX format, making it possible to switch hyphenation schemes within one document, using the \TeX command \language. This command may be used as part of language-specific adaptations to translate certain explicit English words in the output (such as ‘Contents’), to simplify accents or punctuation, and to alter the definition of the date command \today. This topic is treated in more detail in Chapter 11.
3 Document Layout and Organization

3.1 Document class

The first command in the preamble of a LaTeX file determines the global processing format for the entire document. Its syntax is:

\documentclass[options]{class}

where some value of class must be given, while [options] may be omitted if default values are acceptable.

The standard values of class, of which one and only one may be given, are: book, report, article, or letter. (The properties of the letter class are explained in Chapter 16.) The basic differences between these classes lie not only in the page layouts, but also in the organization. An article may contain parts, sections, subsections, and so on, while a report can also have chapters. A book also has chapters, but treats even and odd pages differently; also, it prints running heads on each page with the chapter and section titles.

Other classes besides the standard ones exist, as contributions for specific journals, or for book projects. These will have their own set of options and additional commands, which should be described in separate documentation or instructions. However, since they are normally modifications of one of the standard classes, most of the following options apply to them too.

3.1.1 Standard class options

The options available allow various modifications to be made to the formatting. They can be grouped as follows.

Selecting font size

The basic font size is selected with one of the options
This is the size of the font in which the normal text in the document will be set. The default is 10pt, which means that this is the value assumed if no size option is specified. All other font size declarations are relative to this standard size, so that the section titles, footnotes, and so on will all change size automatically if a different basic font size is selected.

Specifying paper size

\LaTeX{} calculates the text line width and lines per page according to the selected font size and paper mode. It also sets the margins so that the text is centered both horizontally and vertically. In order to do this, it needs to know which paper format is being used. This is specified by one of the following options:

- \texttt{letterpaper} (11 × 8.5 in)
- \texttt{a4paper} (29.7 × 21 cm)
- \texttt{legalpaper} (14 × 8.5 in)
- \texttt{a5paper} (21 × 14.8 cm)
- \texttt{executivepaper} (10.5 × 7.25 in)
- \texttt{b5paper} (25 × 17.6 cm)

The default is \texttt{letterpaper}, American letter size paper, 11 × 8.5 in.

Normally, the paper format is such that the longer dimension is the vertical one, the so-called \textit{portrait} mode. With the option \texttt{landscape} the shorter dimension becomes the vertical one, the \textit{landscape} mode. (One still has to ensure that the output is printed as landscape; see for example page 232.)

Page formats

The text on the page may be formatted into one or two columns with the options

- \texttt{onecolumn}
- \texttt{twocolumn}

The default is \texttt{onecolumn}. In the case of the \texttt{twocolumn} option, the separation between the columns as well as the width of any rule between them may be specified by \texttt{\columnsep} and \texttt{\columnseprule}, described below.

The even- and odd-numbered pages may be printed differently according to the options

- \texttt{oneside}
- \texttt{twoside}

With \texttt{oneside}, all pages are printed the same; however, with \texttt{twoside}, the running heads are such that the page number appears on the right on odd
3.1. Document class

pages and on the left on even pages. *It does not force the printer to output double-sided.* The idea is that when these are later printed back-to-back, the page numbers are always on the outside where they are more easily noticed. This is the default for the book class. For article and report, the default is oneside.

With the book class, chapters normally start on a right-hand, odd-numbered page. The options

```
openright  openany
```

control this feature: with openany a chapter always starts on the next page, but with openright, the default, a blank page may be inserted if necessary.

Normally the title of a book or report will go on a separate page, while for an article, it is placed on the same page as the first text. With the options

```
notitlepage  titlepage
```

this standard behavior may be overruled. See Sections 3.3.1 and 3.3.2.

Further options

The remaining standard options are:

- `leqno`  Equation numbers in displayed formulas will appear on the left instead of the normal right side (Section 5.1).

- `fleqn`  Displayed formulas will be set flush left instead of centered (Section 5.1). The amount of indentation may be set with the parameter `\mathindent` described below.

- `openbib`  The format of bibliographies may be changed so that segments are set on new lines. By default, the texts for each entry are run together.

- `draft`  If the \LaTeX line-breaking mechanism does not function properly and text must stick out into the right margin, then this is marked with a thick black bar, to make it noticeable.

- `final`  The opposite of `draft`, and the default. Lines of text that are too wide are not marked in any way.

If multiple options are to be given, they are separated by commas, as for example, `\documentclass[11pt, twoside, fleqn]{article}`. The order of the options is unimportant. If two conflicting options are specified, say `oneside` and `twoside`, it is not obvious which one will be effective. That depends entirely on the definitions in the class file itself, so it would be best to avoid such situations.
Parameters associated with some options

Some options make use of parameters that have been given certain default values:

\texttt{\textbackslash mathindent}

specifies the indentation from the left margin for the equation numbers when \texttt{fleqn} is selected (Section 5.1);

\texttt{\textbackslash columnssep}

specifies the space between the two columns for the \texttt{twocolumn} option (see Figure H.2 on page 603);

\texttt{\textbackslash columnseprule}

determines the width of the vertical line between the two columns for the \texttt{twocolumn} option. The default is zero width, that is, no vertical rule (see Figure H.2).

The standard values of these parameters may be changed with the \LaTeX command \texttt{\setlength}. For example, to change \texttt{\mathindent} to 2.5 cm, give

\texttt{\setlength{\mathindent}{2.5cm}}

These parameters may be assigned values either in the preamble or at any place in the document. Parameters in the preamble apply to the entire document, whereas those within the text are in effect until the next change or until the end of the environment in which they were made (Section 2.3). In the latter case, the previous values become effective once more.

\textbf{Exercise 3.1:} Take your text file from Exercise 2.2 and change the initial command \texttt{\documentclass[article]} first to \texttt{\documentclass[11pt]{article}} and then to \texttt{\documentclass[12pt]{article}} and print the results of each \LaTeX processing. Compare the line breaking of these outputs with that of Exercise 2.2.

Note: if there are some improper word divisions, you call tell \LaTeX where the correct division should occur with the command \texttt{\-,} for example, \texttt{man\textbackslash u\textbackslash script}. (This is one of the few words that the \TeX English word divider does not handle properly.) Additional means of modifying word division are given in Section 2.8. If there are warnings of the sort \texttt{Overfull \hbox ...} during the \LaTeX processing, \TeX was not able to break the lines cleanly. In the output, these lines will extend beyond the right margin. The usual cause is that \TeX was not able to divide some word, either because it is indivisible or because \TeX's word division routines were not adequate. Here again a suggested hyphenation in the text can solve the problem. Other solutions will be given shortly.

\textbf{Exercise 3.2:} Now employ \texttt{\documentclass[twocolumn]{article}} in your text file. If you now receive a number of warnings with \texttt{Underfull \hbox ...}, then these lines will indeed be left and right justified but will have too much empty
3.1. Document class

Space between the words. Check the output yourself to see whether the word spacing is acceptable. If not, try giving some hyphenation suggestions in the first words of the next line.

Note: if you use the classes book or report instead of article in the preceding exercises, you will notice no difference in the outputs. These classes affect the subsequent structural elements of the document. Basically, you should use article for short articles (say 10–20 pages) and report for longer reports that are to be organized into chapters. The chapters always begin on a new page. The class book is available for producing books.

3.1.2 Loading packages

In Section 1.5.3 we explained how \LaTeX{} can be extended by packages which are either part of the core installation or contributed by engaged users. How to write your own packages is described in Appendix D.

A package is nothing more than a set of \LaTeX{} (or \TeX{}) commands stored in a file with the extension .sty, although there are some special commands that may only appear within them. To invoke a package, simply call

\usepackage\{package\}

in the preamble, where package is the root name of the file. More than one package may be loaded with one call to \usepackage. For example, two packages provided with standard \LaTeX{} are stored in files makeidx.sty (Section 9.4.3) and ifthen.sty (Section 8.3.5). They may be read in together with

\usepackage\{makeidx,ifthen\}

A package may have options associated with it, which may be selected in the same way as for document classes: by including the option names within square braces. The general syntax is thus:

\usepackage\{opt1,opt2,...\}\{package1,package2,...\}

where all the listed options will be applied to all the selected packages. If any of the packages does not understand one of the options, a warning message is output to the monitor.

3.1.3 Global and local options

One interesting feature about options specified with the \documentclass command is that they also apply to any packages that follow. This means that if several packages all take the same option, it is only necessary to declare it once in \documentclass. For example, one might design a package to modify article for generating a local house style that might do different things for single or double column text; this package could make use of the class options onecolumn and twocolumn to achieve this. Or it could elaborate on the draft
option to produce double line spacing, as for a manuscript. Alternatively, sev-
eral packages might have language-dependent features that could be activated
with options like french or german; it is sufficient to list such options only in
\documentclass to apply them to all packages. Such options are called global,
for they are passed on to all subsequent packages automatically.

Global options need not be limited to the standard class options listed in
Section 3.1.1. A warning message is printed only if neither the class nor any
of the packages understand one or more of them. By contrast, any options
specified with \usepackage will be applied only to those packages listed in that
one command; and it is applied to all of them. A warning is printed if one or
more of those packages does not recognize any one of these local options.

3.1.4 Class and package versions

Class and package files normally have an internal version specification in the
form of their release date, as yyyy/mm/dd. If you wish to make use of some
feature that you know was added on a certain date, you include that date in
square brackets after the class or package name. An example of this is shown in
Section 3.2.4 on page 46.

The version date may also be added to the \documentclass command to
ensure that the right version of the class file is being employed. The reason for
doing this is to ensure that the source files are processed properly, say on other
systems.

3.2 Page style

The basic page format is determined by the page style. With one exception,
this command is normally given in the preamble. Its form is:

\pagestyle{style}

The mandatory argument style takes on one of the following values:

plain The page head is empty, the foot contains the centered page
number. This is the default for the article and report classes
when no \pagestyle is given in the preamble.

empty Both head and footlines are empty; no page numbers are printed.

headings The head contains the page number as well as title information
(chapter and section headings); the foot is empty. This is the
default for book class.

myheadings The same as headings except that the page titles in the head are
not chosen automatically but rather are given explicitly by the
commands \markright or \markboth (see below).
The command
\thispagestyle{style}
functions exactly as \pagestyle except that it affects only the current page. For example, the page numbering may be suppressed for just the current page with the command \thispagestyle{empty}. It is only the printing of the page number that is suppressed; the next page will be numbered just as though the command had never been given.

3.2.1 Heading declarations

For the page styles headings and myheadings, the information appearing in the headline may be given with the declarations

\markright{right\_head}
\markboth{left\_head}{right\_head}

The declaration \markboth is used with the document class option twoside, with even-numbered pages considered to be on the left and odd-numbered pages on the right. Furthermore, the page number is printed on the left side of the head for a left page and on the right side for a right page.

For one-sided output, all pages are considered to be right-handed. In this case, the declaration \markright is appropriate. It may also be used with two-sided output to overwrite the right\_head given in \markboth.

With the page style headings, the standard titles in the page headline are the chapter, section, or subsection headings, depending on the document and page style, according to the following scheme:

<table>
<thead>
<tr>
<th>Style</th>
<th>Left Page</th>
<th>Right Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>book, report</td>
<td>one-sided</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>two-sided</td>
<td>Chapter</td>
</tr>
<tr>
<td>article</td>
<td>one-sided</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>two-sided</td>
<td>Section</td>
</tr>
</tbody>
</table>

If there are more than one \section or \subsection on a page, it is the heading of the last one that appears in the page head.

3.2.2 Customized head and footlines

The standard page styles described in Section 3.2 select how the head and footlines are to appear, and what information they contain. This is a very limited choice, and the fancyhdr package by Piet van Oostrum offers the user considerable more flexibility.

This package makes available an additional page style named fancy which the user can easily redefine. Head and footlines consist each of
three parts, left, center, right, each of which can be individually defined with

\lhead{Left head} \chead{Center head} \rhead{Right head}  \\
lfoot{Left foot}  \cfoot{Center foot}  \rfoot{Right foot}

where the various texts may be explicit, or a command like \thepage to print the current page number. Both head and footlines may be decorated with a rule, the widths of which are set by commands \headrulewidth and \footrulewidth. By default, the fancy head and footlines are much the same as for the headings page style, but the head rule is set to 0.4 pt and the foot rule set to 0 (no rule). The rules may be redefined with, for example,

\renewcommand{\footrulewidth}{0.4pt}

to turn the foot rule on.

The above defining commands are in fact specific examples of the more general commands \fancyhead and \fancyfoot, where

\lhead{..} is \fancyhead[L]{..} \\
\cfoot{..} is \fancyfoot[C]{..} \\
\rhead{..} is \fancyhead[R]{..}

and so on, with L C R standing for 'left', 'center', 'right'.

For two-sided output with the twoside option, one normally wants the left and right parts to alternate with page number. The easiest way to do this is with

\fancyhead[LE,RO]{Text 1} \fancyhead[LO,RE]{Text 2}

to put the same Text 1 in the left part of even pages, and right part of odd pages, and Text 2 for the other way round. With \fancyhead{}, all head line parts are set to blanks, something that should be done before resetting them explicitly. Similarly \fancyfoot{} sets all foot entries to blank.

The default (two-sided) definitions for the fancy page style are

\fancyhead[EL,OR]{\textsl{\rightmark}} \fancyhead[ER,OL]{\textsl{\leftmark}}

where \textsl and \leftmark contain the automatic texts for the headings page style generated by the \chapter, \section, \subsection commands (Section 3.3.3), while \textsl (Section 4.1.4) sets its argument in a slanted typeface. The user may also make use of these to redefine the head line with automatic texts.

There is also the most general \fancyhf command taking optional arguments [H] and [F] to apply to head or footlines. Thus \fancyhf[HL]{..} is the same as \fancyhead[L]{..}. Clearly, \fancyhf{} resets everything.
In many classes, the first page of a chapter, or the very first page of the document, is switched to plain automatically. If the user wants to change this, he or she must redefine that page style. The fancyhdr package simplifies this task with

\fancypagestyle{plain}{definitions}

where definitions consist of \fancyhead, \fancyfoot, and/or rule redefinitions that are to apply to the revised plain style. In fact, any existing page style can be redefined in this way.

### 3.2.3 Page numbering

The declaration that specifies the style of the page numbering has the form

\pagenumbering{num_style}

The allowed values of num_style are:

- arabic for normal (Arabic) numerals,
- roman for lower case Roman numerals,
- Roman for upper case Roman numerals,
- alph for lower case letters,
- Alph for upper case letters.

The standard value is arabic. This declaration resets the page counter to 1. In order to paginate the foreword of a document with Roman numerals and the rest with Arabic numbers beginning with page 1 for chapter 1, one must declare \pagenumbering{roman} at the start of the foreword and then reset the page numbering with \pagenumbering{arabic} immediately after the first \chapter command. (See Section 3.3.5 for the preferred method.)

Pages may be numbered starting with a value different from 1 by giving the command

\setcounter{page}{page_num}

where page_num is the number to appear on the current page.

**Exercise 3.3:** Expand your exercise text file so that it fills more than one page of output and include the following preamble:

\documentclass{article}
\pagestyle{myheadings} \markright{Exercises}
\pagenumbering{Roman}
\begin{document}
3.2.4 Paragraph formatting

The following parameters affect the appearance of a paragraph and may be given new values with \setlength as explained in Section 8.2:

\parskip
The distance between paragraphs, expressed in units of ex so that it will automatically change with character font size. This should be a rubber length.

\parindent
The amount of indentation for the first line of a paragraph.

\baselinestretch
This is a number that magnifies the normal distance between baselines, the line on which the letters sit. This number is initially 1, for standard line spacing. It may be changed to another number with

\renewcommand{\baselinestretch}{factor}

where factor is any decimal number, such as 1.5 for a 50% increase. This then applies to all font sizes. If this command is given outside the preamble, it does not come into effect until another font size has been selected (Section 4.1.2).

These parameters may be set either in the preamble or anywhere in the text of the document. In the latter case, the changes remain in effect until the next change or until the end of the environment in which they were made (Section 2.3).

To suppress indentation for one paragraph, or to force it where it would otherwise not occur, place

\noindent or \indent

at the beginning of the paragraph to be affected.

Normally, the first paragraph of a section is not indented, not even with \indent. However, by including the package indentfirst one ensures that all paragraphs are indented.

By default, \LaTeX{} indicates paragraphs by indenting the first line. An alternative is without indentation but with extra spacing between paragraphs. One could redefine \parindent and \parskip accordingly, or one could employ one of the oldest and simplest packages dating back to \LaTeX{} 2.09 days: parskip, written by H. Partl. For consistency, this package also makes some changes in the parameters for lists (Section 4.3). One loads this package with

\usepackage{parskip}
There is a recent update to the parskip package by Robin Fairbairns that includes the option parfill, given as

\usepackage[parfill]{parskip}

that avoids ugly-looking rectangular paragraphs by ensuring that there is always space at the end of the last line. This update is dated April 9, 2001, so to be sure that this package version is loaded, add this date as

\usepackage[parfill]{parskip}[2001/04/09]

as explained in Section 3.1.4. A warning will be issued on processing if the actual version of parskip is earlier than this. There will also be warning about the unknown option parfill.

Exercise 3.4: Add the following to the preamble of your exercise file:

\usepackage{parskip}
\renewcommand{\baselinestretch}{1.2}

After processing this exercise, repeat it with another value for the parameter \baselinestretch, say 1.5, in order to get a feeling for how it works. Remove these lines from the exercise file afterwards.

3.2.5 Page format

Each page consists of a head, the body containing the actual text, and a foot. The selection of the page style determines what information is to be found in the head and footlines.

\LaTeX uses default values for the distances between the head, body, and foot, for the upper and left margins, and for the text line width and heights of the head, body, and foot. These formatting lengths are illustrated in Figure 3.1 on the next page. They may be changed by declaring new values for them, preferably in the preamble, with the command \setlength (Section 8.2). For example, give

\setlength{\textwidth}{12.5cm}

to make the text line width to be 12.5 cm.

There is also a parameter \linewidth equal to the text line width in whatever environment one is currently in. This must never be changed, but is used when one needs to know this width.

More detailed diagrams of the page formats for one- and two-column outputs are shown at the end of Appendix H in Figures H.1 and H.2.

You can examine your own page layout with the \texttt{layout} package from the tools collection. Simply issue the command \texttt{layout} and a diagram similar to that in Figure 3.1 will be drawn at that point, together with a list of the current values of the layout parameters. Naturally, you would not do this in the middle of the final version of a document, but only as diagnostic check.
In order to calculate the page layout precisely, one must realize that \TeX measures all distances from a point one inch from the top of the paper and one inch from the left edge. Thus the total left margin is \oddsidemargin plus one inch. The \TeX parameters \paperwidth and \paperheight, which include this extra inch, are given their values by the paper size option in the \documentclass command; they are used internally to calculate the margins so that the text is centered. The user may also take advantage of them for calculations.

With document class book or with the option twoside, the bottom edge of the body will always appear at exactly the same position on every page. In the other classes or options, it will vary slightly. In the first two cases, the constant bottom edge is produced by the internal command \flushbottom, whereas the varying bottom is produced by the command \raggedbottom. The user may apply these declarations to change the behavior of the bottom edge at any time, independent of document class and options.

Exercise 3.5: You can change the page format of your text by altering the above parameters. Add the following to the preamble of your text:

\setlength{\textwidth}{13cm}  \setlength{\textheight}{20.5cm}

The upper and left margins of your output will now seem too small. Select new values for \oddsidemargin and \topmargin to correct this. Note: do not
forget the 1 inch margin at the left and top from which additional margins are measured. You must take this into account when you select \oddsidemargin and \topmargin.

Exercise 3.6: Expand your text so that the output requires more than two full pages with the reduced page format. Add \flushbottom to the preamble and check that the last line of all pages is at exactly the same location.

Exercise 3.7: Remove the command \flushbottom and select the document class \documentclass[twoside]{article}. Now the last lines are at the same location without the \flushbottom command. On the other hand, the left margin of the odd pages probably does not agree with the right margin of the even pages. Adjust the value of \evensidemargin to correct this.

3.2.6 Simplified page formatting

Getting the page layout to be exactly the way you want it can be very tedious. Just centering the text on the page involves a complex set of settings that are not at all intuitive. The \geometry package by Hideo Umeki offers considerable assistance.

With this package, one can easily give values for some of the layout parameters, and the rest will be set automatically, taking into account the total paper size. For example, to set \textwidth to 15 cm and \textheight to 25 cm on A4 paper, one gives

\begin{verbatim}
\usepackage{geometry}
\geometry{a4paper,textwidth=15cm,textheight=25cm}
\end{verbatim}

which will also automatically set \oddsidemargin and \topmargin so that the text is centered horizontally and vertically, including the head and footlines. Or, one can set all the margins to be 1 inch on US letter paper with

\begin{verbatim}
\geometry{letterpaper,margin=1in}
\end{verbatim}

Rather than using the \geometry command, one may also place the parameters as options to \usepackage, for example as

\begin{verbatim}
\usepackage[a4paper,left=3cm,right=2cm]{geometry}
\end{verbatim}

to set the left and right margins to definite values, and \textwidth to what is left over.

In general, all the parameters in Figure 3.1 may be specified by \geometry by giving their names (without the backslash character). However, the package is far more powerful than that. Here we describe the essential features of version 2.3 from 2000/06/28.

- The paper size is either inherited from the \documentclass option, given as a predefined option like a4paper, given explicitly as
paperwidth=\textwidth and paperheight=\textheight, or as papersize={\textwidth,\textheight}.

• By default, the other layout parameters are set so that \textwidth is 80\% of \paperwidth and \textheight + \headheight + \headsep + \footskip is 90\% of \paperheight, centered horizontally and vertically.

• The text width and height may be set explicitly with width=\textwidth and height=\textheight, or with body={width,\textheight}. The margins are then set to center. Here \textwidth means total text width including head and footlines.

• Margins may be explicitly given with left=lmarg and right=rmarg, or with hmargin={lmarg,rmarg}. Similarly with top=tmarg and bottom=bmarg, or vmargin={tmarg,bmarg}. If only one value of a pair is given, the other is set to that same value, unless the corresponding text height or width has been given explicitly. All margins may be set to a common value with margin=marg.

Unlike \LaTeX standard \oddsidemargin and \topmargin, geometry's margins are measured from the edge of the paper, and not from a point 1 inch removed.

• With nohead, nofoot, noheadfoot, one tells geometry not to include the corresponding head or footline in the height calculation. Thus with noheadfoot, the total height is identical to \textheight.

• With includemp, the marginal note parameters \marginparwidth and \marginparsep (Section 4.10.5) are included in the total width calculation, which is then less than \textwidth. With reversemp, the marginal notes appear in the left margin.

• The text width and height may be set to a fraction of the paper size with hscale=\textwidth and vscale=\textheight, or scale={h,v}. With scale=s, both h and v are set to s. For example, \geometry{scale=0.8} sets width and height to 80\% of \paperwidth and \paperheight respectively.

• For two-sided output, add the option twoside. In this case, the values of the left and right margins will switch for even page numbers. In addition, a quantity of 20 pt is added to and subtracted from the left margin of odd and even pages, respectively. This value may be changed with twosideshift=shift, which also sets the twoside option automatically.

• With the verbose option, the calculated values of all the layout parameters are printed to the monitor and to the transcript file.
As you see, the geometry package is an enormous aid to setting the page layout, working fairly intuitively, automatically setting values for the unspecified parameters in a way that is normally what one would want anyway. There are many more aspects explained in the accompanying documentation, but the above synopsis should cover most requirements.

3.2.7 Single and double column pages

The document class option \texttt{twocolumn} sets the entire document in two columns per page. The default is one column per page. Individual pages may be output in one or two columns with the declarations:

\begin{verbatim}
\twocolumn[header text]
\end{verbatim}

Terminates the current page, starting a new one with \texttt{two columns} per page. The optional \texttt{header text} is written at the top of the page in one column with the width of the whole page.

\begin{verbatim}
\onecolumn
\end{verbatim}

Terminates the current two-column page and continues with one column per page.

The option \texttt{twocolumn} automatically changes certain page style parameters, such as indentation, compared with the one-column format. This does not occur with the command \texttt{\twocolumn}. These additional changes must be made with the corresponding \texttt{\setlength} declarations if they are desired. If the bulk of the document is in two-column format, the class option is to be preferred.

An additional page style parameter is \texttt{\columnwidth}, the width of one column of text. For single column text, this is the same as \texttt{\textwidth}, but when \texttt{\twocolumn} has been selected, \LaTeX\ calculates it from the values of \texttt{\textwidth} and \texttt{\columnsep}. The author should never change this parameter, but he or she may make use of it, for example to draw a rule the width of a column of text.

The length \texttt{\linewidth} is even more general, always containing the text line width in the current environment, minipage, parbox. Within a single column, it is the same as \texttt{\columnwidth}. It too may never be changed.

3.2.8 Multicolumn text

\textbf{Package: multicol} The commands \texttt{\twocolumn} and \texttt{\onecolumn} always start a new page, and when two-column text is terminated, the two columns are of unequal length. These problems are solved with the \texttt{multicol} package in the tools collection, written by Frank Mittelbach, which also allows up to 10 columns of text. Once this package has been loaded, one can switch the number of columns in the middle of a page with

\begin{verbatim}
\begin{multicols}{num_cols}[header text][pre_space]
Text set in num_cols columns
\end{multicols}
\end{verbatim}
where the optional header text is written across all the columns before switching to multicolumns.

Some automatic control for page breaking before and after switching to multicolumns is offered by the two lengths \premulticols and \postmulticols: if the remaining space on the current page is less than \premulticols, a new page is started before switching to multicolumns. Similarly, if at the end of the environment, there is less than \postmulticols on the page, a page break is inserted before continuing. The standard values of these lengths may be altered by the user with \setlength, or in the case of \premulticols may be overridden by the second optional argument pre_space.

The lengths \columnsep and \columnseprule that apply to two-column texts are also in effect for the multicols environment, to set the widths of the gap between columns and a possible separating rule, respectively.

There is also a starred version \multicols*\{num_cols\}... \end{multicols*} for which the columns on the last page are not balanced, with all the remaining space put into the final column.

3.3 Parts of the document

Every document is subdivided into chapters, sections, subsections, and so on. There can be an appendix at the end and at the beginning a title page, table of contents, an abstract, etc. LaTeX has a number of markup commands available to indicate these structures. In addition, sequential numbering and sub-numbering of headings take place automatically. Even a table of contents may be produced with a single command.

The effects of some sectioning commands depend on the selected document class and not all commands are available in every class.

3.3.1 Title page

A title page can be produced either unformatted with the environment

\begin{titlepage} Title page text \end{titlepage}

or with the commands

\title{Title text}
\author{Author names and addresses}
\date{Date text}
\maketitle

in the preprogrammed LaTeX style.

In the standard LaTeX layout for the title page, all entries are centered on the lines in which they appear. If the title is too long, it will be broken
How to Write DVI Drivers

Helmut Kopka\thanks{Tel. [+49] 5556--401--451}\\ Max--Planck--Institut f"ur Aeronomie \and
Phillip G. Hardy\thanks{Tel. [+1] 319--824--7134}\\ University of Iowa

January 15, 2004

Figure 3.2: Sample title page and the text that produced it

up automatically. The author may select the break points himself with the \ command, that is, by giving \title{... \ ... \ ...}.

If there are several authors, their names may be separated with \and from one another, such as \author{G. Smith \and J. Jones}. These names will be printed next to each other in one line. The sequence

\author{Author1 \ Institute1 \ Address1 \and Author2 \ Institute2 \ Address2}

separately centers the entries, one per line, in each of the sets Author1, Institute1, Address1 and Author2, Institute2, Address2 and places the two blocks of centered entries beside each other on the title page.

Instead of printing the author names next to each other, one may position them on top of one another by replacing \and with the \ command. In this case, the vertical spacing may be adjusted with an optional length specification [space] following the \.

If the command \date is omitted, the current date is printed automatically below the author entries on the title page. On the other hand, the command \date{Date text} puts the text Date text in place of the current date. Any desired text may be inserted here, including line break commands \ for more than one line of centered text.
The command

\thanks{Footnote text}

may be given at any point in the title, author, or date text. This puts a marker at that point where the command appears and writes footnote text as a footnote on the title page.

The title page is created using the entries in \title, \author, \date, and \thanks when the command

\maketitle

is issued. The title page itself does not possess a page number and the first page of the following document is number 1. (For book, the page numbering is controlled by the special commands in Section 3.3.5.) A separate title page is only produced for document classes book and report. For article, the command \maketitle creates a title heading on the first page using the centered entries from the \title, \author, and, if present, \date and \thanks declarations. If the document class option titlepage has been given, the title appears on a separate page even for the article class.

An example of a title page in the standard \LaTeX\ format is shown in Figure 3.2 on the previous page. Note that the current date appears automatically since the command \date is missing in the definition of the title page. This command may be used to put any desired text in place of the date.

For the unformatted title page produced with the titlepage environment, the commands \title and \author are left out and the entire title page is designed according to the author's specifications within the environment. To this end he or she may make use of all the structuring commands described in Chapter 4. In this case the printing of the title page is implemented at the end of the titlepage environment, so the command \maketitle is also left out.

**Exercise 3.8:** Remove the declarations for changing the page format in Exercises 3.5–3.7. Add to your exercise text a title heading with the title 'Exercises', your name as author, and your address, together with a date entry in the form 'place, date'. To do this, write the following commands after \begin{document}:

\begin{verbatim}
\title{Exercises} \author{Your name} \\ Your address
\date{Your town, \today} \maketitle
\end{verbatim}

Make sure that you have selected document class article. After printing the document, change the document class command to

\begin{verbatim}
\documentclass[titlepage]{article}
\end{verbatim}

to put the title information on to a title page instead of a title heading. Deactivate these commands by putting the comment character % at the beginning of each of the lines. In this way you avoid getting a title page in the following exercises but you can easily reactivate the commands simply by removing the % characters.
3.3.2 Abstract

The abstract is produced with the command

```
\begin{abstract} Text for the abstract \end{abstract}
```

In document class `report`, the abstract appears on a separate page without a page number; in `article`, it comes after the title heading on the first page, unless the document class option `titlepage` has been selected, in which case it is also printed on a separate page. An abstract is not possible in document class `book`.

3.3.3 Sections

The following commands are available for producing automatic, sequential sectioning:

```
\part \chapter \subsection \paragraph
\section \subsubsection \subparagraph
```

With the exception of `\part`, these commands form a sectioning hierarchy. In document classes `book` and `report`, the highest sectioning level is `\chapter`. The chapters are divided into sections using the `\section` command, which are further subdivided by means of `\subsection`, and so on. In document class `article`, the hierarchy begins with `\section` since `\chapter` is not available.

The syntax of all these commands is

```
/sec_command[short title]{title} or
/sec_command*{title}
```

In the first case, the section is given the next number in the sequence, which is then printed together with a heading using the text `title`. The text `short title` becomes the entry in the table of contents (Section 3.4) and the page head (provided that page style `headings` has been selected). If the optional `short title` is omitted, it is set equal to `title`; this is the normal situation unless `title` is too long to serve for the other entries.

In the second (*-form) case, no section number is printed and no entry in the table of contents is made (however, see Section 3.4.3).

The size of the title heading and the depth of the numbering depend on the position of the sectioning command within the hierarchy. For document class `article`, the `\section` command generates a single number (say 7), the `\subsection` command a double number with a period between the two parts (say 7.3), and so on.

In document classes `book` and `report`, the chapter headings are given a single number with the `\chapter` command, the `\section` command creates the double number, and so on. Furthermore, the command `\chapter` always starts a new page and prints `Chapter n` over the chapter title, where
\section{Document Layout and Organization}

\begin{itemize}
\item \textbf{n} is the current chapter number. At this point in the present book, we are in \textit{Chapter 3, Section 3.3, Subsection 3.3.3}.
\item For each sectioning command there is an internal counter that is incremented by one every time that command is called, and reset to zero on every call to the next higher sectioning command. These counters are not altered by the *-forms, a fact that can lead to difficulties if standard and *-forms of the commands are mixed such that the *-forms are higher in the hierarchy than the standard forms. There are no problems, however, if the *-forms are always lower than the standard forms. The sequence
\begin{verbatim}
\section \subsection \subsubsection*
\end{verbatim}
numbers the headings for \texttt{section} and \texttt{subsection} while leaving the headings for \texttt{subsubsection} without any numbering.
\item The sectioning command \texttt{part} is a special case and does not affect the numbering of the other commands.
\item The automatic numbering of sections means that the numbers might not necessarily be known at the time of writing. The author may be writing them out of their final order, or might later introduce new sections or even remove some. If he or she wants to refer to a section number in the text, some mechanism other than typing the number explicitly will be needed. The \LaTeX\ cross-reference system, described in detail in Section 9.2.1, accomplishes this task with the two basic commands
\begin{verbatim}
\label{name} \ref{name}
\end{verbatim}
the first of which assigns a keyword \texttt{name} to the section number, while the second may be used as reference in the text for printing that number. The keyword \texttt{name} may be any combination of letters, numbers, or symbols. For example, in this book the command \texttt{\label{sec:xref}} has been typed in at the start of Section 9.2.1, so that this sentence contains the input text \textit{at the start of Section} \texttt{\ref{sec:xref}}.
\item A second referencing command is \texttt{\pageref} for printing the page number where the corresponding \texttt{\label} is defined.
\item The referencing commands may be used in many other situations for labeling items that are numbered automatically, such as figures, tables, equations.
\end{itemize}

Every sectioning command is assigned a level number such that \texttt{section} is always level 1, \texttt{subsection} level 2,... \texttt{subparagraph} level 5. In document class \texttt{article}, \texttt{part} is level 0 while in \texttt{book} and \texttt{report} classes, \texttt{part} is level −1 and \texttt{chapter} becomes level 0. Section numbering is carried out down to the level given by the number \texttt{secnumdepth}. This limit is set to 2 for \texttt{book} and \texttt{report}, and to 3 for \texttt{article}. This means that for \texttt{book} and \texttt{report}, the section numbering extends only to the level of \texttt{subsection} and for \texttt{article} to \texttt{subsubsection}.

In order to extend (or reduce) the level of the section numbering, it is necessary to change the value of \texttt{secnumdepth}. This is done with the command
\begin{verbatim}
\setcounter{secnumdepth}{num}
\end{verbatim}
(The command \setcounter is explained in Section 8.1.) In article, \textit{num} may take on values from 0 to 5, and in \textit{book} and \textit{report} from −1 to 5.

It is possible to change the initial value of a sectioning command within a document with the command

\begin{verbatim}
\setcounter{sec.name}{num}
\end{verbatim}

where \textit{sec.name} is the name of the sectioning command without the preceding \char\ character. This procedure may be useful when individual sections are to be processed by \LaTeX as single files. For example,

\begin{verbatim}
\setcounter{chapter}{2}
\end{verbatim}

sets the \texttt{\chapter} counter to 2. The counter will be incremented on the next call to \texttt{\chapter} which then produces \textbf{Chapter 3}.

3.3.4 Appendix

An appendix is introduced with the declaration

\begin{verbatim}
\appendix
\end{verbatim}

It has the effect of resetting the section counter for \texttt{article} and the chapter counter for \texttt{book} and \texttt{report} and changing the form of the numbering for these sectioning commands from numerals to capital letters A, B, .... Furthermore, the word ‘Chapter’ is replaced by ‘Appendix’ so that subsequent chapter headings are preceded by ‘Appendix A’, ‘Appendix B’, etc. The numbering of lower sectioning commands contains the letter in place of the chapter number, for example A.2.1.

3.3.5 Book structure

To simplify the structuring of a book, the commands

\begin{verbatim}
\frontmatter
\preface, \table of contents
\mainmatter
\main body of text
\backmatter
\bibliography; index, colophon
\end{verbatim}

are provided in the \texttt{book} class. The \texttt{\frontmatter} command switches page numbering to Roman numerals and suppresses the numbering of chapters; \texttt{\mainmatter} resets the page numbering to 1 with Arabic numbers and reactivates the chapter numbering; this is once again turned off with \texttt{\backmatter}.

\textbf{Exercise 3.9:} Insert at the beginning of your exercise text the command \texttt{\section{Title A}} and at some appropriate place near the middle \texttt{\section{Title B}}. Select some suitable text for Title A and Title B. Insert at appropriate places
some `subsection` commands with reasonable subtitles. Remove the commands included from Exercise 3.3:

\begin{verbatim}
\pagestyle{myheadings} \markright{Exercises}
\pagenumbering{Roman}
\end{verbatim}

and print the results.

**Exercise 3.10:** Include the additional command `\chapter{Chapter title}` with your own appropriate Chapter title before your first `\section` command. Change the document class command to `\documentclass[\textit{twoside}]{\textit{report}}` and call the page style command `\pagestyle{\textit{headings}}` in the preamble. Note the twofold effect of the sectioning commands in the headings and in the page headlines. Compare the results with the table in Section 3.2.1.

**Exercise 3.11:** Change the chapter command to

\begin{verbatim}
\chapter[\textit{Short form}]{Chapter title}
\end{verbatim}

by putting an abbreviated version of Chapter title for Short form. Now the page head contains the shortened title where the full chapter title previously appeared.

### 3.4 Table of contents

#### 3.4.1 Automatic entries

\LaTeX{} can prepare and print a table of contents automatically for the whole document. It will contain the section numbers and corresponding headings as given in the standard form of the sectioning commands, together with the page numbers on which they begin. The sectioning depth to which entries are made in the table of contents can be set in the preamble with the command

\begin{verbatim}
\setcounter{tocdepth}{\textit{num}}
\end{verbatim}

The value `\textit{num}` has exactly the same meaning and effect as it does for the counter `\textit{secnumdepth}` described above, by which the maximum level of automatic subsectioning is fixed. By default, the depth to which entries are included in the table of contents is the same as the standard level to which automatic sectioning is done: that is, to level `\textit{subsection}` for book and report and to level `\textit{subsubsection}` for article.

#### 3.4.2 Printing the table of contents

The table of contents is generated and printed with the command

\begin{verbatim}
\tableofcontents
\end{verbatim}
given at the location where the table of contents is to appear, which is normally after the title page and abstract.

This leads to a paradox, for the information in the table of contents is to be printed near the beginning of the document, information that cannot be known until the end. \LaTeX{} solves this problem as follows: the first time the document is processed, no table of contents can be included but instead \LaTeX{} opens a new file with the same name as the source file but with the extension `.toc`; the entries for the table of contents are written to this file during the rest of the processing.

The next time \LaTeX{} is run on this document, the `\tableofcontents` command causes the `.toc` file to be read and the table of contents is printed. As the processing continues, the `.toc` file is updated in case there have been major changes since the previous run. This means that the table of contents that is printed is always the one corresponding to the previous version of the document. For this reason, it may be necessary to run \LaTeX{} more than once on the final version.

### 3.4.3 Additional entries

The *-form sectioning commands are not entered automatically in the table of contents. To insert them, or any other additional entry, the commands

\begin{verbatim}
\addcontentsline{toc}{sec-name}{entry text}
\addtocontents{toc}{entry text}
\end{verbatim}

may be used.

With the first command, the entries will conform to the format of the table of contents, whereby section headings are indented more than those for chapter but less than those for subsection. This is determined by the value of the argument `sec-name`, which is the same as one of the sectioning commands without the `\` character (for example, `section`). The `entry.text` is inserted in the table of contents along with the page number. This command is most useful to enter unnumbered section headings into the table of contents. For example,

\begin{verbatim}
\section*{Author addresses}
\addcontentsline{toc}{section}{Author addresses}
\end{verbatim}

The `\addtocontents` command puts any desired command or text into the `.toc` file. This could be a formatting command, say `\newpage`, which takes effect when the table of contents is printed.

### 3.4.4 Other lists

In addition to the table of contents, lists of figures and tables can also be generated and printed automatically by \LaTeX{}. The commands to produce these lists are

\begin{verbatim}
\listoffigures reads and/or produces file .lof
\listoftables reads and/or produces file .lot
\end{verbatim}
The entries in these lists are made automatically by the `\caption` command in the `figure` and `table` environments (see Section 7.4). Additional entries are made with the same commands as for the table of contents, the general form of which is

```
\addcontentsline{file}{format}{entry}
\addtocontents{file}{entry}
```

where `file` stands for one of the three types `toc` (table of contents), `lof` (list of figures), or `lot` (list of tables). The argument `format` is one of the sectioning commands for the table of contents, as described above, or `figure` for the list of figures, or `table` for the list of tables. The argument `entry` stands for the text that is to be inserted into the appropriate file.

**Exercise 3.12:** In your exercise file, insert after the deactivated title page commands

```
\pagenumbering{roman}
\tableofcontents \newpage
\pagenumbering{arabic}
```

Process your exercise file twice with \LaTeX and print out the second results. Deactivate the above commands with `%` before doing the next run.
There are a variety of ways to display or emphasize the text: changing font style or font size, centering, indentation, marking the paragraphs, and so on. \LaTeX{} supplies us with commands for the most common forms of display.

Many parts of this chapter violate the concept of logical markup, especially those dealing with selection of fonts. The author should not attempt to decorate the document with arbitrary switches of font size and style, but should pack his or her source text into a structure that indicates its purpose. The exercises in this book are an example of this. Rather than starting each one with the word ‘Exercise’ in bold face followed by an explicit number, and then shifting to a slanted font, we defined an exercise environment to do all that automatically. This not only ensures consistency, it also allows a change of style to be easily implemented, simply by redefining the environment. This is where the typographical font commands come into play. They should not appear in the main text at all, but rather in the preamble as part of the definitions of environments and commands.

On the other hand, many of the topics in this chapter really do involve logical markup, such as the verse, quote, and quotation environments, lists, bibliographies, theorems, and tables.

### 4.1 Changing font

In typography, a set of letters, numbers, and characters of a certain size and appearance is called a font. The standard font in \LaTeX{} for the main body of text is an upright, Roman one of medium weight, in the size specified in the \texttt{\documentclass} statement at the start. The three possible basic sizes are 10, 11, and 12 pt, depending on the size options 10pt (default), 11pt, and 12pt. (Recall, there are 72.27 points per inch or about 28.45 pt per cm.) The parenthesis characters ( ) extend the full height and depth of the font size.
Chapter 4. Displayed Text

The differences in the visual appearance of the three standard sizes are greater than would be expected from the ratios of the numbers:

This is an example of the 10 pt font. ( )
And this is the 11 pt font for comparison. ( )
And finally this is a sample of 12 pt font. ( )

4.1.1 Emphasis

The usual way to emphasize text in a typewritten manuscript is by underlining. The typesetter will transform underlined text into italics for the printed version. Switching from standard to emphasized text is carried out in \LaTeX with the command \texttt{\textbackslash em} or the declaration \texttt{\em}.

The \texttt{\em} declaration functions just as the other font declarations described below: the change of font remains in effect until negated by another appropriate declaration (which can be \texttt{\em} itself), or until the end of the current environment (Section 2.2). An environment may also be created with a pair of curly braces \{\ldots\}. The command \texttt{\emph}, on the other hand, operates only on the text in the following argument. This is easiest way to emphasize short pieces of text, as for example:

This is the easiest way to \texttt{\emph}emphasize\texttt{\emph} short ...

The \texttt{\em} declaration is more appropriate for longer text that is enclosed in an environment, named or nameless.

...enclosed in an environment, {\em named or nameless.}

Note carefully the difference between the declaration that remains in effect until the local environment is ended with the closing curly brace, and the command that operates on an argument enclosed in curly braces. Another more subtle difference is that the command \texttt{\emph} automatically inserts extra spacing at the end if necessary, the so-called italic correction, to improve the appearance at the interface between sloping and upright fonts.

Both the declaration and the command switch to an emphasizing font. That means, if the current font is upright it switches to italics, whereas if the text is already slanted, an upright font is selected.

Nested emphasis is possible and is simple to understand:

The \texttt{\emph{first}}, \texttt{second}, and \texttt{\emph{third font switch}}
The \texttt{\{\em first, \{\em second, and \{\em third font switch\}}} both produce 'The first, second, and third font switch'.

4.1.2 Choice of font size

The following declarations are available in \LaTeX for changing the font size:
all of which are relative to the standard size selected in the document class option. In this book, the standard size is 10 pt, which is then the size selected with \texttt{\normalsize}.

The font size declarations behave as all other declarations: they make an immediate change that remains in effect until counteracted by another size declaration, or until the current environment comes to an end. If issued within curly braces \{..\}, the effect of the declaration extends only to the closing brace, as in a nameless environment:

\begin{verbatim}
 normal \{\texttt{\large large \Large larger}\} normal again
 normal large large normal again
\end{verbatim}

Changing the font size with one of the above commands also automatically changes the interline spacing. For every font size, there is a corresponding natural line spacing \texttt{\baselineskip}. This may be altered at any time. If the natural line spacing is 12 pt, the command \texttt{\setlength{\baselineskip}{15pt}} will increase it to 15 pt.

The value of \texttt{\baselineskip} that is effective at the end of the paragraph is used to make up the whole paragraph. This means that if there are several changes to \texttt{\baselineskip} within a paragraph, only the last value given will be taken into account.

With every change in font size, \texttt{\baselineskip} is reset to its natural value for that size. Any previous setting with \texttt{\setlength} will be nullified.

In order to create a change in the line spacing that is valid for all font sizes, one must make use of the factor \texttt{\baselinestretch}, which has a normal value of 1. The true interline spacing is really

\[
\texttt{\baselinestretch} \times \texttt{\baselineskip}
\]

which maintains the same relative spacing for all font sizes. The user may change this spacing at any time with:

\[
\texttt{\renewcommand{\baselinestretch}{factor}}
\]

where \texttt{factor} is any decimal number. A value of 1.5 increases the interline spacing (baseline to baseline) by 50% over its natural size for all font sizes.

The new value of \texttt{\baselinestretch} does not take effect until the next change in font size. In order to implement a new value in the current font size, it is necessary to switch to another size and back again immediately. If the present font size is \texttt{\normalsize}, the sequence

\[
\texttt{\small \normalsize}
\]

will do the trick. Any size command may be used in place of \texttt{\small}.  
4.1.3 Font attributes

The size of a font is only one of several attributes that may be used to describe it. With the New Font Selection Scheme (NFSS), which was introduced as part of \LaTeX, it is possible to select fonts strictly by these attributes, as described in Appendix A. However, for normal usage, there are some declarations and corresponding commands to simplify this procedure.

For the Computer Modern fonts provided with \TeX{} and \LaTeX{}, the following attributes and values exist:

**Family:** for the general overall style. Traditional typographical families have names like *Baskerville*, *Bodoni*, *Times Roman*, *Helvetica*, and so on. The standard \LaTeX{} installation provides three families with declarations

- \texttt{\textbackslash rmfamily} to switch (back) to a Roman font;
- \texttt{\textbackslash tffamily} to switch to a typewriter font;
- \texttt{\textbackslash sffamily} to select a sans serif font.

**Shape:** for the form of the font. The shape declarations available with the standard installation are

- \texttt{\textbackslash upshape} to switch (back) to an upright font;
- \texttt{\textbackslash itshape} to select an *italic* shape;
- \texttt{\textbackslash slshape} to choose a font that is *slanted*;
- \texttt{\textbackslash scshape} to switch to CAPS AND SMALL CAPS.

**Series:** for the width and/or weight (boldness) of the font. The declarations possible are

- \texttt{\textbackslash mdseries} to switch (back) to medium weight;
- \texttt{\textbackslash bfseries} to select a *bold face* font.

These do not exhaust all the possible attribute settings, but they do cover the most standard ones, especially for the Computer Modern fonts. For other fonts, especially PostScript ones, additional attribute values exist. See Section A.1 for more details.

These declarations are used just like any others, normally enclosed in a pair of curly braces \{\ldots\}, such as \{\texttt{\textbackslash scshape Romeo and Juliet}\} producing ROMEO AND JULIET. For longer sections of text, an environment is preferable:

\begin{fontstyle} \ldots text in new font \ldots \end{fontstyle}

This keeps better track of the beginning and end of the switch-over. For \texttt{font-style}, any of the above font commands may be used, leaving off the initial \texttt{\textbackslash} character.

Since changing any one attribute leaves the others as they were, all possible combinations may be obtained. (However, this does not mean
that a font exists for each possible combination; if not, a substitution will be made.) If we select first a bold series with \bfseries, and then a slanted shape with \slshape, we obtain a bold, slanted font.

\begin{verbatim}
 normal and {\bfseries bold and
 \slshape slanted} and back} again.
\end{verbatim}

produces: normal and \bfseries bold and \slshape slanted and back \textbackslash again.

Finally, the declaration \normalfont resets all the attributes (except size) back to their defaults: Roman, upright, medium weight. It is often useful to issue this command just to be sure of the font in effect.

\subsection{4.1.4 Font commands}

For each of the font declarations listed above, there is a corresponding \textit{font command} that sets its argument in a font with the specified attribute.

Family:    \textrm{\texttt{\textsf{\textnormal}}} \texttt{text} \texttt{texttt\{text\} \texttt{\textsf{text}}
Shape:     \textup{\textit{\textsl{\textsc{\textnormal}}} \texttt{textup\{text\} \texttt{\textit{\textsl{\textsc{text}}}}
Series:    \textmd{\textbf{\textsl{\textsc{\textnormal}}} \texttt{textmd\{text\} \texttt{\textbf{\textsl{\textsc{text}}}}
Default:   \textnormal{\texttt{\textsf{\textnormal}}} \texttt{textnormal\{text\}}
Emphasis:  \texttt{\textbf{\textit{\textnormal}}} \texttt{\textbf{\textit{\textnormal}}}

Note that the \texttt{\textbf{\textit{\textnormal}}} command is included here, corresponding to the declaration \texttt{\textnormal}. The argument of \texttt{\textnormal} is set in the standard font selected with \texttt{\normalfont}.

The use of such commands to change the font for short pieces of text, or single words, is much more logical than placing a declaration inside an implied environment. The previous example now becomes

\begin{verbatim}
 normal and \textbf{bold and \textsl{slanted} and back} again.
\end{verbatim}

As for the \texttt{\textbf{\textit{\textnormal}}} command, these font commands automatically add any necessary \textit{italic correction} between upright and slanted/italic fonts.

The old two-letter \TeX declarations such as \texttt{\\tt} and \texttt{\\bf}, which were part of \LaTeX 2.09, are still available but are now considered obsolete and should be avoided. They are listed for reference in Appendix F.

\subsection{4.1.5 Additional fonts}

It is likely that your computing center or your \TeX installation has even more fonts and sizes than those listed above. If so, they may be made available for use within a \LaTeX document either by referring to them by name, or by their attributes, if they have been set up for NFSS.

To load a new font explicitly by name, the command
\newfont{\fnt}{name scaled factor} or 
\newfont{\fnt}{name at size}

is given, which assigns the font to the new command named \fnt. In the first case, factor is a number 1000 times the scaling factor that is to be used to magnify or reduce the font from its basic or design size. In the second case, the font is scaled to be of the size specified. To install a slanted, sans serif font of size 20.74 pt, as \sss, we load cmssi17 at 20.74pt with

\newfont{\sss}{cmssi17 at 20.74pt}

Now the declaration \sss switches directly to this font but without altering the baseline separation.

Alternatively, the new font declaration can be made by attributes with (see Section A.3.2)

\DeclareFixedFont{\sss}{OT1}{cmss}{m}{sl}{20.74}

Indeed, if one wants to use the current encoding and \sffamily without knowing what they are, or without worrying so precisely what size must be stated, it is also possible to give

\DeclareFixedFont{\sss}{\encodingdefault}{\sfdefault} {m}{sl}{20}

(The defaults are explained in Section A.3.1.)

### 4.1.6 Character sets and symbols

The individual character sets are each stored in their own files. The names of the 75 standard \TeX fonts are listed in Section G.2.2, where many of them are also printed out.

Each symbol within a character set is addressed by means of a number between 0 and 127 (or 255). The command

\symbol{num}

will produce that symbol with the internal identification number num in the current font. The symbol 62 in the present font has the internal number 62 and can be printed with the command \symbol{62}. The identification number may also be given as an octal (prefix ’) or hexadecimal (prefix ”) number. Thus the symbol commands \symbol{28}, \symbol{34}, and \symbol{1C} are all identical, producing ‘ø’.

The \symbol command may also be used to generate symbols for which no other command has been defined: for example, \ttfamily\symbol{40} \symbol{42} \symbol{134} produces " in typewriter font.

Section G.2.2 presents the assignments of the identification numbers with the characters for the different symbol families.
4.2 Centering and indenting

4.2.1 Centered text

The environment

\begin{center}
line 1 \line 2 \ldots line n \end{center}

centers the sections of text that are separated by the \line command. (An optional additional line spacing may be inserted with \[len].) If the text is too long for one line, it is split over several lines using uniform word spacing, filling the whole line width as best it can, except for the last line. Word division does not occur.

Within an environment, the command \centering may be used to center the following text, again with \line as the line divider. The effect of this declaration lasts until the end of that environment.

A single line may be centered by typing its text as the argument of the \TeX command \centerline{text}.

4.2.2 One-sided justification

The environments

\begin{flushleft}
line 1 \line 2 \ldots line 2 \end{flushleft}
\begin{flushright}
line 1 \line 2 \ldots line 2 \end{flushright}

produce text that is left (flushleft) or right (flushright) justified. If a section of text does not fit on to one line, it is spread over several with fixed word spacing, the same as for the \center environment. Again, word division does not occur.

The same results may be produced within an environment with the declarations

\raggedright replacing the flushleft environment, and
\raggedleft replacing the flushright environment.

4.2.3 Two-sided indentation

A section of text may be displayed by indenting it by an equal amount on both sides, with the environments

\begin{quote}
text \end{quote}
\begin{quotation}
text \end{quotation}

Additional vertical spacing is inserted above and below the displayed text to separate it visually from the normal text.

The text to be displayed may be of any length; it can be part of a sentence, a whole paragraph, or several paragraphs.
Paragraphs are separated as usual with an empty line, although no empty lines are needed at the beginning and end of the displayed text since additional vertical spacing is inserted here anyway.

The difference between the above two forms is thus:

In the quotation environment, paragraphs are marked by extra indentation of the first line, whereas in the quote environment, they are indicated with more vertical spacing between them.

The present text is produced within the quotation environment, while the sample above was done with the quote environment.

The quotation environment is only really meaningful when the regular text makes use of first-line indentation to show off new paragraphs.

4.2.4 Verse indentations

For indenting rhymes, poetry, verses, etc. on both sides, the environment

\begin{verse} poem \end{verse}

is more appropriate.

Stanzas are separated by blank lines
while the individual lines of the stanza are divided by the \ command.

If a line is too long for the reduced text width, it will be left and right justified and continued on the next line, which is indented even further.

The above indenting schemes may be nested inside one another. Within a quote environment there may be another quote, quotation, or verse environment. Each time, additional indentations are created on both sides of the text and vertical spacing is added above and below; these quantities however decrease as the depth of nesting increases. A maximum of six such nestings is allowed.

Exercise 4.1: Put some appropriate sections of text in your exercise file into the quote and quotation environments, that is, enclose these sections within

\begin{quote} . . . . . . \end{quote} or
\begin{quotation} . . . . . \end{quotation}

commands.
Exercise 4.2: Make up a new file with the name poem.tex and type your favorite poem in the verse environment. Select 12pt as the standard font size and italic as the typeface. Put the title of the poem before the verse environment in a larger bold typeface, such as \Large\bfseries. Include the name of the poet right justified.
Note: remember that you may include declarations to change the font style or size within an environment and that these remain in effect only until the end of that environment.

Exercise 4.3: Make up another file with the name title.tex. Do you recall the titlepage environment for producing a free-form title page? If not, refer to Section 3.3.1. Create a title page with this environment using font sizes and styles of your choice, centering all the entries.
Note: within the titlepage environment you may of course make use of the center environment, but it is also sufficient to give the \centering declaration instead, since this will remain in effect only until the end of the titlepage environment.

Choose the individual line spacings with the command \[\text{len}\] using an appropriate value for the spacing len. Remember that vertical spacing before the first line of text must be entered with the \^form of the command \vspace[\text{len}] (see Section 2.7.3).

Experiment with different font sizes and styles for the various parts of the title page, such as title, author's name, address, until you are satisfied with the results.

Compare your own title page with that of Exercise 3.8. If your creation appeals to you more, include it in your standard exercise file by replacing the commands \title, \author, \date, and \maketitle with the titlepage environment and your own entries.

4.3 Lists

There are three environments available for producing formatted lists:

\begin{itemize} \item list text \end{itemize}
\begin{enumerate} \item list text \end{enumerate}
\begin{description} \item list text \end{description}

In each of these environments, the list text is indented from the left margin and a label, or marker, is included. What type of label is used depends on the selected list environment. The command to produce the label is \item.

4.3.1 Sample itemize

- The individual entries are indicated with a black dot, a so-called bullet, as the label.
Chapter 4. Displayed Text

- The text in the entries may be of any length. The label appears at the beginning of the first line of text.
- Successive entries are separated from one another by additional vertical spacing.

The above text was produced as follows:
\begin{itemize}
\item The individual entries are indicated with a black dot, a so-called \texttt{bullet}, as the label.
\item The text in the entries may be of any length. The label appears at the beginning of the first line of text.
\item Successive entries are separated from one another by additional vertical spacing.
\end{itemize}

4.3.2 Sample \texttt{enumerate}

1. The labels consist of sequential numbers.
2. The numbering starts at 1 with every call to the \texttt{enumerate} environment.

The above example was generated with the following text:
\begin{enumerate}
\item The labels consist of sequential numbers.
\item The numbering starts at 1 with every call to the \texttt{enumerate} environment.
\end{enumerate}

4.3.3 Sample \texttt{description}

\texttt{purpose} This environment is appropriate when a number of words or expressions are to be defined.

\texttt{example} A keyword is used as the label and the entry contains a clarification or explanation.

\texttt{other uses} It may also be used as an author list in a bibliography.

The above sample was created using the following:
\begin{description}
\item[\texttt{purpose}] This environment is appropriate when a number of words or expressions are to be defined.
\item[\texttt{example}] A keyword is used as the label and the entry contains a clarification or explanation.
\item[\texttt{other uses}] It may also be used as an author list in a bibliography.
\end{description}
The `\item[option]` command contains an optional argument that appears in bold face as the label.

### 4.3.4 Nested lists

The above lists may be included within one another, either mixed or of one type, to a depth of four levels. The type of label used depends on the depth of the nesting. The indentation is always relative to the left margin of the enclosing list. A fourfold nesting of the `itemize` environment appears as follows:

- The label for the first level is a black dot, a *bullet*.
  - That of the second level is a long dash.
    * That of the third level is an asterisk.
      · And the label for the fourth level is a simple dot.
      · At the same time, the vertical spacing is decreased with increasing depth.
    * Back to the third level.
  - Back to the second level.
- And here we are at the first level of `itemize` once again.

Similarly for the `enumerate` environment, where the style of the numbering changes with the nesting level:

1. The numbering at the first level is with Arabic numerals followed by a period.
   (a) At the second level, it is with lower case letters in parentheses.
      i. The third level is numbered with lower case Roman numerals with a period.
      A. At the fourth level, capital letters are used.
      B. The label style can be changed, as described in the next section.
      ii. Back to the third level.
   (b) Back to the second level.
2. And the first level of `enumerate` again.

An example of a nested list with mixed types:

- The `itemize` label at the first level is a bullet.
  1. The numbering is with Arabic numerals since this is the first level of the `enumerate` environment.
Chapter 4. Displayed Text

- This is the third level of the nesting, but the second itemize level.
  (a) And this is the fourth level of the overall nesting, but only the second of the enumerate environment.
  (b) Thus the numbering is with lower case letters in parentheses.
- The label at this level is a long dash.

2. Every list should contain at least two points.

  • Blank lines ahead of an \item command have no effect.

  The above mixed list was produced with the following text:

  \begin{itemize}
  \item The \texttt{itemize} label at the first level is a ...
  \begin{enumerate}
  \item The numbering is with Arabic numerals since this ...
  \begin{itemize}
  \item This is the third level of the nesting, but the ...
  \begin{enumerate}
  \item And this is the fourth level of the overall ...
  \item Thus the numbering is with lower case letters ...
  \end{enumerate}
  \item The label at this level is a long dash.
  \end{enumerate}
  \item Every list should contain at least two points.
  \end{itemize}

  \item Blank lines ahead of an \verb+\item+ command ...
  \end{itemize}

Exercise 4.4: Produce a nested list using the itemize and enumerate environments as in the above example, but with a different sequence of these commands.

Exercise 4.5: Prepare a list of conference participants with their place of residence using the description environment, where the name of the participant appears as the argument in the \item command.

Note: for all three types of lists, any text before the first \item command will yield an error message on processing.

4.3.5 Changing label style

The labels, or markers, used in the itemize and enumerate environments can be easily changed by means of the optional argument in the \item command. With \item[+] the label becomes +, and with \item[2.1:] it is 2.1:. The optional argument takes precedence over the standard
label. For the `enumerate` environment, this means that the corresponding counter is *not* automatically incremented and the user must do the numbering manually.

The optional label appears right justified within the area reserved for the label. The width of this area is the amount of indentation at that level less the separation between label and text; this means that the left edge of the label area is flush with the left margin of the enclosing level.

It is also possible to change the standard labels for all or part of the document. The labels are generated with the internal commands

\labelitemi, \labelitemii, \labelitemiii, \labelitemiv
\labelenumi, \labelenumii, \labelenumiii, \labelenumiv

The endings i, ii, iii, and iv refer to the four possible levels.

These commands may be altered with \renewcommand. For example, to change the label of the third level of the `itemize` environment from ∗ to +, give

\renewcommand{\labelitemiii}{+}

Similarly the standard labels for the `enumerate` environment may be changed. However, here there is an additional complication that there is a counter for each `enumerate` level, named `enumi`, `enumii`, `enumiii`, and `enumiv`. As explained in Section 8.1.4, the value of a counter can be printed using one of the commands `\arabic`, `\roman`, `\Roman`, `\alph`, or `\Alph`, where the style of each command should be obvious from its name. That is, `\Roman{xyz}` prints the current value of the counter `xyz` in upper case Roman numerals, whereas `\alph{xyz}` prints it as a lower case letter (with `a` corresponding to 1 and `z` to 26).

These counters, together with the counter style commands, must be used in the redefinitions of the label commands. For example, to change the second-level label to Arabic numerals followed by ‘.‘, it is necessary to give

\renewcommand{\labelenumii}{\arabic{enumii}.}

which redefines `\labelenumii` to the value of counter `enumii` printed in Arabic, plus the characters ‘.’. In this way, all the numbering levels may be changed. It is even possible to include more than one counter:

\renewcommand{\labelenumii}{\Alph{enumi}.\arabic{enumii}}

which will produce for every call to `\item` at level two the value of the counter `enumi` as a capital letter followed by the value of counter `enumii` as a number: that is, in the form A.1, A.2, ..., B.1, B.2, ... and so on.

If the new standard labels are to apply to the whole document, the redefining commands should be included in the preamble. Otherwise, they are valid only within the environment in which they appear.
Chapter 4. Displayed Text

Exercise 4.6: Change the standard labels for the \texttt{itemize} environment into a long dash — (written \texttt{----}) for the first level, to a medium dash - (--) for the second level, and to a hyphen - for the third level.

Exercise 4.7: Change the standard labels for the \texttt{enumerate} environment for the first level to (I), (II), \ldots, and for the second level to the Roman numerals of the first level followed by the number for the second level in the form I–1; I–2; \ldots, II–1; II–2; \ldots.

An alternative method of customizing the enumeration labels is with the \texttt{enumerate} package in the tools collection (Section B.5.4). Once this package has been loaded, the \texttt{enumerate} environment accepts an optional argument specifying the text of the label. The characters A a I i 1 represent the number in alphabetical, Roman, Arabic styles. If these characters appear elsewhere in the label text, they must be in \texttt{\{\}}. For example,

\begin{verbatim}
\begin{enumerate}
[\{Case\} A]
\item Witness tells the truth \Rightarrow \text{Case A Witness tells the truth}
\item Witness is lying \Rightarrow \text{Case B Witness is lying}
\end{enumerate}
\end{verbatim}

4.4 Generalized lists

Lists such as those in the three environments \texttt{itemize}, \texttt{enumerate}, and \texttt{description} can be formed in a quite general way. The type of label and its width, the depth of indentation, spacings for paragraphs and labels, and so on, may be wholly or partially set by the user by means of the \texttt{list} environment:

\begin{verbatim}
\begin{list}{\texttt{stnd_lbl}}{\texttt{list_decl}} item_list \end{list}
\end{verbatim}

Here \texttt{item_list} consists of the text for the listed entries, each of which begins with an \texttt{item} command that generates the corresponding label.

The \texttt{stnd_lbl} contains the definition of the label to be produced by the \texttt{item} command when the optional argument is missing (see below).

The list parameters described in Section 4.4.2 are set by \texttt{list_decl} to whatever new values the user wishes.

4.4.1 Standard label

The first argument in the \texttt{list} environment defines the \texttt{stnd_lbl}, that is, the label that is produced by the \texttt{item} command when it appears without an argument. In the case of an unchanging label, such as for the \texttt{itemize} environment, this is simply the desired symbol. If this is to be a mathematical symbol, it must be given as $\texttt{symbol\_name}$, enclosed in $\texttt{\$} signs. For example, to select $\Rightarrow$ as the label, \texttt{stnd_lbl} must be defined to be $\texttt{\$\backslash Rightarrow\$}$. 
However, the label is often required to contain a sequential numeration. For this purpose, a counter must be created with the \newcounter{\textit{name}} command, where name is its designation. This command must appear before the first application of the counter in a list environment. Suppose a counter named \textit{marker} has been defined for this use, then the argument \textit{stdLbl} could be any of the commands for printing counters described in Section 4.3.5: for example, \textit{arabic{\textit{marker}}} produces a running Arabic number.

Even more complex labels can be made up in this way. If the sequential labels are to be A–I, A–II, . . . , \textit{stdLbl} is set to A–\roman{\textit{marker}}.

Before a counter can function properly within the standard label, it must be associated with that list by including in the list decl the command \usecounter{\textit{counter}}, where \textit{counter} is the name of the counter to be assigned (\textit{marker} in the above example).

The standard label is actually generated by the command \makelabel{\textit{label}}, which is called by the \textit{item} command. The user can redefine \makelabel with the aid of the \renewcommand in the list declaration:

\renewcommand{\makelabel}{\textit{new definition}}

If the standard label is defined in this manner, the corresponding entry in the list environment is left blank. This is because \makelabel is the more general command and overrides the other definition.

### 4.4.2 List style parameters

There are a number of style parameters used for formatting lists that are set by \LaTeX{} to certain standard values. These values may be altered by the user in the list decl for that particular list. The assignment is made in the usual way with the \setlength command. However, if the assignment is made outside the list environment, in most cases it will simply be ignored. This is because there are preset default values for each parameter at each level that can only be overridden by list decl.

The style parameters are listed below and are also illustrated in Figure 4.1 on the next page, which is based on one taken from Lamport (1985, 1994).

\texttt{\topsep} is the vertical spacing in addition to \texttt{\parskip} that is inserted between the list and the enclosing text above and below. Its default value is set at each list level and cannot be globally redefined outside the list decl.

\texttt{\partopsep} is the vertical spacing in addition to \texttt{\topsep} + \texttt{\parskip} that is inserted above and below the list when a blank line precedes the first or follows the last \texttt{item} entry. It may be redefined globally, but only for the first and second levels.
\itemsep is the vertical spacing in addition to \parskip that is inserted between two \item entries. As for \topsep and \parskip, its default value is reset at each level and cannot be globally changed.

\leftmargin is the distance from the left edge of the current environment to the left margin of the list text. There are default values for it at each level that may be globally redefined, as described in Section 4.4.6.

\rightmargin is the distance from the right edge of the current environment to the right margin of the list text. Its standard value is 0 pt, which can only be altered in list_decl.

\listparindent is the indentation depth of the first line of a paragraph within an \item with respect to the left margin of the list text. It is

\begin{figure}
\centering
\begin{tikzpicture}
  \node (item1) {Item 1 \textbf{Paragraph 1}};
  \node (item2) [below of=item1] {Item 2};
  \node (label1) [left of=item1] {Label};
  \node (label2) [left of=item2] {Label};
  \draw[->] (item1) -- (label1) node [midway, above] {\labelwidth};
  \draw[->] (item1) -- (leftmargin) node [midway, above] {\leftmargin};
  \draw[->] (item1) -- (itemindent) node [midway, above] {\itemindent};
  \draw[->] (item1) -- (itemsep) node [midway, above] {\itemsep};
  \draw[->] (item1) -- (parsep) node [midway, above] {\parsep};
  \draw[->] (item1) -- (rightmargin) node [midway, above] {\rightmargin};
  \draw[->] (item2) -- (itemsep) node [midway, above] {\itemsep};
  \draw[->] (item2) -- (parsep) node [midway, above] {\parsep};
  \draw[->] (item2) -- (rightmargin) node [midway, above] {\rightmargin};
  \draw[->] (item1) -- (item2) node [midway, below] {\itemsep + \parsep [+ \partopsep]};
\end{tikzpicture}
\caption{The list parameters}
\end{figure}
normally set to 0 pt so that no indentation occurs. This can only be changed in \texttt{list.decl}.

\texttt{\textbackslash labelwidth}

is the width of the box reserved for the label. The text of the label is printed right justified within this space. A new default value may be set globally which then applies to all list levels.

\texttt{\textbackslash labelsep}

is the spacing between the label box and the list text. A new value may be assigned globally, but it is only effective at the first level.

\texttt{\textbackslash itemindent}

is the distance by which the label and the first line of text in an \texttt{\textbackslash item} are indented to the right. It is normally set to 0 pt and so has no effect. This value can only be redefined in \texttt{list.decl}.

When changing the vertical spacings from their standard values, it is recommended that a rubber length (Section 2.4.2) be used.

The label created by the \texttt{\textbackslash item} command normally appears right justified within a box of width \texttt{\textbackslash labelwidth}. It is possible to make it left justified, as in the following list of parameters, by putting \texttt{\textbackslash hfill} at the end of the definition of the standard label or in the \texttt{\textbackslash makelabel} command.

### 4.4.3 Example of a user's list

List of Figures:

**Figure 1:** Page format with head, body, and foot, showing the meaning of the various elements involved.

**Figure 2:** Format of a general list showing its elements.

**Figure 3:** A demonstration of some of the possibilities for drawing pictures with \LaTeX.

This list was produced with the following input:

\texttt{\newcounter{fig}}
\texttt{\begin{list}{\bfseries$\upshape$\arabic{fig}:}}
\texttt{\{}\texttt{\usecounter{fig}}
\texttt{\setlength{\textwidth}{2cm}\setlength{\leftmargin}{2.6cm}\setlength{\labelsep}{0.5cm}\setlength{\rightmargin}{1cm}\setlength{\parsep}{0.5ex plus0.2ex minus0.1ex}\setlength{\itemsep}{0ex plus0.2ex} \texttt{\slshape}}
\texttt{\item Page format with head, body, and foot, showing the meaning of the various elements involved.}
\texttt{\item Format of a general list showing its elements.}
\texttt{\item A demonstration of some of the possibilities for drawing pictures with \LaTeX.}
\texttt{\end{list}}
The command `\newcounter{fig}` sets up the counter `fig`. The standard label is defined to be the word `Figure` in upright, bold face, followed by the running Arabic number, terminated by `:`. This label is printed for each `\item` command.

The list declaration contains `\usecounter{fig}` as its first command, which makes the counter `fig` operational within the list. The width of the label box (`\labelwidth`) is set to 2.0 cm, the left margin of the list text (`\leftmargin`) to 2.6 cm, the distance between the label and the text (`\labelsep`) to 0.5 cm, and the right edge of the list (`\rightmargin`) is set to be 1 cm from that of the enclosing text.

The vertical spacing between paragraphs within an item (`\parsep`) is 0.5 ex but can be stretched an extra 0.2 ex or shrunk by 0.1 ex. The additional spacing between items (`\itemsep`) is 0 ex, stretchable to 0.2 ex.

Standard values are used for all the other list parameters. The last command in the list declaration is `\slshape`, which sets the list text in a slanted typeface.

**Note:** If `\upshape` were not given in the label definition, the text of each `\item` would also be slanted, as `Figure 1:`.

### 4.4.4 Lists as new environments

If a particular type of list is employed several times within a document, it can become tiresome typing the same `stnd_lbl` and `list_decl` into the `list` environment every time. \LaTeX{} offers the possibility of defining a given list as an environment under its own name. This is achieved by means of the `\newenvironment` command.

For example, the list in the above example can be stored so that it may be called at any time with the name `figlist`:

```latex
\newenvironment{figlist}{\begin{list}
  {\bfseries\upshape Figure \arabic{fig}:
  \usecounter{fig}\ldots
}{\end{list}}
```

It can then be called with

```latex
\begin{figlist} item_list \end{figlist}
```

so that it behaves as a predefined list environment.

**Exercise 4.8:** Define a new environment with the name `sample` that produces a list in which every call to `\item` prints labels `Sample A`, `Sample B`, and so on. The labels are to be left justified within a box of width 20 mm, and the distance between the label box and the item text is to be 2 mm, with a total left margin of 22 mm. The right edge of the text is to be moved in 5 mm from that of the enclosing text. The extra vertical spacing between two items is to be 1 ex plus 0.5 ex minus 0.4 ex in addition to the normal paragraph spacing. Secondary paragraphs within an
item are to be indented by 1em. The normal paragraph separation should be 0ex, expandable to 0.5ex.

\LaTeX{} itself makes frequent use of the list environment to define a number of other structures. For example, the quote environment is defined as

\begin{verbatim}
\newenvironment{quote}{\begin{list}{}{\setlength{\rightmargin}{\leftmargin}}\
\item[]}{\end{list}}
\end{verbatim}

This environment is thus a list in which the value of \rightmargin is set to the current value of \leftmargin which has a default value of 2.5 em. The list itself consists of a single \item call with an empty label, a call that is automatically included in the definition of quote with the entry \item[].

In the same way, \LaTeX{} defines the quotation and verse environments internally as special list environments. The left margins and the vertical spacings around the structures are left as the standard values for the list environment, and are therefore changed only when the standard values themselves are altered.

Finally, as an example of a possible user-defined special list we offer

\begin{verbatim}
\newenvironment{lquote}{\begin{list}{}{\item[]}}{\end{list}}
\end{verbatim}

which creates an lquote environment that does nothing more than indent its enclosed text by the amount \leftmargin, with the right edge flush with that of the normal text, since \rightmargin has the standard value of 0 pt.

4.4.5 Trivial lists

\LaTeX{} also contains a trivlist environment, with syntax

\begin{verbatim}
\begin{trivlist} enclosed text \end{trivlist}
\end{verbatim}

in which the arguments \texttt{stdlbl} and \texttt{listdecl} are omitted. This is the same as a list environment for which the label is empty, \leftmargin, \labelwidth, and \itemindent are all assigned the value 0 pt, while \listparindent is set equal to \parindent and \parskip to \parskip.

\LaTeX{} uses this environment to create further structures. For example, the call to the \texttt{center} environment generates internally the sequence

\begin{verbatim}
\begin{trivlist} \centering \item[] enclosed text \end{trivlist}
\end{verbatim}

The environments \texttt{flushleft} and \texttt{flushright} are similarly defined.

4.4.6 Nested lists

Lists can be nested within one another with the list environments \texttt{itemize}, \texttt{enumerate}, and \texttt{description}, to a maximum depth of six. At each level, the new left margin is indented by the amount \leftmargin relative to the margin of the next higher one.

As mentioned earlier, it is only possible to change the standard values of a limited number of the list parameters with declarations in the preamble. One exception is the indentations of the left margins for the different nesting levels.
These are set internally by the parameters \leftmargini, where $n$ stands for i, ii, iii, iv, v, or vi. These values can be changed by the user; for example, by declaring \setlength{\leftmargini}{12mm}, the left margin of the fourth-level list is shifted 12 mm from that of the third. These declarations must be made outside of the list environments and not in the \textit{list.defl}.

At each level of list nesting, the internal macro \@listn ($n$ being i to vi) is called. This sets the value of \leftmargin equal to that of the corresponding \leftmargini, unless \leftmargin is explicitly declared in the list environment. That is, there does not exist a single standard value for \leftmargin externally, but rather six different ones. The parameter \leftmargin has meaning only within a list environment.

4.5 Theorem-like declarations

In scientific literature one often has text structures like

\textbf{Theorem 1 (Balzano–Weierstrass)} Every infinite set of bounded points possesses at least one maximum point.

or

\textbf{Axiom 4.1} The natural numbers form a set $S$ of distinct elements. For any two elements $a$, $b$, they are either identical, $a = b$, or different from one another, $a \neq b$.

Similar structures frequently appear with names such as \textit{Definition, Corollary, Declaration, Lemma}, instead of \textit{Theorem} or \textit{Axiom}. What they have in common is that a keyword and a running number are printed in \textbf{bold face} and the corresponding text in \textit{italic}.

Of course, these could be generated by the user by explicitly giving the type styles and appropriate number, but if a new structure of that type is later inserted in the middle of the text, the user would have the tedious job of renumbering all the following occurrences. With the command

\begin{verbatim}
\newtheorem{struct_type}{struct_title}[in_counter]
\end{verbatim}

$\LaTeX$ will keep track of the numbering automatically. Here \textit{struct_type} is the user's arbitrary designation for the structure, while \textit{struct_title} is the word that is printed in bold face followed by the running number (for example, \textit{Theorem}). If the optional argument \textit{in_counter} is missing, the numbering is carried out sequentially throughout the entire document. However, if the name of an existing counter, such as \textit{chapter}, is given for \textit{in_counter}, the numbering is reset every time that counter is augmented, and both are printed together, as in \textbf{Axiom 4.1} above.

The predefined structures are called with the command

\begin{verbatim}
\begin{struct_type}[extra_title] text \end{struct_type}
\end{verbatim}
which also increments the necessary counter and generates the proper number. The above examples were produced with

\begin{theorem}[Balzano--Weierstrass] Every .... \end{theorem}
\begin{axiom} The natural numbers form ........... \end{axiom}

The optional \texttt{extra\_title} also appears in bold face within parentheses ( ) following the running number.

Occasionally a structure is not numbered on its own but together with another structure. This can be included in the definition with another optional argument

\begin{tabbing}
\begin{tabbing}
\end{tabbing}

\begin{tabbing}
where \texttt{num\_like} is the name of an existing theorem structure that shares the same counter. Thus by defining \texttt{\newtheorem{subthrm}{Sub-Theorem}}\newtheorem{subthrm}{Sub-Theorem}, the two structures \texttt{theorem} and \texttt{subthrm} will be numbered as a single series: \texttt{Theorem 1, Sub-Theorem 2, Sub-Theorem 3, Theorem 4}, and so on.

For more powerful theorem tools, see the AM\texttt{s amsthm} package (Section 12.3.1) and the \texttt{theorem} package in the tools collection (Section B.5.4).

4.6 \hspace{0.5em} \textbf{Tabulator stops}

4.6.1 \hspace{0.5em} Basics

On a typewriter it is possible to set \textit{tabulator stops} at various positions within a line; then by pressing the tab key the print head or carriage jumps to the next tab location.

A similar possibility exists in L\LaTeX\ with the \texttt{tabbing} environment:

\begin{tabbing}
\begin{tabbing}
\end{tabbing}

One can think of the set tab stops as being numbered from left to right. At the beginning of the tabbing environment, no tabs are set, except for the left border, which is called the \textit{zeroth} tab stop. The stops can be set at any spot within a line with the command \texttt{\=}, and a line is terminated by the \texttt{\\\ } command:

Here is the \texttt{\=first tab stop, followed by\=} the second\\

sets the first tab stop after the blank following the word \textit{the}, and the second immediately after the word \textit{by}.

After the tab stops have been set in this way, one can jump to each of the stops in the subsequent lines, starting from the left margin, with the command \texttt{\>}. A new line is started with the usual \texttt{\\} command.
Example:

\begin{tabbing}
\hspace{1cm} Type \quad Quality \quad Color \quad Price \hspace{1cm} \\
\hspace{1cm} Paper \> med. \> white \> low \hspace{1cm} \\
\hspace{1cm} Leather \> good \> brown \> high \hspace{1cm} \\
\hspace{1cm} Card \> bad \> gray \> med. \hspace{1cm} \\
\end{tabbing}

\begin{tabbing}
\hspace{1cm} Type \quad Quality \quad Color \quad Price \hspace{1cm} \\
\hspace{1cm} Paper \> med. \> white \> low \hspace{1cm} \\
\hspace{1cm} Leather \> good \> brown \> high \hspace{1cm} \\
\hspace{1cm} Card \> bad \> gray \> med. \hspace{1cm} \\
\end{tabbing}

4.6.2 Sample line

It is often advantageous or even necessary to set the tab stops in a sample line that is not actually printed. It could contain, for example, the widest entries in the various columns that appear later, or the smallest intercolumn spacing between stops. The sample line may also contain \hspace commands to force the distance between stops to be a predetermined amount.

To suppress the printing of the sample line, it is ended with the command \kill instead of the \\ terminator.

\hspace*{3cm}=sample column =\hspace{4cm} = \kill

In addition to the left border, the above statement sets three tab stops:

\begin{center}
\begin{tabular}{c|c|c}
Left border & 1st tab stop & 2nd tab stop & 3rd tab stop \\
3cm & sample column & 4cm
\end{tabular}
\end{center}

An \hspace command at the beginning of a sample line must be of the *-form, otherwise the inserted spacing will be deleted at the line margin.

4.6.3 Tab stops and the left margin

The left border of each line of the tabbing environment is at first identical with the left margin of the enclosing environment, and is designated the zeroth stop. By activating the ‘tab key’ \> at the start of a line, one sets the following text beginning at the first tab stop. However, the command \+ has the same effect, putting the left border permanently at the first stop, for all subsequent lines. With \+\+ at the beginning or end of a line, all the next lines will start two stops further along. There can be as many \+ commands in all as there are tab stops set on the line.

The command \- has the opposite effect: it shifts the left border for the following lines one stop to the left. It is not possible to set this border to be to the left of the zeroth stop.

The effect of the \+ commands may be overridden for a single line by putting \< at the start for each tab to be removed. This line then starts so many tabs to the left of the present border. With the next \\ command, the new line begins at the current left border determined by the total number of \+ and \- commands.
4.6.4 Further tabbing commands

Tab stops can be reset or added in every line. The command `\=` will add a stop if there have been sufficient `\>` commands to have jumped to the last stop, otherwise it will reset the next stop.

For example:

```
Old column 1 Old column 2
Left column  Middle col  Extra col
New col 1 New col 2  Old col 3
Column 1 Column 2  Column 3
```

```
Old column 1  \=  Old column 2
\>
Left column  \>  Middle col
\>
New col 1  \=  New col 2  \>
\>
Old col 3
\>
Column 1\>  Column 2 \>  Column 3
\end\{tabbing\}
```

Occasionally it is desirable to be able to reset the tab stops and then to reuse the original ones later. The command `\pushtabs` accomplishes this by storing the current tabs and removing them from the active line. All the tab stops can then be set once again. The stored stops can be reactivated with the command `\poptabs`. The `\pushtabs` command may be given as many times as needed, but there must be the same number of `\poptabs` commands within any one tabbing environment.

It is possible to position text on a tab stop with `left_text \`right_text`, where `left_text` goes just before the current tab (or left border) with a bit of spacing, while `right_text` starts exactly at the stop. The amount of spacing between the `left_text` and the tab stop is determined by the tabbing parameter `\tabbingsep`. This may be changed by the user with the `\setlength` command as usual.

Text may be right justified up against the right border of a line with the command `\`text`. There must not be any more `\>` or `\=` commands in the remainder of the line.

The commands `\=`, `\`, and `\` function as accent commands outside of the `tabbing` environment (Section 2.5.7). If these accents are actually needed within `tabbing`, they must be produced with `\a=`, `\a\acute`, and `\a\` instead. For example, to produce ó, ò, or ò inside a `tabbing` environment, one must give `\a\'o`, `\a\acute{o}`, or `\a=a\acute{o}`. The command `\-` also has another meaning outside of the `tabbing` environment (suggested word division) but since lines are not broken automatically within this environment, there is no need for an alternative form.

Here is an example illustrating all the tabbing commands:
Apples: consumed by: people, horses, and sheep.
Reasonably juicy:

Grapefruits: a delicacy (see also: melons, pumpkins)

Horses feed on apples

\begin{tabbing}
Apples: > consumed by: \= people\+\+\+\+\+\,
\= horses \\,
\= and \ ' sheep\--\--\--\--\--\,
\= reasonably juicy\--\--\--\--\--\,
Grapefruits: > a delicacy\\,
\pushtabs
(see also: \= melons\\,
\= > pumpkins)\\,
\poptabs
Horses \> feed on \> apples
\end{tabbing}

4.6.5 Remarks on tabbing

\LaTeX{} treats the tabbing environment like a normal paragraph, breaking a page if necessary between two lines within the environment. However, the commands \texttt{\newpage} and \texttt{\clearpage} are not allowed within it, and the command \texttt{\pagebreak} is simply ignored. If the user wishes to force a page break within the tabbing environment, there is a trick that he or she may employ: specify a very large interline spacing at the end of the line where the break should occur (for example, \texttt{\[10cm\]}). This forces the break and the spacing disappears at the start of the new page.

Each line of text is effectively within a \{ \} pair, so that any size or font declarations remain in force only for that one line. The text need not be put explicitly inside a pair of curly braces.

It is not possible to nest tabbing environments within one another.

Beware: the tab jump command \texttt{\>} always moves to the next logical tab stop. This could actually be a move backwards if the previous text is longer than the space available between the last two stops. This is in contrast to the way the tabulator works on a typewriter.

There is no automatic line breaking within the tabbing environment. Each line continues until terminated by a \texttt{\\}} command. The text could extend beyond the right margin of the page. The user must take care that this does not happen.

The commands \texttt{\hfill}, \texttt{\hrulefill}, and \texttt{\dotfill} have no effect inside a tabbing environment, since no \texttt{stretching} takes place here.

\textit{Exercise 4.9: Generate the following table with the tabbing environment.}

Project: Total Requirements = $900\,000.00
\begin{tabbing}
of which \hspace{1cm} 2003 = $450\,000.00
2004 = $350\,000.00
2005 = $100\,000.00
\end{tabbing}
2003 approved: $350\,000.00  Deficiency: $100\,000.00
2004 $300\,000.00  $150\,000.00
2005 $250\,000.00  Surplus: $150\,000.00
Boxes

A box is a piece of text that \TeX\ treats as a unit, like a single character. A box (along with the text within it) can be moved left, right, up, or down. Since the box is a unit, \TeX\ cannot break it up again, even if it was originally made up of smaller individual boxes. It is, however, possible to put those smaller boxes together as one pleases when constructing the overall box.

This is exactly what \TeX\ does internally when it carries out the formatting: the individual characters are packed in character boxes, which are put together into line boxes horizontally with rubber lengths inserted between the words. The line boxes are stacked vertically into paragraph boxes, again with rubber lengths separating them. These then go into the page body box, which with the head and foot boxes constitutes the page box.

\TeX\ offers the user a choice of three box types: LR boxes, paragraph boxes, and rule boxes. The LR (left–right) box contains material that is ordered horizontally from left to right in a single line. A paragraph box will have its contents made into vertically stacked lines. A rule box is a rectangle filled solidly with black, usually for drawing horizontal and vertical lines.
4.7.1 LR boxes

To create LR boxes containing single line text the commands

\mbox{text} and \makebox[width][pos]{text}
\fbox{text} and \framebox[width][pos]{text}

are available. The two commands at the left produce an LR box with a width exactly equal to that of the text given between the braces \{ \}. The \fbox command is the same as \mbox except that the \text is also framed.

With the two commands at the right, the width is predetermined by the optional length argument \textwidth. The other optional argument \textpos specifies how the text is positioned within the box. With no value given, the text is centered. Otherwise \textpos may be

- l to left justify the text,
- r to right justify it,
- s to stretch it to fill up the full width.

Thus \makebox[3.5cm]{centered text} creates a box of width 3.5 cm in which the text is centered, as centered text, filled with white space, while with \framebox[3.5cm][r]{right justified} the text is right justified inside a framed box of width 3.5 cm: right justified.

One may also give

\framebox[3.5cm][s]{stretched\dotfill text}

to fill up the box, as stretched \ldots \text, in which case some rubber length (Section 2.4.2) or other filler (page 30) must be added where the stretching is to occur.

If the text has a natural width that is larger than that specified in width, it will stick out of the box on the left, right, or both sides, depending on the choice of \textpos. For example,

\framebox[2mm]{centered} produces centered

The above application may appear rather silly for \framebox, but it can indeed be very useful for \makebox. A width specification of 0 pt for \makebox can generate a centered, left, or right justified positioning of text in diagrams made with the picture environment (see Chapter 13 for examples). It may also be used to cause two pieces of text to overlap, as \makebox[0pt][l]{/}S prints a slash through an S, as $\$\$.

Note: Length specifications must always contain a dimensional unit, even when they are zero. Thus 0pt must be given for the width, not 0.

It is also possible to specify the width of an LR box relative to its natural dimensions (those produced by the simple \mbox command):

- \textwidth is the natural width of the box,
- \textheight is the distance from baseline to top,
- \textdepth is the distance from baseline to bottom,
- \texttotalheight is \textheight plus \textdepth.
To make a framed box such that the width is six times the total height, containing centered text,

\framebox[6\totalheight]{Text}

Note: These special length parameters only have meaning within the width specification of an LR box, or within the height specification of a paragraph box, as shown below. In any other context, they will produce an error message.

If a set piece of text is to appear in several places within the document, it can be stored by first giving the command

\newsavebox{\boxname}

to create a box with the name \boxname. This name must conform to LATEX command name syntax (letters only) with an initial \. The name must not conflict with any existing LATEX command names. After such a box has been initiated, the commands

\sbox{\boxname}{text} or
\savebox{\boxname}[width][pos]{text}

will store the contents text for future use. The optional arguments width and pos have the same meanings as for \makebox and \framebox. Now with the command

\usebox{\boxname}

the stored contents are inserted into the document text wherever desired, as a single unit.

The contents of an LR box may also be stored with the environment

\begin{lrbox}{\boxname}
  text
\end{lrbox}

This is equivalent to \sbox{\boxname}{text}. Its advantage is that it allows text within a user-defined environment (Section 8.4) to be stored for future use with \usebox.

4.7.2 Vertical shifting of LR boxes

The command

\raisebox{lift}[height][depth]{text}

produces an \mbox with contents text, raised above the current baseline by an amount lift. The optional arguments tell LATEX to treat the box as though its extension above the baseline were height and that below were depth. Without these arguments, the box has its natural size determined by text and lift. Note that lift, height, and depth are lengths (Section 2.4.1). If lift is negative, the box is lowered below the baseline.

For example:
Baseline \raisebox{1ex}{high} and \raisebox{-1ex}{low} and back again

produces: Baseline \text{high} and \text{low} and back again.

The values for \textit{height} and \textit{depth} can be totally different from the actual ones of the \textit{text}. Their effect is to determine how far away the previous and next lines of text should be from the current line, based on the heights and depths of all the boxes (characters are also boxes) in the line. By raising a box but specifying \textit{height} to be the regular character size, the raised box will overprint the line above, and similarly for \textit{depth} when a box is lowered.

4.7.3 Parboxes and minipages

Whole paragraphs can be put into separate vertical boxes (or \textit{parboxes} in the \LaTeX{} jargon) with the command

\parbox\[\textit{pos}\]{\textit{width}}\{\textit{text}\}

or with the environment

\begin{minipage}\[\textit{pos}\]{\textit{width} text \end{minipage}

Both produce a vertical box of width \textit{width}, in which the lines of text are stacked on top of each other as in normal paragraph mode.

The optional positioning argument \textit{pos} can take on the values

- \texttt{b} to align the bottom edge of the box with the current baseline,
- \texttt{t} to align the top line of text with the current baseline.

Without any positioning argument, the parbox is centered vertically on the baseline of the external line of text.

The positioning argument is only meaningful when the \texttt{parbox} command or the \texttt{minipage} environment occurs within a line of text, for otherwise the current line and its baseline have no meaning. If the parbox is immediately preceded by a blank line, it begins a new paragraph. In this case, the vertical positioning of the parbox is made with reference to the following elements of the paragraph. These could be further parboxes. If the paragraph consists of only a single parbox or minipage, the positioning argument is meaningless and has no effect.

Examples:

\parbox{3.5cm}{\sloppy This is a 3.5 cm wide parbox. It is vertically centered on the} \hfill CURRENT LINE \hfill
\parbox{5.5cm}{Narrow pages are hard to format. They usually produce many warning messages on the monitor. The command \texttt{\symbol{92}sloppy} can stop this.}
This is a 3.5 cm wide parbox. It is vertically centered on the CURRENT LINE.

Narrow pages are hard to format. They usually produce many warning messages on the monitor. The command \texttt{sloppy} can stop this.

\begin{minipage}[b]{4.3cm}
The minipage environment creates a vertical box like the parbox command. The bottom line of this minipage is aligned with the\end{minipage}\hfill
\parbox{3.0cm}{middle of this narrow parbox, which in turn is aligned with} \hfill
\begin{minipage}[t]{3.8cm}
the top line of the right hand minipage. It is recommended that the user experiment with the positioning arguments to get used to their effects.\end{minipage}

In Section 4.7.7 we demonstrate how parboxes can be vertically stacked in any desired manner relative to one another.

The \texttt{parbox} command produces a vertical box containing the text just like the \texttt{minipage} environment. However, the latter is more general. The \texttt{text} in a \texttt{parbox} may not contain any of the centering, list, or other environments described in Sections 4.2–4.5. These may, on the other hand, appear within a \texttt{minipage} environment. That is, a \texttt{minipage} can include centered or indented text, as well as lists and tabbings.

\subsection{Problems with vertical placement}

Vertical positioning of minipages and parboxes can often lead to unexpected results, which can be explained by showing more graphically how a box is treated by \LaTeX. Suppose we want to place two parboxes of different heights side by side, aligned on their first lines, and the two together set on the current line of text at the bottom. The 'obvious' way of doing this is

\begin{verbatim}
\begin{minipage}[b]{..}
\parbox[t]{..}{..} \hfill \parbox[t]{..}{..}
\end{minipage}
\end{verbatim}
which does not work, for it produces instead the following results:

Current line

| The boxes are made visible here by framing them, and by marking the baselines (the vertical alignment points) with black dots. |

The reason for this is that each parbox or minipage is treated externally as a single character with its own height and depth above and below the baseline. As far as the outer minipage is concerned, it contains only two ‘characters’ on the same line, and that line is both the top and bottom one. Thus the bottom line of the outer minipage is indeed aligned with the line of text, but that bottom line is simultaneously the top line. The solution is to add a dummy second line to the outer box, as

\parbox[t]{..}{..} \hfill \parbox[t]{..}{..} \mbox{}

The dummy line may not be entirely empty, hence the \mbox.

A similar problem occurs if two boxes are to be aligned with their bottom lines, and the pair aligned at the top with the current line of text. Here there are two possibilities to add a dummy \textit{first} line.

\mbox{} \\hspace*{-\baselineskip} \mbox{} \hspace*{\baselineskip}

aligns with the very top, or

aligns with the first text line.

An example of the first case is shown on page 176. Dummy lines are also needed for the solution of Exercise 4.10 on page 93.

4.7.5 \textbf{Paragraph boxes of specific height}

The complete syntax of the \parbox command and minipage environment includes two more optional arguments:

\parbox[\textit{pos}][\textit{height}][\textit{inner.pos}]{\textit{width}}{\textit{text}}

\begin{minipage}[\textit{pos}][\textit{height}][\textit{inner.pos}]{\textit{width}}
\text{\textit{text}}
\end{minipage}

In both cases, \textit{height} is a length specifying the height of the box; the parameters \textit{\textendash height, width, depth, and \textendash totalheight} may be employed within the \textit{height} argument in the same way as in the \textit{width} argument of \texttt{\makebox} and \texttt{\framebox} (page 86).
The optional argument \texttt{inner_pos} states how the text is to be positioned \textit{internally}, something that is only meaningful if \texttt{height} has been given. Its possible values are:

- \texttt{t} to push the text to the top of the box,
- \texttt{b} to shove it to the bottom,
- \texttt{c} to center it vertically,
- \texttt{s} to stretch it to fill up the whole box.

In the last case, rubber lengths (Section 2.4.2) should be present where the vertical stretching is to take place.

Note the difference between the external positioning argument \texttt{pos} and the internal one \texttt{inner_pos}: the former states how the box is to be aligned with the surrounding text, while the latter determines how the contents are placed within the box itself.

Example:

\begin{minipage}[t][2cm][t]{3cm}
This is a minipage of height 2\textasciicircum{}cm with the text at the top.
\end{minipage}\hrulefill
\parbox[t][2cm][c]{3cm}{In this parbox, the text is centered on the same height.}\hrulefill
\begin{minipage}[t][2cm][b]{3cm}
In this third paragraph box, the text is at the bottom.
\end{minipage}

This is a minipage of height 2 cm with the text at the top. In this parbox, the text is centered on the same height. In this third paragraph box, the text is at the bottom.

The \texttt{\hrulefill} commands between the boxes show where the baselines are. All three boxes are the same size and differ only in their values of \texttt{inner_pos}.

### 4.7.6 Rule boxes

A rule box is a basically a filled-in black rectangle. The syntax for the general command is:

\begin{verbatim}
\rule[lift]{width}{height}
\end{verbatim}

which produces a solid rectangle of width \texttt{width} and height \texttt{height}, raised above the baseline by an amount \texttt{lift}. Thus \texttt{\rule{8mm}{3mm}} generates \rule{8mm}{3mm}. Without the optional argument \texttt{lift}, the rectangle is set on the baseline of the current line of text.
The parameters \textit{lift}, \textit{width}, and \textit{height} are all lengths (Section 2.4.1). If \textit{lift} has a negative value, the rectangle is set below the baseline.

It is also possible to have a rule box of zero width. This creates an invisible line with the given \textit{height}. Such a construction is called a \textit{strut} and is used to force a horizontal box to have a desired height or depth that is different from that of its contents. For this purpose, \texttt{\vspace} is inappropriate because it adds additional vertical space to that which is already there.

For example: \texttt{\fbox{Text}} produces Text. In order to print Text, one has to tell \TeX{} that the box contents extend above and below the baseline by the desired amounts. This was done with \texttt{\fbox{\rule[-2mm]{0cm}{6mm}Text}}. What this says is that the text to be framed consists of ‘an invisible bar beginning 2 mm below the baseline, 6 mm long, followed by the word \textit{Text’}. The vertical bar indeed remains unseen, but it determines the upper and lower edges of the frame.

4.7.7 Nested boxes

The box commands described above may be nested to any desired level. Including an LR box within a parbox or a minipage causes no obvious conceptual difficulties. The opposite, a parbox within an LR box, is also possible, and is easy to visualize if one keeps in mind that every box is a unit, treated by \LaTeX{} as a single character of the corresponding size.

A parbox inside an \texttt{\fbox} command has the effect that the entire parbox is framed. The present structure was made with
\begin{verbatim}
\fbox{\fbox{\parbox{10cm}{A parbox...}}}
\end{verbatim}
This is a parbox of width 10 cm inside a framebox inside a second framebox, which thus produces the double framing effect.

Enclosing a parbox inside a \texttt{\raisebox} allows vertical displacements of any desired amount. The two boxes here both have positioning \texttt{[b]}, but the one at the right has been produced with:
\begin{verbatim}
\raisebox{1cm}{\begin{minipage}[b]{2.5cm}
a b c d e ... x y z\\
\underline{baseline}
\end{minipage} }
\end{verbatim}
which displaces it upwards by 1 cm. \texttt{baseline}

A very useful structure is one in which \texttt{minipage} environments are positioned relative to one another inside an enclosing \texttt{minipage}. The positioning argument of the outside \texttt{minipage} can be used to align its
contents as a unit with the neighboring text or boxes. An example of this is given in Exercise 4.10.

Finally, vertical boxes such as \parbox commands and minipage environments may be saved as the text in an \sbox or \savebox command, to be recalled later with \usebox, as described in Section 4.7.1.

4.7.8 Box style parameters

There are two style parameters for the frame boxes \fbox and \framebox that may be reset by the user:

\fboxrule determines the thickness of the frame lines,
\fboxsep sets the amount of spacing between the frame and enclosed text.

New values are assigned to these length parameters in the usual \TeX manner with the command \setlength: the line thickness for all the following \framebox and \fbox commands is set to 0.5 mm with \setlength{\fboxrule}{0.5mm}.

The scope of these settings also obeys the usual rule: if they are found in the preamble then they apply to the entire document; if they are within an environment then they are valid only until the end of that environment.

These parameters do not influence the \framebox command that is employed within the picture environment (Section 13.1.4) and which has different syntax and functionality from those of the normal \framebox command.

**Exercise 4.10:** How can the following nested structure be generated? (Note: font size is \footnotesize)

The first line of this 3.5 cm wide minipage or parbox is aligned with the first line of the neighboring minipage or parbox.

This 4.5 cm wide minipage or parbox is positioned so that its top line is at the same level as that of the box on the left, while its bottom line is even with that of the box on the right. The naïve notion that this arrangement may be achieved with the positioning arguments set to t, t, and b is incorrect. Why? What would this selection really produce?

The true solution involves the nesting of two of the three structures in an enclosing minipage, which is then separately aligned with the third one.

Note: there are two variants for the solution, depending on whether the left and middle structures are first enclosed in a minipage, or the middle and right ones. Try to work out both solutions. Incidentally, the third minipage is 3 cm wide.

**Exercise 4.11:** Produce the framed structure shown below and store it with the command \sbox{\warning}{structure}. You will first have to create a box named \warning with the \newsavebox{\warning} command. Print this warning at various places in your exercise file by giving \usebox{\warning}.


Vertical placement of minipages and parboxes can lead to surprising results which may be corrected by the use of dummy lines.

Note: the parbox width is 10 cm. There should be no difficulty producing the framed structure if one follows the previous example for the double framed box. Watch out when writing \sbox{\warning}{\text{structure}} that you have the correct number of closing braces at the end.

Next, change the values for the line thickness (\fboxrule) and frame spacing (\fboxsep) and print your results once more.

### 4.7.9 Further framed boxes

<table>
<thead>
<tr>
<th>Package: fancybox</th>
<th>The fancybox package, by Timothy van Zandt, allows additional framed boxes of various styles. These all make use of the length \fboxsep to set the distance between frame and text, the same as for \fbox and \framebox. Depending on the box type, additional new lengths are also applicable. They may be changed with \setlength to modify the box appearance. These new framed boxes are:</th>
</tr>
</thead>
<tbody>
<tr>
<td>\shadowbox{\text}</td>
<td>The width of the shadow is given by the length \shadowsize, default 4 pt. Multiline text must be placed in a minipage environment, the same as for \fbox.</td>
</tr>
<tr>
<td>\doublebox{\text}</td>
<td>The width of the inner frame is 0.75\fboxrule, that of the outer frame 1.5\fboxrule, and the spacing between the frames is 1.5\fboxrule plus 0.5 pt.</td>
</tr>
<tr>
<td>\ovalbox{\text}</td>
<td>The thickness of the frame is that of \thinlines (Section 13.1.4), the diameter of the corners is set with the command \cornersize{frac}, to frac times the smaller of the box width or height, or with \cornersize*{size} to the length size. The default is frac=.5.</td>
</tr>
<tr>
<td>\Ovalbox{\text}</td>
<td>The frame thickness is set by \thicklines, but otherwise is the same as \ovalbox.</td>
</tr>
</tbody>
</table>
4.8. Tables

This package also allows all pages to be boxed as part of the page style. This is done by issuing \fancypage{cmds1}{cmds2}, where \textit{cmds1} and \textit{cmds2} are commands setting box parameters, terminated by one of the box commands \fbox, \shadowbox, and so on. The first set, \textit{cmds1}, form a box with the head and footlines outside, while the second set draw a box including them. Normally one would only specify one set, leaving the other blank, such as:

\fancypage{\setlength{\fboxsep}{5pt}
  \setlength{\shadowsize}{3pt}\shadowbox}{}

for a shadow box on each page excluding head and footlines. There is also the command \thisfancypage{cmds1}{cmds2} to box just the current page.

4.8 Tables

With the box elements and tabbing environment from the previous sections it would be possible to produce all sorts of framed and unframed tables. However, \LaTeX offers the user far more convenient ways to build such complicated structures.

4.8.1 Constructing tables

The environments \texttt{tabular}, \texttt{tabular*}, and \texttt{array} are the basic tools with which tables and matrices can be constructed. The syntax for these environments is

\begin{array}{pos}{cols}\end{array}
\begin{tabular}{pos}{cols}\end{tabular}
\begin{tabular*}{width}{pos}{cols}\end{tabular*}

The \texttt{array} environment can only be applied in mathematical mode (see Chapter 5). It is described here only because its syntax and the meaning of its arguments are exactly the same as those of the \texttt{tabular} environment. All three environments actually create a minipage. The meaning of the arguments is as follows:

\texttt{pos} Vertical positioning argument (see also the explanation of this argument for parboxes in Section 4.7.3). It can take on the values

\begin{itemize}
  \item[t] the top line of the table is aligned with the baseline of the current external line of text;
  \item[b] the bottom line of the table is aligned with the external baseline;
\end{itemize}

with no positioning argument given, the table is centered on the external baseline.
width This argument applies only to the tabular* environment and determines its overall width. In this case, the cols argument must contain the @-expression (see below) @\{\extracolsep\{\fill\}\} somewhere after the first entry. For the other two environments, the total width is fixed by the textual content.

cols The column formatting argument. There must be an entry for every column, as well as possible extra entries for the left and right borders of the table or for the intercolumn spacings.

The possible column formatting symbols are

- l the column contents are left justified;
- r the column contents are right justified;
- c the column contents are centered;
- p{wth} the text in this column is set into lines of width wth, and the top line is aligned with the other columns. In fact, the text is set in a parbox with the command \parbox[t]{wth}{column text};

-N{num}{cols} the column format contained in cols is reproduced num times, so that *{5}{|c|} is the same as |c|c|c|c|c|.

The available formatting symbols for the left and right borders and for the intercolumn spacing are

- | draws a vertical line;
- || draws two vertical lines next to each other;
- @{text} this entry is referred to as an @-expression, and inserts text in every line of the table between the two columns where it appears.

An @-expression removes the intercolumn spacing that is automatically put between each pair of columns. If white space is needed between the inserted text and the next column, this must be explicitly included with \hspace{ } within the text of the @-expression. If the intercolumn spacing between two particular columns is to be something other than the standard, this may be easily achieved by placing @{\hspace{wth}} between the appropriate columns in the formatting argument. This replaces the standard intercolumn spacing with the width wth.

An \extracolsep{wth} within an @-expression will put extra spacing of amount wth between all the following columns, until countermanded by another \extracolsep command. In contrast to the standard spacing, this additional spacing is not removed by later @-expressions. In the tabular* environment, there must be a command @\{\extracolsep\{\fill\}\} somewhere in the column format so that all the subsequent intercolumn spacings can stretch out to fill the predefined table width.
4.8. Tables

If the left or right borders of the table do not consist of a vertical line, spacing is added there of an amount equal to half the normal intercolumn spacing. If this spacing is not wanted, it may be suppressed by including an empty @-expression @{} at the beginning or end of the column format.

Rows contain the actual entries in the table, each horizontal row being terminated with a \ command. These rows consist of a sequence of column entries separated from each other by the & symbol. Thus each row in the table contains the same number of column entries as in the column definition cols. Some entries may be empty. The individual column entries are treated by \TeX as though they were enclosed in braces { }, so that any changes in type style or size are restricted to that one column.

\hline This command may only appear before the first row or immediately after a \ row termination. It draws a horizontal line the full width of the table below the row that was just ended, or at the top of the table if it comes at the beginning.

Two \hline commands together draw two horizontal lines with a little space between them.

\cline{m-n} This command draws a horizontal line from the left side of column m to the right side of column n. Like \hline, it may only be given just after a \ row termination and there may be more than one after another. The command \cline{1-3} \cline{5-7} draws two horizontal lines from column 1 to 3 and from column 5 to 7, below the row that was just ended. In each case, the full column widths are underlined.

\multicolumn{num}{col}{text} This command combines the following num columns into a single column with their total width including intercolumn spacings. The argument col contains exactly one of the positioning symbols l, r, or c, with possible @-expressions and vertical lines |. A value of 1 may be given for num when the positioning argument is to be changed for that column in one particular row.

In this context, a 'column' starts with a positioning symbol l, r, or c, and includes everything up to but excluding the next one. The first column also includes everything before the first positioning symbol. Thus |c@{}r| contains three columns: the first is |c@{}|, the second r, and the third |.

The \multicolumn command may only come at the start of a row or right after a column separation symbol &.

\vline This command draws a vertical line with the height of the row at the location where it appears. In this way, vertical lines that do not extend the whole height of the table may be inserted within a column.
4.8.2 Table style parameters

There are a number of style parameters used in generating tables which \LaTeX sets to standard values. These may be altered by the user, either globally within the preamble or locally inside an environment. They should not be changed within the tabular environment itself.

\texttt{\textbackslash tabcolsep} is half the width of the spacing that is inserted between columns in the \texttt{tabular} and \texttt{tabular*} environments;

\texttt{\textbackslash arraycolsep} is the corresponding half intercolumn spacing for the \texttt{array} environment;

\texttt{\textbackslash arrayrulewidth} is the thickness of the vertical and horizontal lines within a table;

\texttt{\textbackslash doublerulesep} is the separation between the lines of a double rule.

Changes in these parameters can be made with the \texttt{\setlength} command as usual. For example, to make the line thickness to be 0.5 mm, give \texttt{\setlength{\arrayrulewidth}{0.5\textbackslash mm}}. Furthermore, the parameter \texttt{\arraystretch} can be used to change the distance between the rows of a table.

This is a multiplying factor, with a standard value of 1. A value of 1.5 means that the inter-row spacing is increased by 50\%. A new value is set by redefining the parameter with the command

\texttt{\renewcommand{\arraystretch}{factor}}

4.8.3 Table examples

Creating tables is much easier in practice than it would seem from the above list of formatting possibilities. This is best illustrated with a few examples.

The simplest table consists of a row of columns in which the text entries are either centered or justified to one side. The column widths, the spacing between the columns, and thus the entire width of the table are automatically calculated.
Position | Club | Games | W | T | L | Goals | Points
--- | --- | --- | --- | --- | --- | --- | ---
1 | Amesville Rockets | 33 | 19 | 13 | 1 | 66:31 | 51:15
2 | Borden Comets | 33 | 18 | 9 | 6 | 65:37 | 45:21
3 | Clarkson Chargers | 33 | 17 | 7 | 9 | 70:44 | 41:25
4 | Daysdon Bombers | 33 | 14 | 10 | 9 | 66:50 | 38:28
5 | Edgartown Devils | 33 | 16 | 6 | 11 | 63:53 | 38:28
6 | Freeburg Fighters | 33 | 15 | 7 | 11 | 64:47 | 37:29
7 | Gadsby Tigers | 33 | 15 | 7 | 11 | 52:37 | 37:29
8 | Harrisville Hotshots | 33 | 12 | 11 | 10 | 62:58 | 35:31
9 | Idleton Shovers | 33 | 13 | 9 | 11 | 49:51 | 35:31
10 | Jamestown Hornets | 33 | 11 | 11 | 11 | 48:47 | 33:33
11 | Kingston Cowboys | 33 | 13 | 6 | 14 | 54:45 | 32:34
12 | Lonsdale Stompers | 33 | 12 | 8 | 13 | 50:57 | 32:34
13 | Marsdon Heroes | 33 | 9 | 13 | 11 | 50:42 | 31:35
14 | Norburg Flames | 33 | 10 | 8 | 15 | 50:68 | 28:38
15 | Ollison Champions | 33 | 8 | 9 | 16 | 42:49 | 25:41
16 | Petersville Lancers | 33 | 6 | 8 | 19 | 31:77 | 20:46
17 | Quincy Giants | 33 | 7 | 5 | 21 | 40:89 | 19:47
18 | Ralston Regulars | 33 | 3 | 11 | 19 | 37:74 | 17:49

The above table is made up of eight columns, the first of which is right justified, the second left justified, the third centered, the next three right justified again, and the last two centered. The column formatting argument in the `tabular` environment thus appears as

```
{rlcrrrcc}
```

The text to produce this table is

```
\begin{tabular}{rlcrrrcc}
Position & Club & Games & W & T & L & Goals & Points \\
1 & Amesville Rockets & 33 & 19 & 13 & 1 & 66:31 & 51:15 \\
2 & Borden Comets & 33 & 18 & 9 & 6 & 65:37 & 45:21 \\
... & ..... & .. & .. & .. & .. & ... & ... \\
17 & Quincy Giants & 33 & 7 & 5 & 21 & 40:89 & 19:47 \\
18 & Ralston Regulars & 33 & 3 & 11 & 19 & 37:74 & 17:49 \\
\end{tabular}
```

In each row, the individual columns are separated from one another by the symbol `&` and the row itself is terminated with the `\` command. The `[0.5ex]` at the end of the first row adds extra vertical spacing between the first two rows. The last row does not need the termination symbol since it is ended automatically by the `\end{tabular}` command.

The columns may be separated by vertical rules by including the symbol `|` in the column formatting argument. Changing the first line to

```
\begin{tabular}{r|ll|c|rrr|c|c}
... & ..... & .. & .. & .. & .. & ... & ... \\
```

results in

<table>
<thead>
<tr>
<th>Position</th>
<th>Club</th>
<th>Games</th>
<th>W</th>
<th>T</th>
<th>L</th>
<th>Goals</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Amesville Rockets</td>
<td>33</td>
<td>19</td>
<td>13</td>
<td>1</td>
<td>66:31</td>
<td>51:15</td>
</tr>
<tr>
<td>2</td>
<td>Borden Comets</td>
<td>33</td>
<td>18</td>
<td>9</td>
<td>6</td>
<td>65:37</td>
<td>45:21</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>17</td>
<td>Quincy Giants</td>
<td>33</td>
<td>7</td>
<td>5</td>
<td>21</td>
<td>40:89</td>
<td>19:47</td>
</tr>
<tr>
<td>18</td>
<td>Ralston Regulars</td>
<td>33</td>
<td>3</td>
<td>11</td>
<td>19</td>
<td>37:74</td>
<td>17:49</td>
</tr>
</tbody>
</table>

The same symbol | before the first or after the last column format generates a vertical line on the outside edge of the table. Two symbols || produce a double vertical line. Horizontal lines over the whole width of the table are created with the command \hline. They may only appear right after a row termination \ or at the very beginning of the table. Two such commands \hline\hline draw a double horizontal line.

\begin{tabular}{|r|l||c|rrr|c|c|} \hline
<table>
<thead>
<tr>
<th>Position</th>
<th>Club</th>
<th>Games</th>
<th>W</th>
<th>T</th>
<th>L</th>
<th>Goals</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Amesville Rockets</td>
<td>33</td>
<td>19</td>
<td>13</td>
<td>1</td>
<td>66:31</td>
<td>51:15</td>
</tr>
<tr>
<td>2</td>
<td>Borden Comets</td>
<td>33</td>
<td>18</td>
<td>9</td>
<td>6</td>
<td>65:37</td>
<td>45:21</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>17</td>
<td>Quincy Giants</td>
<td>33</td>
<td>7</td>
<td>5</td>
<td>21</td>
<td>40:89</td>
<td>19:47</td>
</tr>
<tr>
<td>18</td>
<td>Ralston Regulars</td>
<td>33</td>
<td>3</td>
<td>11</td>
<td>19</td>
<td>37:74</td>
<td>17:49</td>
</tr>
</tbody>
</table>
\end{tabular}

The table now appears as

<table>
<thead>
<tr>
<th>Position</th>
<th>Club</th>
<th>Games</th>
<th>W</th>
<th>T</th>
<th>L</th>
<th>Goals</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Amesville Rockets</td>
<td>33</td>
<td>19</td>
<td>13</td>
<td>1</td>
<td>66:31</td>
<td>51:15</td>
</tr>
<tr>
<td>2</td>
<td>Borden Comets</td>
<td>33</td>
<td>18</td>
<td>9</td>
<td>6</td>
<td>65:37</td>
<td>45:21</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>17</td>
<td>Quincy Giants</td>
<td>33</td>
<td>7</td>
<td>5</td>
<td>21</td>
<td>40:89</td>
<td>19:47</td>
</tr>
<tr>
<td>18</td>
<td>Ralston Regulars</td>
<td>33</td>
<td>3</td>
<td>11</td>
<td>19</td>
<td>37:74</td>
<td>17:49</td>
</tr>
</tbody>
</table>

In this case, the row termination \ must be given for the last row too because of the presence of \hline at the end of the table.

In this example, all rows contain the same entry in the third column, that is, 33. Such a common entry can be automatically inserted in the column format as an @-expression of the form @{text}, which places text between the neighboring columns. This could be accomplished for our example by changing the column format to

\{r1@{ 33 }rrrc\} or \{|r1||@{ 33 }|rrr|c|c|\}

so that the text ‘33’, blanks included, appears between the second and third columns in every row. This produces the same table with slightly different row entries: for example, the fourth row would now be given as
The column format now consists of only seven column definitions, rlrrrcc. The previous third column c has been removed, and so each row contains one less column separation symbol &. The new third column, the number of games won, begins with the second & and is separated from the club name by the contents of @{ 33 }, which is entered automatically without any additional & symbol.

The last two columns give the relations between goals and points won and lost as a centered entry of the form \( mn \). The colons ‘:’ are only coincidentally ordered exactly over one another since two-digit numbers appear in every case on both sides of the colon. If one entry had been 9:101, the colon would have been shifted slightly to the left as the entire entry was centered.

A vertical alignment of the ‘:’ independent of the number of digits can also be achieved using an @-expression of the form r@{:}l in the column format. This means that a colon is placed in every row between a right and a left justified column. The column formatting argument in the example now becomes

\[
{rl@{ 33 }rrrr@{:}lr@{:}l}
\]

and the row entry is

4 & Daysdon Bombers & 14 & 10 & 9 & 66:50 & 38:28 \ \]

Each of the former c columns has been replaced by the two columns in r@{:}l. An @-expression inserts its text between the neighboring columns, removing the intercolumn spacing that would normally be there. Thus the r column is justified flush right with the ‘:’ and the following l column flush left.

The same method can be employed when a column consists of numbers with decimal points and a varying number of digits. The entries for the goal and point relationships are now made up of two columns positioned about the ‘:’ symbol. This causes no problems for entering the number of goals won and lost or for the number of plus and minus points, since each entry has its own column. The column headings, however, are the words ‘Goals’ and ‘Points’, stretching over two columns each and without the colon. This is accomplished with the \multicolumn command, which merges selected columns in a particular row and redefines the column format. The first row of the unframed soccer table is then

\[
\text{Position & Club & W & T & L & \multicolumn{2}{c}{Goals} & \multicolumn{2}{c}{Points}}
\]

Here \multicolumn{2}{c}{Goals} means that the next two columns are to be combined into a centered column, containing the text ‘Goals’. For the framed table, the new formatting argument in the \multicolumn commands must be \{c\} since the vertical line symbol \| was also removed when the old columns were combined. In deciding what belongs to a given
The table of final results for our soccer league 2002/03 is to have the following title:

\begin{tabular}{|r|l||rrr|r@{:}l|r@{:}l||c|}
\hline
\multicolumn{10}{|c|}{\bfseries 1st Regional Soccer League --- Final Results 2002/03} \\\n\hline
\end{tabular}

\begin{tabular}{rrrrrrrr}
& & & & & & & & \end{tabular}

The horizontal lines for positions 3–5, 7–14, and 17 were made with the command \cline{1-9} while all the others used \hline:

\begin{verbatim}
11 & Kingston Cowboys & 13 & 6 & 14 & 54&45 & 32&34 & Medium Teams \cline{1-9}
18 & Ralston Regulars & 3 & 11 & 19 & 37&74 & 17&49 & Demoted
\end{verbatim}

The last two rows of the table deserve a comment. The remark ‘Demoted’ is vertically placed in the middle of the two rows. This is accomplished by typing

\begin{verbatim}
& \raisebox{1.5ex}[0pt]{Demoted}\\hline
\end{verbatim}
The \texttt{\raisebox} command lifts the text ‘Demoted’ by 1.5 ex. If the optional argument [0pt] had been left out, this lifting of the box would have increased the total height of the last row by 1.5 ex. This would have resulted in correspondingly more vertical spacing between the horizontal line of row 17 and the text of row 18. This additional spacing is suppressed by the optional argument \texttt{height = [0pt]}. (See Section 4.7.2 for a description of the \texttt{\raisebox} command.)

Occasionally one wants to increase the vertical spacing between horizontal lines and enclosed text. The soccer table would look better if the heading were thus:

\begin{verbatim}
\multicolumn{10}{|c|}{\rule[-3mm]{0mm}{8mm}\bfseries 1st Regional Soccer League --- Final Results 2002/03}\hline
\end{verbatim}

The included rule has a width of 0 mm, which makes it invisible, extends 3 mm below the baseline, and is 8 mm high. It thus stretches 5 mm (8 − 3) above the baseline. It effectively pushes the horizontal lines away from the baseline in both directions. If a row consists of more than one column, it is sufficient to include a \texttt{strut} in only one of them since the size of the whole row is determined by the largest column.

**Exercise 4.12:** Produce your own table for the final results of your favorite team sport in the same manner as for the soccer results above. Watch out that the colons ‘:’ are properly aligned for the goals and points relationships.

**Exercise 4.13:** Generate the following timetable.

<table>
<thead>
<tr>
<th>Day</th>
<th>6.15–7.15 pm</th>
<th>7.20–8.20 pm</th>
<th>8.30–9.30 pm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Subj.</td>
<td>Teacher Room</td>
<td>Subj.</td>
</tr>
<tr>
<td>Mon.</td>
<td>UNIX</td>
<td>Dr. Smith</td>
<td>Fortran</td>
</tr>
<tr>
<td></td>
<td>Comp. Ctr</td>
<td>Hall A</td>
<td>Conf. Room</td>
</tr>
<tr>
<td>Tues.</td>
<td>\LaTeX</td>
<td>Miss Baker</td>
<td>Fortran</td>
</tr>
<tr>
<td></td>
<td>Conf. Room</td>
<td>Conf. Room</td>
<td>Hall A</td>
</tr>
<tr>
<td>Wed.</td>
<td>UNIX</td>
<td>Dr. Smith</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>Comp. Ctr</td>
<td>Hall B</td>
<td>Hall B</td>
</tr>
<tr>
<td>Fri.</td>
<td>\LaTeX</td>
<td>Miss Baker</td>
<td>C++</td>
</tr>
<tr>
<td></td>
<td>Conf. Room</td>
<td>Conf. Room</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 4. Displayed Text

The entries ‘Day’ and ‘Subj.’ are raised in the same way as ‘Demoted’ was in the soccer table. To simplify its application, one can introduce a user-defined command with

\newcommand{\rb}[1]{\raisebox{1.5ex}[0pt]{$#1$}} (see Section 8.3.2)

so that \rb{entry} behaves the same as \raisebox{1.5ex}[0pt]{$entry$}. This can be used, for example, as \rb{ Mon. } or \rb{UNIX} to elevate the entries by the necessary amount.

In all the above examples, the entries in the individual columns are each a single line. Some tables contain certain columns with several lines of text that are somewhat separated from the rest of the row:

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>FBD 360</td>
<td>Desktop: XP3600+ Processor, 512 MB DDR-RAM, 80 GB hard disk, 16x DVD drive,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>32x CDRW drive, 64 MB TV output, Windows XP, 15” monitor</td>
<td>999.00</td>
</tr>
<tr>
<td>FBD 480</td>
<td>Desktop Deluxe: same as FBD 360 but with XP4800+ Processor, 48x CDRW drive,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>17” monitor</td>
<td>1399.00</td>
</tr>
<tr>
<td>FBT 240</td>
<td>Laptop: XP2400+, 512 MB RAM, 40 GB hard disk, 56 kb modem, 32x speed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DVD/CD-RW drive, 2x USB, 15” display, Windows XP, Infrared interface</td>
<td>1299.00</td>
</tr>
</tbody>
</table>

The above table is made up of three columns, the first left justified, the third right justified. The middle column contains several lines of text with a line width of 7.5 cm. This is generated with the column formatting symbol \{lp\{width\}. The whole column formatting argument in this example is \{lp\{7.5cm\}r\}.

\begin{tabular}{lp{7.5cm}r}
\bfseries Model & Description & \bfseries Price \\
FBD 360 & \small{\bfseries Desktop}: XP3600+ Processor, 512 MB DDR-RAM, 80 GB hard disk, 16x DVD drive, 32x CDRW drive, 64 MB TV output, Windows XP, 15” monitor & 999.00 \\\nFBD 480 & Desktop Deluxe: same as FBD 360 but with XP4800+ Processor, 48x CDRW drive, 17” monitor & 1399.00 \\
FBT 240 & Laptop: XP2400+, 512 MB RAM, 40 GB hard disk, 56 kb modem, 32x speed DVD/CD-RW drive, 2x USB, 15” display, Windows XP, Infrared interface & 1299.00 \\
\end{tabular}

The text for the middle column is simply typed in, being broken up into lines of width 8.0 cm automatically. The column is separated from the others with the & symbol in the usual way.

Warning: The line termination command \ \ is ambiguous within a p column, for it can either start a whole new row, or if \raggedright or \centering have been given, it ends a line of text within that column entry. In this case, if this is the last column in the row, the only way to
terminate the row is with `\tabularnewline` which always starts a new row.

**Exercise 4.14:** Produce the following table.

<table>
<thead>
<tr>
<th>Course and Date</th>
<th>Brief Description</th>
<th>Prerequisites</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction to LSEDIT March 14 - 16</td>
<td>Logging on — explanation of the VMS file system — explanation and intensive application of the VMS editor LSEDIT — user modifications</td>
<td>none</td>
</tr>
<tr>
<td>Introduction to \LaTeX \ March 21 - 25</td>
<td>Word processors and formatting programs — text and commands — environments — document and page styles — displayed text — math equations — simple user-defined structures</td>
<td>LSEDIT</td>
</tr>
</tbody>
</table>

The final example describes a blank form produced as a framed table. The difficulty here is to set the heights and widths of the empty boxes, since these are normally determined automatically by the text entries. The example shows how this may be accomplished with the help of `struts` and `\hspace` commands.

```
\begin{tabular} {|l|c|c|c|}
\hline
\multicolumn{4}{|c|}{\bfseries Budget Plan 2003--2004}\hline
\hline
Project & Nr. & Name & \hline
\hline
Year & 2003 & 2004 & 2005 & \hline
|€| US $ & |€| US $ & |€| US $ & \\
\hline
Investment Costs & & & & \\
Operating Costs & & & & \\
Industrial Contracts & & & & \\
Signature & & & Authorization & \\
\hline
\end{tabular}
```

\sbox{\k}{\framebox[4mm]{\rule{0mm}{3mm}}}
\sbox{\kkk}{\usebox{\k}\usebox{\k}\usebox{\k}}
\newsavebox{\k}
\newsavebox{\kkk}
\begin{tabular} {|l|c|c|c|}
\end{tabular}

\begin{verbatim}
\multicolumn{4}{|c|}{\bfseries Budget Plan 2003--2004}\\\
\hline
\hline
Project & Nr. & Name & \hline
\hline
Year & 2003 & 2004 & 2005 & \hline
|€| US $ & |€| US $ & |€| US $ & \\
\hline
Investment Costs & & & & \\
Operating Costs & & & & \\
Industrial Contracts & & & & \\
Signature & & & Authorization & \\
\hline
\end{verbatim}
The first three lines are only indirectly related to the table construction. They arrange for three empty boxes to be drawn when the command \usebox{\kkk} is given (see Section 4.7.1).

Except for the command \hspace{2.5cm} to set the column widths of the last three columns and the command \vline to draw a vertical line within a column, this example contains nothing new that was not in the previous examples. It is only necessary to give a brief explanation of the last row in the table:

The command \multicolumn{4}{|l|}{\rule[-1.0cm]{0mm}{1.3cm}Signature \hspace{5cm}\vline˜Authorization} \ \ \hline
\end{tabular}

The above examples clearly illustrate how the column widths and row heights are automatically determined for tables. These sizes, however, may be influenced by struts and \hspace commands. In addition, the commands described in Section 4.8.2 permit the intercolumn and interrow spacings as well as line thickness to be altered. For example,

\setlength{\tabcolsep}{5mm}

inserts 5 mm of spacing before and after every column; that is, it produces an intercolumn spacing of 10 mm. Section 4.8.2 gives more information about the use of these table style parameters.

4.8.4 Extension packages for tables

As powerful as the tabular environment is, it does have limitations. For this reason, there are a number of tools packages (Section B.5.4) that add additional features for constructing tables.
The array package extends the normal functionality of the \texttt{tabular} and \texttt{array} environments by adding several column formatting arguments, and by allowing the user to be able to define his or her own such arguments. \texttt{m\{wth\}} produces a column of width \texttt{wth} which is aligned vertically in the middle. (The standard \texttt{p\{wth\}} aligns the text with the top line.) \texttt{b\{wth\}} is like \texttt{p} and \texttt{m} but aligns the text on the bottom line. \texttt{\{decl\}} inserts \texttt{decl} before the next column; thus \texttt{\{\bfseries\}} sets the entire column in bold face without having to type \texttt{\bfseries} in each row. \texttt{\{decl\}} inserts \texttt{decl} after the last column; to have a centered column in math mode, give \texttt{\{\$\}c\{\$\}}. \texttt{\{decl\}} inserts \texttt{decl} between two columns without removing the normal intercolumn spacing, as for \texttt{\@\{decl\}}.

With \texttt{\newcolumntype\{type\}\{decl\}} one can define new column specifiers for multiple applications. For example, to have C defined as a centered math column, give

\begin{verbatim}
\newcolumntype{C}{>{$}c<{$}}
\end{verbatim}

The height of all rows can be increased by setting \texttt{\extrarowheight} to some value with \texttt{\setlength}; this is useful to prevent horizontal lines from being too close to the text below them.

With \texttt{\firsthline} and \texttt{\lasthline}, horizontal lines can be issued before the first and after the last rows respectively without interfering with the vertical alignment of the table.

The \texttt{dcolumn} package loads the \texttt{array} package and defines a column specifier \texttt{D} to align a column of numbers on a decimal point. Its syntax is

\texttt{D\{in\_char\}\{out\_char\}\{number\}}

where \texttt{in\_char} is the input character for the decimal point (say \texttt{.}), \texttt{out\_char} is the character that is output (say \texttt{\cdotp}), and \texttt{number} is the maximum number of decimal places. If \texttt{number} is negative, the column is centered on the decimal point, otherwise it is right justified. Later versions allow \texttt{number} to specify the number of digits on both sides of the decimal point, for example as 3.2.

The \texttt{tabularx} package loads the \texttt{array} package and defines an environment \texttt{tabularx} which makes a table of a desired total width, like \texttt{tabular*}, but in which the columns expand, not the intercolumn spacings. The column specifier \texttt{X} is used to indicate the expandable columns, and is equivalent to \texttt{p\{wth\}} where \texttt{wth} is adjusted to the necessary size. For example,

\begin{verbatim}
\begin{tabularx}{10cm}{c X l X} ... \end{tabularx}
\end{verbatim}

produces a table of width 10 cm with columns 2 and 4 expandable.
The delarray package loads the array package and redefines the\ array environment so that it may be enclosed in braces that are automatically adjusted in size, as with the\ \left and\ \right commands of Section 5.4.1. The braces surround the column specifier. For example,

\begin{array}\{cc\} a & b \\ c & d \end{array} \Rightarrow \begin{bmatrix} a & b \\ c & d \end{bmatrix}

The longtable package produces tables extending over several pages. It does not require the array package, but does recognize its extra features if loaded. The longtable environment takes the same column formatting argument as tabular and array, but has additional row entries at the start to determine:

- those rows that appear at the start of the table, terminated by\ \endfirsthead; this often includes the main\ \caption;
- those at the top of every continuation page, terminated by\ \endhead; these normally include an additional\ \caption and the column headers;
- those at the bottom of each page, terminated by\ \endfoot;
- and those rows at the end of the table, terminated by\ \endlastfoot.

An example of a long table is:

\begin{longtable}\{|l|c|r|\}
\caption[Short title]{Demonstration of a long table}\\
\hline
Left & Center & Right \\
\hline \endfirsthead
\caption[]{\emph{continued}}\\
\hline Left & Center & Right \\
\hline \endhead
\hline \multicolumn{3}{r}{\emph{continued on next page}} \\
\endfoot
\hline \endlastfoot
Twenty-two & fifty & A hundred and eighty \\
22 & 50 & 180 \\
\ldots \ldots \ldots \\
\end{longtable}

The\ \caption command normally may only appear within\ table and\ figure environments (Section 7.4) but may also be used within\ longtable which never goes into a\ table environment; in a continued row,\ \caption must have an empty optional argument\ [ ] to prevent multiple entries in the list of tables.

A\ \newpage command forces a new page within the long table.

Up to four\ \LaTeX runs may be needed to get the column widths right.
### Primary Energy Consumption

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Consumption (in million tons of BCU(^a))</td>
<td>347.7</td>
<td>390.2</td>
<td>385.0</td>
</tr>
<tr>
<td>of which (percentages)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>petroleum</td>
<td>52.1</td>
<td>47.6</td>
<td>43.2</td>
</tr>
<tr>
<td>bituminous coal</td>
<td>19.1</td>
<td>19.8</td>
<td>20.0</td>
</tr>
<tr>
<td>brown coal</td>
<td>9.9</td>
<td>10.0</td>
<td>8.6</td>
</tr>
<tr>
<td>natural gas</td>
<td>14.2</td>
<td>16.5</td>
<td>15.1</td>
</tr>
<tr>
<td>nuclear energy</td>
<td>2.0</td>
<td>3.7</td>
<td>10.1</td>
</tr>
<tr>
<td>other(^b)</td>
<td>2.7</td>
<td>2.3</td>
<td>3.0</td>
</tr>
</tbody>
</table>

\(^a\)BCU = Bituminous Coal Unit (1 ton BCU corresponds to the heating equivalent of 1 ton of bituminous coal = 8140 kwh)

\(^b\)Wind, water, solar energy, etc.


### 4.8.5 Floating tables

In Chapter 7 we explain how figures and tables can be made to float, that is, to be moved to the top or bottom of a page, or to a separate page altogether. The reason for doing this is that the author does not know in advance where the page breaks will occur, and so does not know if there is enough room on the current page for such a large object. The float mechanism allows the material to be saved and then placed later at an appropriate location. It also allows for table and figure captions and automatic numbering.

Table material is made to float with the environment:

\begin{table}
\textbf{head text} \\
\textit{table} \\
\textbf{foot text} \\
\end{table}

where \textit{table} stands for the entire table as defined in a \texttt{tabular} environment, \textbf{head text} for whatever text appears above the table, and \textbf{foot text} for that below. Widths, spacing, and positioning of the texts relative to the table are all matters for the user to arrange.

Independently of the enclosed text, everything that appears between \begin{table} and \end{table} is normally placed at the start of the current page. If a table already occupies the top of the page, an attempt is made to place it at the page bottom, if there is enough space for it. Otherwise, it will be placed on the next page, where further tables may be accumulated. The surrounding text is printed as though the table were not there. For further details about floats in general, including automatic sequential numbering, see Chapter 7.

The table at the top of this page was generated within the text at this location with the following (excluding the footnotes, which are described later in Section 4.10.4):
Chapter 4. Displayed Text

\begin{table}
\textbf{Primary Energy Consumption}\\[1ex]
\begin{tabular*}{118mm}{@{}ll...rr@{}}
\end{tabular*}\\[0.5ex]
\textit{Source:} Energy Balance Study Group, ...
\end{table}

There are a number of formatting parameters that may be used in connection with the \texttt{table} environment, which are described together with those for figures in Section 7.3.

Exercise 4.15: Complete the above text for the table on the previous page (without the footnotes). Pay attention to the following questions (check the explanations for the @-expressions in Section 4.8.1):

1. What is the effect of the @{} entries at the beginning and end of the formatting definition?

2. The \texttt{tabular*} environment generates a table with a given width, here 118 mm. What would be the effect of @{extracolsep{\fill}} at the beginning of the formatting definition?

3. Where in the formatting definition should @{extracolsep{\fill}} and the countermanding @{\hspace{1em}@{extracolsep{1em}}} appear in order to format the table as it is printed here? How would the table appear if only @{\extracolsep{1em}} were given as countermand?

4.9 Printing literal text

Occasionally it is necessary to print text exactly as it is typed, with all special characters, blanks, and line breaks appearing literally, unformatted, and in a typewriter font. Lines of computer code or samples of \LaTeX\ input text are examples of such literal text. This is accomplished with the environments

\begin{verbatim}
\begin{verbatim*}
\end{verbatim*}
\end{verbatim}

A new line is inserted before and after these environments.

With the *-form, blanks are printed with the symbol \_ to make them visible.

As an example, on page 115 some input text is printed to demonstrate the use of footnotes in forbidden modes. This was done with

\begin{verbatim}
\addtocounter{footnote}{-1}\footnotetext{Small insects}
\stepcounter{footnote}\footnotetext{Large mammals}
\end{verbatim}

Literal text may also be printed within a line using the commands \texttt{\verb} and \texttt{\verb*}, as for example
where the first character after \verb or \verb* (here =) is the delimiter, such that all text up to the next occurrence of that character is printed literally. This character may not appear in the literal text, obviously.

In contrast to the behavior in the \verbtext environment, the literal text must be all on one line in the input text, otherwise an error message is printed. This is to indicate that you just might have forgotten to repeat the delimiting character.

**Important:** neither the \verbtext environment nor the \verb command may be used in an argument of any other command!

**Exercise 4.16:** Reproduce some input lines from this book as literal text.

### 4.9.1 Extension packages for literal text

**Package:** alltt
The standard package alltt (Section B.5.3, page 392) provides an alltt environment that also prints its contents literally in a typewriter font, except that the characters \{ \} retain their normal meaning. Thus \LaTeX commands can be included within the literal text. For example,

\begin{alltt}
Underlining \underline{typewriter} text is also possible.
Note that dollar ($) and percent (%) signs are treated \textit{literally}.
\end{alltt}

**Package:** shortvrb
The standard package shortvrb (Section B.5.3, page 393) offers a shorthand for the \verb command. After issuing \MakeShortVerb{\textbackslash |}, one can print short literal text with |text|. The counter command \DeleteShortVerb{\textbackslash |} then restores the original meaning to |. Any character may be temporarily turned into a literal switch this way.

One problem with the \verbtext environment is that the entire literal text is input and stored before processing, something that can lead to memory overflows. The \verbtext package in the tools collection (Section B.5.4, page 396) re-implements the environment to avoid this problem. A minor drawback is that there must not be any other text on the same line as the \verbtext.

The \verbtext package offers two other extra features. It provides a comment environment that simply ignores its contents, as though each line started with a % sign (Section 4.11). And it adds a command \verbtextinput{filename} to input the specified file as literal text. This is useful for listing actual computer programs rather than copying them into the \LaTeX file.
4.9.2 Email and Internet addresses

Email and Internet addresses present some special problems that are solved with the `url` package by Donald Arseneau. These addresses are best listed in a typewriter font, often contain special symbols, and should never be hyphenated, since the hyphen could be interpreted as part of the address. The obvious solution would be to use the `\verb` command, which fulfills all these conditions. However, if the address does not fit on the current line, it will stick out into the right margin. Using `\texttt` instead will allow the words in the address to be broken over two lines, but at the price of inserting an ambiguous hyphen.

Inserting the address with the `url` command allows it to be broken at non-letters between words, without a hyphen. Moreover, its argument is treated literally, so all special characters are printed as given, just as with `\verb`. In fact, the argument of the `url` command may either be enclosed in curly braces as usual, or delimited by some arbitrary character, just as with `\verb`.

An Internet address may be given as `\url{http://address.edu/home/page/}`, or an email address might be given as `\url=fred.smith@general.services.gov=`.

The `url` command is to be preferred for another reason: it conforms to logical markup indicating the purpose of its argument. In fact, the `hyperref` package of Section 10.2.4 will even turn the argument into an active link, something it does not do for `\verb` arguments.

The printed appearance of the address can be controlled by specifying `\urlstyle{style}`, where `style` is one of `tt` (default), `rm`, `sf`, or `same`, for typewriter, Roman, sans serif, or unchanged font, respectively.

A fixed address can be predefined with, e.g.,

```latex
\urldef{\myurl}\url{myname@mydomain.uk}
```

Now `\myurl` is used to print `myname@mydomain.uk`.

There is also a `\path` command for giving directory/folder names, with the same syntax as `\url`. As logical markup, it has a different significance; for example, the `hyperref` package does not turn it into a link.

For further information on advanced uses of this package, see the comments at the end of the file `url.sty`.

4.10 Footnotes and marginal notes

4.10.1 Standard footnotes

Footnotes are generated with the command

```latex
\footnote{footnote_text}
```
which comes immediately after the word requiring an explanation in a footnote. The text `footnote.text` appears as a footnote in a smaller typeface at the bottom of the page. The first line of the footnote is indented and is given the same footnote marker as that inserted in the main text. The first footnote on a page is separated from the rest of the page text by means of a short horizontal line.

The standard footnote marker is a small, raised number\textsuperscript{1}, which is sequentially numbered. This footnote is produced with:

\[ ... \text{raised number} \footnote{The usual method of marking footnotes in a typewritten ... same page.}, \text{which is} ... \]

The footnote numbering is incremented throughout the whole document for the `article` class, whereas it is reset to 1 for each new chapter in the `report` and `book` classes.

The \footnote command may only be given within the normal paragraph mode, and not within math or LR modes. In practice, this means it may not appear within an LR box (Section 4.7.1) or a parbox (Section 4.7.3). However, it may be used within a \minipage environment, in which case the footnote text is printed beneath the minipage and not at the bottom of the actual page.\textsuperscript{2}

The \footnote command must immediately follow the word that is to receive the note, without any intervening blanks or spacing. A footnote at the end of a sentence can be given after the period, as in the last example above:

\[ ... \text{of the actual page.} \footnote{With nested ... wrong place.} \]

### 4.10.2 Non-standard footnotes

If the user wishes the footnote numbering to be reset to 1 for each \section command with the `article` class, this may be achieved with

\[ \setcounter{footnote}{0} \]

just before or after a \section command.

The internal footnote counter has the name `footnote`. Each call to \footnote increments this counter by one and prints the new value in Arabic numbering as the footnote marker. A different style of marker can be implemented with the command

\[ \renewcommand{\thefootnote}{\text{number style}{footnote}} \]

\textsuperscript{1}The usual method of marking footnotes in a typewritten manuscript with *, **, etc., could also be done; however, since the page breaks are not known at the time of typing the text, there would be a problem of avoiding duplication of the symbols on the same page.

\textsuperscript{2}With nested minipages, the footnote comes after the next \texttt{end\{minipage\}} command, which could be at the wrong place.
where \textit{number\_style} is one of the counter print commands described in Section 4.3.5: \texttt{\arabic}, \texttt{\roman}, \texttt{\Roman}, \texttt{\alph}, or \texttt{\Alph}. However, for the counter \texttt{footnote}, there is an additional counter print command available, \texttt{\fnsymbol}, which prints the counter values 1–9 as one of nine symbols:

\begin{verbatim}
* † ‡ § ¶ ∥ ** †† ‡‡
\end{verbatim}

It is up to the user to see that the footnote counter is reset to zero sometime before the tenth \texttt{\footnote} call.

An optional argument may be added to the \texttt{\footnote} command

\begin{verbatim}
\footnote[\texttt{num}]{\texttt{footnote\_text}}
\end{verbatim}

where \texttt{num} is a positive integer that is used instead of the value of the footnote counter for the marker. In this case, the footnote counter is not incremented. For example**,

\begin{verbatim}
\renewcommand{\thefootnote}{\fnsymbol{footnote}}
\end{verbatim}

For example\footnote[7]{The 7th symbol ... marker.},
\begin{verbatim}
\renewcommand{\thefootnote}{\arabic{footnote}}
\end{verbatim}

where the last line is necessary to restore the footnote marker style to its standard form. Otherwise, all future footnotes would be marked with symbols and not with numbers.

### 4.10.3 Footnotes in forbidden modes

A footnote marker can be inserted in the text with the command

\begin{verbatim}
\footnotemark[\texttt{num}]
\end{verbatim}

even where the \texttt{\footnote} command is normally not allowed, that is, in LR boxes, tables, and math mode. The marker is either the optional argument \texttt{num} or, if it is omitted, the incremented value of the footnote counter. The footnote itself is not generated. This must be done external to the forbidden mode with the command

\begin{verbatim}
\footnotetext[\texttt{num}]{\texttt{footnote\_text}}
\end{verbatim}

If the optional argument has been used for the footnote marker, the same \texttt{num} must be given as the option for the text command. Similarly, if no option was used for the marker, none may appear with the text. The footnote will be generated with the value of \texttt{num} or with that of the footnote counter.

This counter is incremented by a call to \texttt{\footnotemark} without an optional argument. The corresponding \texttt{\footnotetext} command, on the other hand, does not alter the counter.

**The 7th symbol appears as the footnote marker.**
If there are a number of \footnotemark commands without optional arguments appearing before the next \footnotetext command, it is necessary to adjust the counter with the command

\addtocounter{footnote}{dif}

where dif is a negative number saying how many times the counter must be set back. Then before every \footnotetext command, the counter must be incremented by one. This can be done either with the command \addtocounter, with dif=1, or with the command

\stepcounter{footnote}

which adds 1 to the given counter.

For example: \fbox{mosquitoes\footnotemark and elephants\footnotemark}

For example: \fbox{mosquitoes\footnotemark\ and \ elephants\footnotemark}

generates the footnote markers 3 and 4. Now the counter has the value 4.

In order for the first \footnotetext outside the framed box to operate with the correct counter value, it must first be decremented by one. The two footnote texts are made with

\addtocounter{footnote}{-1}\footnotetext{Small insects}
\stepcounter{footnote}\footnotetext{Large mammals}

immediately following the \fbox{} command. The footnote counter now has the same value as it did on leaving the \fbox.

4.10.4 Footnotes in minipages

As mentioned in Section 4.10.1, footnote commands are allowed inside the minipage environment. However, the footnote appears underneath the minipage, not below the main page.

Footnote commands within a minipage\footnote{The marker is a raised lower-case letter.} have a different marker style. The footnote comes after the next \end{minipage} command. Minipage footnotes have a counter separate from that of the main page, called mpfootnote, counting independently of footnote.

\begin{minipage}{6cm}
Footnote commands within a minipage\footnote{The marker is a raised lower-case letter.} have a different...
\end{minipage}

Footnotes within a tabular environment can normally only be generated with the commands described above: \footnotemark within the table and

\footnotemark[3]{Small insects}
\footnotemark[4]{Large mammals}
Chapter 4. Displayed Text

\footnotetext{outside the environment. However, if the \texttt{tabular} environment is inside a minipage, normal \texttt{footnote} commands may also be used inside the table. The footnote appears below the table where the minipage comes to an end.

\textbf{Exercise 4.17}: Produce a number of footnotes in your standard exercise file by inserting them where you think fit and by selecting some appropriate footnote text.

\textbf{Exercise 4.18}: Redefine the command \texttt{\thefootnote} so that the footnote markers become the symbols illustrated in Section 4.10.2. Add the redefinition to the preamble of your standard exercise file.

\textbf{Exercise 4.19}: Complete Exercise 4.15 so that the footnotes $^a$ and $^b$ appear as in the table on page 109.

\subsection*{4.10.5 Marginal notes}

Notes in the page margin are produced with the command

\begin{verbatim}
\marginar\texttt{\{note_text\}}
\end{verbatim}

which puts the text \texttt{note_text} into the margin beginning at the level of the line where the command is given. The marginal note appearing here was

\begin{verbatim}
... The marginal note \marginar\texttt{This\} is a\} margin-\}al note\}
\end{verbatim}

appearing here ...

The text is normally enclosed in a parbox of width 1.9 cm (0.75 in). Such a narrow box causes great difficulties with line breaking, which is why the lines are manually broken with the $\textbackslash$\ command in the above example. Such a box is far more appropriate for marginal notes in the form of a single symbol, such as the arrow shown here.

Another common use for the marginal note is to draw attention to certain text passages by marking them with a vertical bar in the margin. This is often done to indicate changes in a text, for comparison with earlier versions after updated single sheets have been redistributed. The example marking this paragraph was made by including

\begin{verbatim}
\marginar\texttt{\rule[-17.5mm]{1mm}{20mm}}
\end{verbatim}

in the first line.

The width of the marginal note can be changed with a style parameter described in the next section. The user must watch out that the total page width does not become too big for the printer.

By default, marginal notes appear in the right-hand margin of the page, or in the outer margin when the \texttt{twoside} option has been selected. ‘Outer’ means the right margin for odd pages, and left for even ones. With the \texttt{twocolumn} option, they are placed in the outside margins: left for the left column and right for the right one.
This leads to a problem for marginal markings such as the arrow illustrated on the previous page. On this page, it must point in the opposite direction. In fact, its direction depends on which side of the page it is to appear, and that in turns depends on the page number or column. Since these are not known at the time of writing (and may even change with later revisions) it is necessary to have another solution. This is provided by the extended syntax of the \marginpar command

\marginpar[left_text]{right_text}

This form of the command contains two versions of the marginal text, left_text to go into the left margin, and right_text for the right margin, depending on which one is selected. Both the arrows on this and the previous page were generated with the same command

\marginpar[\hfill$\Longrightarrow$]{$\Longleftarrow$}

(The mathematical arrow commands are explained in Section 5.3.5.)

Without the \hfill command in the above \marginpar example, the arrow in the left margin appears as it does at the side of this paragraph, too far over to the left. The reason for this is that the \marginpar command sets its contents flush with the left edge of the narrow margin box, made visible here with a frame. This left edge is aligned with the main text only when the note is put on the right side; however, in the left margin, it is displaced from the main text. The \hfill command has the effect of setting the contents flush with the right edge of the margin box, which is then properly aligned with the main text.

A similar device was used to make the first marginal note in this section. The actual command given was

\marginpar[\flushright This\ is a\ margin-al note] 
{This\ is a\ margin-al note}

Here \flushright (Section 4.2.2) is equivalent to putting an \hfill on each line.

The standard positioning of the marginal notes can be switched with \reversemarginpar. Once this command has been given, marginal notes will appear in the left margin, or in the 'inner' margin for the twoside option. The command \normalmarginpar restores normal behavior. These commands have no effect with the twocolumn option.

If the note appears at the bottom of a page, it will extend downwards below the last line of regular text. For this reason, and because of the difficulties with line breaking for narrow columns, marginal notes should be kept short, limited to a few words or a symbol.

4.10.6 Style parameters for footnotes and marginal notes

There are two footnote style parameters that may be changed as needed, either in the preamble or locally within an environment.

\footnotesep
The vertical spacing between two footnotes. This is a length that can be changed with the \setlength command.
The command that draws a horizontal line between the page text and the footnotes. It should not add any net vertical spacing. It may be changed, for example, by

\renewcommand{\footnoterule}
{\rule{wth}{hght}\vspace{-hght}}

A value of 0 cm for the \textit{hght} produces an invisible line of zero thickness.

The following style parameters may be changed to redefine how marginal notes appear:
- \marginparwidth determines the width of the margin box;
- \marginparsep sets the separation between the margin box and the main text;
- \marginparpush is the smallest vertical distance between two marginal notes.

These parameters are all lengths and are assigned new values as usual with the \texttt{\setlength} command.

## 4.11 Comments within text

All computer languages provide a means of inserting comments into the code. These are explanatory notes, documentation, history of development, or alternative text or code that has been temporarily deactivated. Comment lines are completely ignored during processing. They are only intended for human readers inspecting the source text.

In \LaTeX{}, the comment character is the percent sign \texttt{%}. When this character appears in the text, it and the rest of the line are ignored. If a comment is several lines long, each line must be prefixed with \texttt{%}.

As for other single character commands, the percent sign itself is printed with the command \texttt{\%}, as explained in Section 2.5.4.

The comment character \texttt{%} is also useful for experimenting with text or definitions of user commands or formatting parameters, to try alternatives without deleting the old versions. By ‘commenting out’ selective lines, one can play around with variations without losing them.

Large sections of text may be more effectively commented out with the \texttt{comment} environment from the \texttt{verbatim} package (page 111).

Finally, the \texttt{%} character has an important role to play in suppressing implied blanks at the end of a line. This is especially desirable in user definitions where unexpected blanks can creep in between otherwise invisible declarations with arguments. See Section 8.5.2.

**Exercise 4.20:** Comment out the changes from Exercise 4.18 in your preamble. These commands may be reactivated later by removing the \texttt{%} character.
Mathematical Formulas

Mathematics is the soul of \TeX. It was because the setting of mathematical formulas is so complicated in normal printing, not to mention on a typewriter, that Donald Knuth invented his text formatting system. On the other hand, the soul of \LaTeX is logical markup. Nevertheless, all the power of \TeX’s math setting is also available in \LaTeX, offering an unbeatable combination.

In this chapter, we confine ourselves to the elements of mathematical typesetting available in standard \LaTeX. The simplifications and additional elements provided by \AMS-\LaTeX are reserved for Chapter 12.

Mathematical formulas are produced by typing special descriptive text. This means that \LaTeX must be informed that the following text is to be interpreted as a mathematical formula, and it must also be told when the math text has come to an end and normal text recommences. The processing of math text is carried out by switching to math mode. Mathematical environments serve this purpose.

5.1 Mathematical environments

Mathematical formulas may occur within a line of text, as \((a + b)^2 = a^2 + 2ab + b^2\), or separated from the main text as

\[
\int_0^\infty f(x) \, dx \approx \sum_{i=1}^n w_i e^{x_i} f(x_i)
\]

These two types are distinguished by referring to them as text and displayed formulas respectively.

Text formulas, or equations, are generated with the environment

\begin{verbatim}
\begin{math} \text{formula_text} \end{math}
\end{verbatim}

Since text formulas are often very short, sometimes consisting of only a single character, a shorthand version is available as \( \text{formula_text} \).
However, most authors prefer the very short form $formula_text$ which is actually the \TeX method. All three are essentially the same, and there is no reason not to use the $ sign.

The contents of the formula, formula_text, consist of math constructs, which are described in the following sections.

Displayed formulas, or equations, are produced in the environments

\begin{displaymath} formula_text \end{displaymath}
\begin{equation} formula_text \end{equation}

The difference between these two is that the equation environment automatically adds a sequential equation number. The displaymath environment may be given with the shorthand forms \[
\] or $$ . . . $$.

By default displayed formulas are centered horizontally with the equation number, if it is present, set flush with the right margin. By selecting the document class option fleqn (Section 3.1.1), the formulas are set left justified with an adjustable indentation. This option remains valid for the entire document whereas the amount of indentation may be changed at will with \setlength{\mathindent}{indent}, where indent is a length specification. Moreover, the document class option leqno sets the equation numbers flush with the left margin throughout the whole document.

Finally, multiline formulas can be created with the environments

\begin{eqnarray} formula_text \end{eqnarray}
\begin{eqnarray*} formula_text \end{eqnarray*}

where the standard form adds a sequential equation number for each line and the *-form is without equation numbers.

## 5.2 Main elements of math mode

### 5.2.1 Constants and variables

Numbers that appear within formulas are called constants, whereas simple variables are represented by single letters. The universal practice in mathematical typesetting is to put constants in Roman typeface and variables in italics. \LaTeX adheres to this rule automatically in math mode. Blanks are totally ignored and are included in the input text simply to improve the appearance for the writer. Spacing between constants, variables, and operators like +, −, = are set automatically by \LaTeX. For example $z=2a+3y$, $ z = 2 a + 3 y $ both produce $ z = 2a + 3y $.

Mathematical symbols that are available on the keyboard are

+ − = < > / : ! ’ | [ ] ( )

all of which may be used directly in formulas. The curly braces { } serve the purpose of logically combining parts of the formula and therefore
cannot act as printable characters. To include braces in a formula, the
same commands \{ and \} are used as in normal text.

$$M(s) < M(t) < |M| = m$$

$${y''} = c \{ f[y',y(x)] + g(x) \}$$

**Preparation:** Create a new \LaTeX\ file with the name `math.tex` containing at first
only the commands `\documentclass{article}, `\begin{document}, and `\end{document}.

**Exercise 5.1:** Produce the following text with your math exercise file: ‘The
derivative of the indirect function \( f[\{g(x)\}] \) is \( f'[\{g(x)\}]g'(x) \). For
the second derivative of the product of \( f(x) \) and \( g(x) \) one has \( [f(x)g(x)]'' = f''(x)g(x) + 2f'(x)g'(x) + f(x)g''(x) \).’

Note: higher derivatives are made with multiple ’ symbols: \$y'''$ yields \( y''' \).

## 5.2.2 Exponents and indices

Mathematical formulas often contain exponents and indices, characters
that are either raised or lowered relative to the main line of the formula,
and printed in a smaller typeface. Although their mathematical meanings
are different, superscripts and subscripts are typographically the same
things as exponents and indices, respectively. It is even possible that
exponents themselves have exponents or indices, and so on. These are
produced by multiple applications of the raising and lowering operations.

\LaTeX\ and TeX make it possible to create any combination of exponents
and indices with the correct type size in a simple manner: the character
command \(^\) sets the next character as an exponent (raised), while the
character command \_< sets it as an index (lowered).

\[ x^2 \ x'^2 \ a_n \ a_{-n} \ x^n_i \ x^\_n_{-i} \]

When exponents and indices occur together, their order is unimportant.
The last example above could also have been given as \( x_{-i}^n \).

If the exponent or index contains more than one character, the group
of characters must be enclosed in braces \{ \}:

\[ x^{2n} \ x^{\{2n\}} \ x_{2y} \ x_{\{2y\}} \ x_{2}^{\{y\}^2} \ x_{\{2\}^y} \ x_{\{y\}^1} \]

Multiple raisings and lowerings are generated by applying \(^\) and \_< to
the exponents and indices:

\[ A^{x_i^2}_{j^2,km}{\cdot} \ x_{\{i\}^2}^{{\cdot}j} \ x_{\{2\}^n}^{{\cdot}j} \ x_{\{n,m\}}^{{\cdot}j} \]

**Note:** The raising and lowering commands \(^\) and \_< are only permitted in
math mode.
5.2.3 Fractions

Short fractions, especially within a text formula, are best represented using the slash character /, as in \((n+m)/2\) for \((n + m)/2\). For more complicated fractions, the command

\[
\text{\textbackslash frac\{numerator\}\{denominator\}
\]

is employed to write the numerator on top of the denominator with a horizontal fraction line of the right width between them.

\[
\frac{1}{x + y} \quad \text{\textbackslash frac\{1\}\{x+y\}
\]

\[
\frac{a^2 - b^2}{a + b} = a - b \quad \text{\textbackslash frac\{a^2 - b^2\}\{a+b\} = a-b
\]

Fractions may be nested to any depth within one another.

\[
\frac{a}{x-y} + \frac{b}{x+y} \quad \text{\textbackslash frac\{\textbackslash frac\{a\}\{x-y\}\} + \textbackslash frac\{b\}\{x+y\}\}
\]

\[
1 + \frac{a-b}{a+b} \quad \text{\textbackslash frac\{1\} + \textbackslash frac\{a-b\}\{a+b\}\}
\]

\LaTeX{} sets fractions within fractions in a smaller typeface. Section 5.5.2 describes how the automatic type sizes may be overridden if \LaTeX{}'s choice is unsuitable.

5.2.4 Roots

Roots are printed with the command

\[
\text{\textbackslash sqrt\{n\}\{arg\}
\]

as in the example: \(\sqrt[3]{8} = 2\) produces \(\sqrt[3]{8} = 2\). If the optional argument \(n\) is omitted, the square root is generated: \(\sqrt{a}\) yields \(\sqrt{a}\).

The size and length of the root sign are automatically fitted to \(arg\):

\[
\sqrt{x^2 + y^2 + 2xy} = x+y\quad \sqrt{x^2 + y^2 + 2xy} = x+y, \text{ or}
\]

\[
\sqrt[n]{\frac{x^n - y^n}{1 + u^{2n}}} \quad \text{\textbackslash sqrt\{n\}\{\textbackslash frac\{x^n - y^n\}\{1 + u^{2n}\}\}
\]

Roots may be nested inside one another to any depth:

\[
\sqrt[3]{-q + \sqrt{q^2 + p^3}} \quad \text{\textbackslash sqrt\{3\}\{q + \text{\textbackslash sqrt\{q^2 + p^3\}\}}
\]
5.2.5 Sums and integrals

Summation and integral signs are made with the two commands $\sum$ and $\int$, which may appear in two different sizes depending on whether they occur in a text or displayed formula.

Sums and integrals very often possess upper and lower limits. These are printed with the exponent and index commands $^\text{^}$ and $^\text{_}$. The positioning of the limits also depends on whether the formula is in text or displayed.

In a text formula $\sum_{i=1}^n$ and $\int_a^b$ produce $\sum_{i=1}^n$ and $\int_a^b$, whereas in a displayed formula they appear as at the left below:

\[
\sum_{i=1}^n \int_a^b \text{limits immediately following the integral sign: } \int_{x=0}^{x=1} x \text{ limits}\]

The rest of the formula text coming before and after the sum and integral signs is correctly aligned with them.

\[
2 \sum_{i=1}^n a_i \int_a^b f_i(x)g_i(x) \, dx
\]

Two points must be made regarding integrals such as $\int y \, dx$ and $\int f(z) \, dz$.

First, there should be a little extra spacing between the differential operators $\, dx$ and $\, dz$ and the integrands preceding them. This is achieved with the small spacing command $\,$ mentioned in Section 2.7.1, not with a blank which is ignored in math mode. Second, the differential operator should be written upright, not italic, as explained in Section 5.4.10. This is accomplished by setting it within a $\text{mathrm}$ command (Section 5.4.2). Thus $\int y \, , \text{mathrm\{d\}x}$ and $\int f\{z\} \, , \text{mathrm\{d\}z}$ produce the desired results shown above, whereas $\int y \, dx$ and $\int f(z) \, dz$ yield $\int y \, dx$ and $\int f(z) \, dz$.

5.2.6 Continuation dots—ellipsis

Formulas occasionally contain a row of dots $\ldots$, meaning and so on. Simply typing a period three times in a row produces an undesirable result: $\ldots$, that is, the dots are too close together. Therefore LaTeX provides several commands

\begin{verbatim}
\ldots \text{ low dots } \cdots \text{ center dots } \vdots \text{ vertical dots } \ddots \text{ diagonal dots }
\end{verbatim}

to space the dots correctly. The difference between the first two commands is best illustrated by the examples $a_0, a_1, \ldots, a_n$ and $a_0 + a_1 + \cdots + a_n$, which are produced with $\$a_0, a_1, \ldots, a_n\$$ for the first and $\$a_0 + a_1 + \cdots + a_n\$$ for the second.
The command \ldots is also available in normal text mode, whereas
the other three are only allowed in math mode. In text mode, the command
\ldots may be used in place of \ldots with the same effect.

Exercise 5.2: Generate the following output:
The reduced cubic equation $y^3 + 3py + 2q = 0$ has one real and two complex
solutions when $D = q^2 + p^3 > 0$. These are given by Cardan’s formula as

$$y_1 = u + \nu, \quad y_2 = -\frac{u + \nu}{2} + \frac{1}{2} \sqrt{3}(u - \nu), \quad y_3 = -\frac{u + \nu}{2} - \frac{1}{2} \sqrt{3}(u - \nu)$$

where

$$u = \sqrt[3]{-q + \sqrt{q^2 + p^3}}, \quad \nu = \sqrt[3]{-q - \sqrt{q^2 + p^3}}$$

Note: the spacings between the parts of the displayed equations are made with
the spacing commands \quad and \qquad.

Exercise 5.3: Select the option fleqn in the document class command and in-
clude the specification \setlength{\mathindent}{2cm} in the preamble. Redo
the three $y$ equations above, each as a separate displayed formula, using the
equation environment instead of displaymath or \[
\ldots\] brackets.

Exercise 5.4: Create the following text:
Each of the measurements $x_1 < x_2 < \cdots < x_r$ occurs $p_1, p_2, \ldots, p_r$ times. The
mean value and standard deviation are then

$$x = \frac{1}{n} \sum_{i=1}^{r} p_i x_i, \quad s = \sqrt{\frac{1}{n} \sum_{i=1}^{r} p_i (x_i - x)^2}$$

where $n = p_1 + p_2 + \cdots + p_r$.

Exercise 5.5: Although this equation looks very complicated, it should not
present any great difficulties:

$$\int \frac{\sqrt{(ax + b)^3}}{x} \, dx = \frac{2\sqrt{(ax + b)^3}}{3} + 2b\sqrt{ax + b} + b^2 \int \frac{dx}{x\sqrt{ax + b}}$$

The same applies to $\int_{a}^{b} (dx / \sqrt{x}) = \frac{3}{2} (8^{2/3} + 1^{2/3}) = 15/2$.

5.3 Mathematical symbols

There is a very wide range of symbols used in mathematical text, of which
only a few are directly available from the keyboard. \LaTeX provides many of
the mathematical symbols that are commonly used. They are called with
the symbol name prefixed with the \ character. The names themselves
are derived from their mathematical meanings.
5.3. Mathematical symbols

5.3.1 Greek letters

Lower case letters

\[ \alpha \ \beta \ \gamma \ \delta \ \epsilon \ \zeta \ \eta \ \theta \ \vartheta \ \iota \ \kappa \ \lambda \ \mu \ \nu \ \xi \ \omicron \ \pi \ \varpi \ \rho \ \varrho \ \sigma \ \varsigma \ \tau \ \upsilon \ \phi \ \varphi \ \chi \ \psi \ \omega \]

Upper case letters

\[ \Gamma \ \Delta \ \Theta \ \Lambda \ \Xi \ \Pi \ \Psi \ \Omega \]

The Greek letters are made simply by putting the command character \ before the name of the letter. Upper case (capital) letters are distinguished by capitalizing the first letter of the name. Greek letters that do not appear in the above list are identical with some corresponding Latin letter. For example, upper case \( \rho \) is the same as Latin P and so needs no special symbol.

\LaTeX{} normally sets the upper case Greek letters in Roman (upright) type within a mathematical formula. If they need to be in italics, this can be brought about with the \texttt{math alphabet command \texttt{mathnormal}}: \texttt{$\mathnormal{\Gamma \Pi \Phi}$} appears as \( \Gamma \Pi \Phi \).

Greek letters may only be used in math mode. If they are needed in normal text, the command must be enclosed in \$...\$ signs.

5.3.2 Calligraphic letters

The following 26 \textit{calligraphic} letters may also be used in math formulas:

\( A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z \) These are called with the math alphabet command \texttt{mathcal}:

\texttt{$\mathcal{A, B, C, ..., Z}$}

5.3.3 Binary operators

Two mathematical quantities combined with one another to make a new quantity are said to be joined by a \textit{binary operation}. The symbols that are available for use as binary operators are
Chapter 5. Mathematical Formulas

\pm \ \cap \ \circ \ \bigcirc \ \mp \ \cup \ \bullet \ \Box \ \times \ \uplus \ \Diamond \ \div \ \sqcap \ \triangledown \ \cdot \ \bigtriangleup \ \dagger \ \ddagger \ \setminus \ \star \ \wedge \ \models \ \div \ \sqcup \ \rhd \ \star \ \wedge \ \models \ \div \ \sqcup \ \rhd

The underlined symbol names in the above and following tables are only available if one of the packages \texttt{latexsym} (Section B.5.3) or \texttt{amsfonts} (Section 12.4.4) has been loaded.

5.3.4 Relations and their negations

When two mathematical quantities are compared, they are connected by a relation. The different types of relational symbols for the various comparisons are

\leq \ \leq \ \geq \ \geq \ \neq \ \sim \ \ll \ \gg \ \subset \ \supset \ \approx \ \asymp \ \subseteq \ \supseteq \ \doteq \ \equiv \ \sqsubset \ \sqsupset \ \doteq \ \equiv \ \in \ \ni \ \prec \ \succ \ \vdash \ \dashv \ \models \ \models \ \perp \ \parallel \ \mid

A number of the above symbols may be called by more than one name. For example, \leq may be produced with either \textbackslash le or \textbackslash leq.

The opposite, or negated, meaning of the relation is indicated in mathematics with a slash / through the symbol: = and \neq mean equals and not equals. One may put a slash through any of the above symbols by prefixing its name with \textbackslash not. Thus \textbackslash not\textbackslash in yields \notin. The same is true for the keyboard characters: \textbackslash not=, \textbackslash not>, and \textbackslash not< produce \# and \notin. For \textbackslash not= there is also the special command \textbackslash neq producing \neq which is a symbol on its own and not the combined \#.

The following symbols may be negated in this manner. Note that the last two, \textbackslash not\textbackslash in and \textbackslash not in, are not exactly the same: \notin and \notin. The latter form is the preferred one.
5.3. Mathematical symbols

5.3.5 Arrows and pointers

Mathematical manuscripts often contain arrow symbols, also called pointers. The following arrow symbols are available:

- \leftarrow \gets \leftarrow^{-} \longleftrightarrow \leftarrow \rightarrow to \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \right.
5.3.7 Symbols with two sizes

The following symbols are printed in different sizes depending on whether they appear in text or displayed formulas:

\[ \sum \bigcap \bigcup \bigodot \int \bigvee \bigoplus \prod \bigvee \biguplus \coprod \bigwedge \]

The symbols \( \int \) and \( \sum \) have already been introduced in Section 5.2.5. There it was shown how these symbols may take on upper and lower limits; in the same way, all the above symbols may also be assigned upper and lower limits using the shifting commands \( \hat{\ } \) and \( \_ \). The positioning of the limits varies for some symbols depending on whether they occur in text or displayed formulas. As indicated in Section 5.2.5, the command \( \text{\textbackslash limits} \) forces the limits to be written above and below the symbol where they would otherwise be placed beside it. Similarly the complementary command \( \text{\textbackslash nolimits} \) sets them beside the symbol when the standard positioning is above and below.

\[
\oint_0^\infty \prod_{\nu=0}^n \]

5.3.8 Function names

The universal standard for mathematical formulas is to set variable names in italics but the names of functions in Roman. If one were simply to write the function names \( \sin \) or \( \log \) in math mode, \LaTeX{} would interpret these as variables \( s \ i \ n \) and \( l \ o \ g \) and write them as \( \sin \) and \( \log \). In order to tell \LaTeX{} that a function name is wanted, it is necessary to enter a command consisting of the backslash \( \text{\textbackslash} \) plus the function name. The following names are recognized by \LaTeX{}:

\[
\arccos \cosh \det \inf \limsup \Pr \tan
\arcsin \cot \dim \ker \ln \sec \tanh
\arctan \coth \exp \lg \log \sin
\arg \csc \gcd \lim \max \sinh
\cos \deg \hom \liminf \min \sup
\]
Some of these functions may also appear with limits attached to them. This is easily achieved by means of the subscript command after the name of the function: \( \lim_{x \to \text{infty}} \) yields \( \lim_{x \to \infty} \) in text formulas and \( \lim_{x \to \infty} \) in displayed formulas.

The following function names may accept a limit with the lowering (index) command \( _- \):

- \det
- \gcd
- \inf
- \lim
- \liminf
- \limsup
- \max
- \min
- \Pr
- \sup

Finally, there are the function commands \bmod and \pmod{arg}, both of which produce the function \textit{mod} in one of two forms:

\[
\begin{align*}
& \text{\$ a \bmod b \$ } \Rightarrow \ a \mod b \\
& \text{\$ y \pmod{a+b} \$ } \Rightarrow \ y \ (\text{mod } a + b).
\end{align*}
\]

With AMS-\LaTeX{} (Section 12.2.5) it is possible to define additional function names.

## 5.3.9 Mathematical accents

The following mathematical accents are available within math mode:

- \( \hat{a} \)
- \( \breve{a} \)
- \( \grave{a} \)
- \( \bar{a} \)
- \( \check{a} \)
- \( \acute{a} \)
- \( \tilde{a} \)
- \( \vec{a} \)
- \( \dot{a} \)
- \( \ddot{a} \)
- \( \mathring{a} \)

The letters \( i \) and \( j \) should be printed without their dots when they are given an accent. To accomplish this, type the symbols \imath and \jmath instead of the letters, as in $\vec{\imath} + \tilde{\jmath}$:

There are wider versions of \hat and \tilde available with the names \widehat and \widetilde. In this way, these accents may be placed over parts of a formula:

\[
\begin{align*}
& \sqrt{1-x} = \sqrt{\hat{y}} \quad \text{\$\widehat{1-x}=\hat{-y}$}\$ \\
& \sqrt{xyz} \quad \text{\$\widetilde{xyz}$}
\end{align*}
\]

**Exercise 5.6:** The union of two sets \( A \) and \( B \) is the set of all elements that are in at least one of the two sets, and is designated as \( A \cup B \). This operation is commutative \( A \cup B = B \cup A \) and associative \( (A \cup B) \cup C = A \cup (B \cup C) \). If \( A \subseteq B \), then \( A \cup B = B \). It then follows that \( A \cup \emptyset = A \), \( A \cup \{\emptyset\} = A \) and \( J \cup A = J \).

**Exercise 5.7:** Applying l'Hôpital's rule, one has

\[
\begin{align*}
\lim_{x \to 0} \frac{\ln \sin \pi x}{\ln \sin x} &= \lim_{x \to 0} \frac{\pi \cos \pi x}{\sin \pi x} = \lim_{x \to 0} \frac{\pi \tan x}{\cos x} = \lim_{x \to 0} \frac{\pi / \cos^2 x}{\pi / \cos^2 \pi x} = \lim_{x \to 0} \frac{\cos^2 \pi x}{\cos^2 x} = 1
\end{align*}
\]
Chapter 5. Mathematical Formulas

Exercise 5.8: The gamma function $\Gamma(x)$ is defined as

$$\Gamma(x) \equiv \lim_{n \to \infty} \prod_{\nu=0}^{n-1} \frac{n! n^{x-1}}{x+\nu} = \lim_{n \to \infty} \frac{n! n^{x-1}}{x(x+1)(x+2) \cdots (x+n-1)} \equiv \int_0^\infty e^{-t} t^{x-1} \, dt$$

The integral definition is valid only for $x > 0$ (2nd Euler integral).

Exercise 5.9: Remove the option `fleqn` from the document class command in Exercise 5.3 and redo the output.

Exercise 5.10:

$\alpha \vec{x} = \vec{x} \alpha, \quad \alpha \beta \vec{x} = \beta \alpha \vec{x}, \quad (\alpha + \beta) \vec{x} = \alpha \vec{x} + \beta \vec{x}, \quad \alpha (\vec{x} + \vec{y}) = \alpha \vec{x} + \alpha \vec{y}.$

$\vec{x} \vec{y} = \vec{y} \vec{x}$ but $\vec{x} \times \vec{y} = -\vec{y} \times \vec{x}, \quad \vec{x} \vec{y} = 0$ for $\vec{x} \perp \vec{y}, \quad \vec{x} \times \vec{y} = 0, \quad$ for $\vec{x} \parallel \vec{y}.$

Exercise 5.11: Reproduce Equations 5.1 and 5.2 from the next section.

5.4 Additional elements

The math elements described in the previous sections already permit the construction of very complex formulas, such as

$$\lim_{x \to 0} \frac{\sqrt{1+x} - 1}{x} = \lim_{x \to 0} \frac{(\sqrt{1+x} - 1)(\sqrt{1+x} + 1)}{x(\sqrt{1+x} + 1)} = \lim_{x \to 0} \frac{1}{\sqrt{1+x} + 1} = \frac{1}{2} \quad (5.1)$$

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = 0 \quad \Rightarrow \quad U_M = \frac{1}{4\pi} \oint_{\Sigma} \frac{1}{r} \frac{\partial U}{\partial n} \, ds - \frac{1}{4\pi} \oint_{\Sigma} \frac{\partial}{\partial n} U \, ds \quad (5.2)$$

$$I(z) = \sin\left(\frac{\pi}{2} z^2\right) \sum_{n=0}^{\infty} \frac{(-1)^n \pi^{2n}}{1 \cdot 3 \cdots (4n+1)} z^{4n+1} - \cos\left(\frac{\pi}{2} z^2\right) \sum_{n=0}^{\infty} \frac{(-1)^n \pi^{2n+1}}{1 \cdot 3 \cdots (4n+3)} z^{4n+3} \quad (5.3)$$

By reading the formulas from left to right there should be no difficulty in reconstructing the text that produced them. For example, the last equation is generated with

\begin{verbatim}
\begin{equation}
I(z) = \sin\left( \frac{\pi}{2} z^2 \right) \sum_{n=0}^{\infty} \frac{(-1)^n \pi^{2n}}{1 \cdot 3 \cdots (4n+1)} z^{4n+1} - \cos\left( \frac{\pi}{2} z^2 \right) \sum_{n=0}^{\infty} \frac{(-1)^n \pi^{2n+1}}{1 \cdot 3 \cdots (4n+3)} z^{4n+3}
\end{equation}
\end{verbatim}

The above examples were made using the equation environment instead of the `displaymath` environment or its abbreviated form `\[\ldots\]`, which has the effect of adding the equation numbers automatically. In the document classes `book` and `report`, equations are sequentially numbered within the chapter, the number being preceded by the chapter number and set within parentheses ( ), as illustrated above. For document class
5.4. Additional elements

By default the equation number appears right justified and vertically centered with the equation. If there is not enough room for it on the same line, it is printed right justified below the equation. If the document class option `leqno` has been selected, the equation numbers are set left justified for the entire document.

The automatic numbering of equations means that the author may not know at the time of writing just what the equation number is. The LaTeX cross-reference system described in Section 9.2.1 has already been explained for referring to section numbers (Section 3.3.3) and may also be used for equation numbers. By including a command `\label{name}` within the `equation` environment, one can print the unknown equation number in the text with the command `\ref{name}`, where `name` is a keyword consisting of any combination of letters, numbers, or symbols.

Examining Equation 5.3 more closely, one notices that the two parentheses pairs ( ) in $\cos()$ and $\sin()$ could be somewhat larger. Furthermore, this equation just fills the line width, and if it were any longer, it would have to be broken at some appropriate spot and the parts positioned in a meaningful way relative to one another. None of the math elements described so far can accomplish these requirements.

Even something so simple as including some normal text within a formula has not yet been mentioned. The rest of this section addresses these problems.

Finally, there are times when one is not happy with the sizes that TeX has chosen, as for example in the last integral of Equation 5.2 where $\partial \frac{1}{r}$ would be more desirable than $\partial \frac{1}{r}$. This and other formatting aids, such as adjusting horizontal spacing between parts of formulas, are dealt with in Section 5.5.

5.4.1 Automatic sizing of bracket symbols

Mathematics often contains bracketing symbols, usually in pairs that enclose part of the formula. When printed, these bracket symbols should be the same size as the included partial formula. LaTeX provides a pair of commands

\begin{verbatim}
\left lbrack sub_form \right rbrack
\end{verbatim}

\left lbrack sub_form \right rbrack

\begin{verbatim}
\end{verbatim}

to accomplish this. The command `\left` is placed immediately before the opening (left hand) bracket symbol `lbrack` while `\right` comes just before the closing (right hand) symbol `rbrack`. 
The commands \texttt{\textbackslash left} and \texttt{\textbackslash right} must appear as a pair. For every \texttt{\textbackslash left} command there must be a corresponding \texttt{\textbackslash right} command somewhere afterwards. The pairs may be nested. The first \texttt{\textbackslash left} is paired with the last \texttt{\textbackslash right}; the following \texttt{\textbackslash left} with the second last \texttt{\textbackslash right}, and so on. There must be the same number of \texttt{\textbackslash right} as \texttt{\textbackslash left} commands in a nesting.

The corresponding bracket symbols \texttt{lbrack} and \texttt{rbrack} may be perfectly arbitrary and do not need to be a logical pair.

\[
\vec{x} + \vec{y} + \vec{z} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \left[ \begin{array}{r} x \\ y \end{array} \right] \]

This set of brackets is admittedly unusual but permissible.

Sometimes a formula contains only a single opening or closing bracket without a corresponding counterpart. However, the \texttt{\textbackslash left}...\texttt{\textbackslash right} commands must still be given as a pair, but with a period ‘.’ as an \textit{invisible} bracket symbol.

\[
y = \left\{ \begin{array}{l} -1 : x < 0 \\ 0 : x = 0 \\ +1 : x > 0 \end{array} \right. \]

The array environment in the above example is described in Section 4.8.1 and produces a table in math mode.

The \texttt{\textbackslash left}...\texttt{\textbackslash right} commands may be applied to a total of 22 different symbols. These are

\[
( ( ) ) \quad \lfloor \quad \rfloor \quad \lceil \quad \rceil \quad \{ \quad \}\quad \langle \quad \rangle \quad \uparrow \quad \Downarrow \quad \updownarrow
\]

For example, \texttt{\textbackslash left}...\texttt{\textbackslash right}\texttt{|} produces two vertical bars adjusted in height to contain the enclosed formula text.

\textbf{Exercise 5.12:} In Equation 5.3, generate \( \cos \left( \frac{\pi}{2} z^2 \right) \) and \( \sin \left( \frac{\pi}{2} z^2 \right) \) instead of \( \cos (\frac{\pi}{2} z^2) \) and \( \sin (\frac{\pi}{2} z^2) \).
5.4.2 Ordinary text within a formula

It is often necessary to include some *normal* text within a formula, for example single words such as *and*, *or*, *if*, and so on. In this case one must switch to LR mode (Section 4.7.1) while staying in math mode. This is carried out with the command \texttt{\textbackslash \mbox{normal text}} given inside the formula, together with horizontal spacing commands such as \texttt{\textbackslash qquad} or \texttt{\textbackslash hspace}. For example:

\[
X_n = X_k \quad \text{if and only if} \quad Y_n = Y_k \quad \text{and} \quad Z_n = Z_k
\]

In order to set a longer piece of text beside a displayed formula, as in some of the above examples, it is more appropriate to put both the formula and the text in their own parboxes or minipages, placed side by side with the proper vertical positioning.

On the other hand, if letters from text fonts are required as mathematical symbols, they should be entered with the *math alphabet commands*:

\[
\texttt{\textbackslash \mathcal{B}^0(x) \quad T^i_j} \quad \texttt{\mathbf{B}^0(x)} \quad \texttt{\mathsf{T}^i_j}
\]

The command \texttt{\textbackslash \textnormal} in Section 5.3.1 also belongs to this group. The difference between it and \texttt{\textit} is that it sets its argument in the regular *math* italic font, while the latter uses the normal *text* italic. The letters are the same, but the spacing is different.

\[
\texttt{\mathnormal{differ} \ne \textit{differ}}
\]

All the math alphabet commands set their text in math mode, which means that spaces are ignored as usual. This is not the case for text placed in an \texttt{\mbox}.

5.4.3 Matrices and arrays

\[
\begin{array}{cccc}
  a_{11} & a_{12} & \cdots & a_{1n} \\
  \vdots & \vdots & \ddots & \vdots \\
  a_{n1} & a_{n2} & \cdots & a_{nn}
\end{array}
\]

Structures like the one at the left are the basis for matrices, determinants, system of equations, and so on. They will all be referred to here as *arrays*. 
Arrays are produced by means of the `array` environment, whose syntax and construction are described in Section 4.8.1 on tables. The `array` environment generates a table in math mode, that is, the column entries are interpreted as formula text. For example:

\[
\begin{array}{*{3}{c@{:+:}}c@{;=;}}
   a_{11}x_1 & a_{12}x_2 & \cdots & a_{1n}x_n & b_1 \\
   a_{22}x_1 & a_{22}x_2 & \cdots & a_{2n}x_n & b_2 \\
   \multicolumn{5}{c}{\dotfill} \\
   a_{n1}x_1 & a_{n2}x_2 & \cdots & a_{nn}x_n & b_n \end{array}
\]

As a reminder of the table construction elements (Section 4.8.1): \texttt{@\{t\}} inserts the contents of \texttt{t} between the adjacent columns. In the above example, this is \texttt{:+;} and \texttt{:+;}. The commands \texttt{:} and \texttt{;} have not yet been introduced but they produce small horizontal spacing in math mode (Section 5.5.1). \texttt{*\{3\}c\{c@{:+;}\}c@{=;\}c} is an abbreviation for three repetitions of the column definition \texttt{c@{:+;}}. \texttt{c} defines the column to be one of centered text. \texttt{\textbackslash multicol\{5\}c\{\textbackslash dotfill\}} says that the next five columns are to be merged and replaced by one with centered text. The command \texttt{\textbackslash dotfill} fills the column with dots.

It is possible to nest `array` environments:

\[
\left( \begin{array}{c}
   x_{11} & x_{12} \\
   x_{21} & x_{22} \\
   y \\
   z
\end{array} \right) \quad \left\lfloor \begin{array}{c}
  \text{c@{:+;}} \text{c@{=;\}}
  \text{c@{:+;}} \text{c@{=;\}}
  \text{c@{:+;}} \text{c@{=;\}}
  \text{c@{:+;}} \text{c@{=;\}}
\end{array} \right\rfloor
\]

The outermost array consists of one column with centered text (c). The first entry in this column is also an array, with two centered columns. This array is surrounded left and right by vertical lines with adjusted sizes.

The `array` environment is structurally the same as a vertical box. This means that it is treated as a single character within the surrounding environment, so that it may be coupled with other symbols and construction elements.
\[ \sum_{1 < q_1 < q_2 < \cdots < q_{n-k}} \Delta_{p_1 p_2 \cdots p_{n-k}} \sum_{q_1 < q_2 < \cdots q_k} \left| \begin{array}{llcl} a_{q_1q_1} & a_{q_1q_2} & \cdots & a_{q_1q_k} \\ a_{q_2q_1} & a_{q_2q_2} & \cdots & a_{q_2q_k} \\
\multicolumn{4}{c}{\dotfill} \\
a_{q_kq_1} & a_{q_kq_2} & \cdots & a_{q_kq_k} \end{array} \right| \]

In this example, an array environment is used as an index on the $\Delta$. However, the indices appear too large with respect to the rest of the formula. Section 5.4.6 presents a better solution for array indices.

As for all table environments, an optional vertical positioning parameter $b$ or $t$ may be included with the array environment. The syntax and results are described in Sections 4.7.3 and 4.8.1. This argument is included only if the array is to be positioned vertically relative to its top or bottom line rather than its center.

We suggest that the user try to deduce how the various arrays are structured with the help of the generating text on the right.

**Exercise 5.13:** The solution for the system of equations

\[ F(x, y) = 0 \quad \text{and} \quad \begin{vmatrix} F_{xx} & F_{xy} & F_x' \\ F_{yx} & F_{yy} & F_y' \\ F_x' & F_y' & 0 \end{vmatrix} = 0 \]

yields the coordinates for the possible inflection points of $F(x, y) = 0$.

**Note:** The above displayed formula consists of two sub-formulas, between which the word ‘and’ plus extra spacing of amount \quad are inserted. Instead of enclosing the array environment within size-adjusted vertical lines with \left[...\right], one may use a formatting argument {\ldots} (Section 4.8.1) to produce the vertical lines. Such a structure is called a determinant in mathematics.

**Exercise 5.14:** The shortest distance between two straight lines represented by the equations

\[ \frac{x - x_1}{l_1} = \frac{y - y_1}{m_1} = \frac{z - z_1}{n_1} \quad \text{and} \quad \frac{x - x_2}{l_2} = \frac{y - y_2}{m_2} = \frac{z - z_2}{n_2} \]
is given by the expression

\[ \pm \begin{vmatrix} x_1 - x_2 & y_1 - y_2 & z_1 - z_2 \\ l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \end{vmatrix} \]

\[ \sqrt{\left| \begin{vmatrix} l_1 & m_1 & \mid l_2 & m_2 & n_1 \\ m_1 & n_1 & \mid m_2 & n_2 & l_1 \end{vmatrix}^2 + \mid m_1 & n_1 & \mid m_2 & n_2 & l_1 \end{vmatrix}^2 + \mid n_1 & l_1 & \mid n_2 & l_2 & m_1} \]

If the numerator is zero, the two lines meet somewhere.

Note: we do not recommend using \{|cc|\} in the formatting argument of the three determinants in the denominator under the root sign. Here the \texttt{\left|...\right|} pair should be applied. Try out both possibilities for yourself and compare the results.

**Exercise 5.15:** Laurent expansion: using \( c_n = \frac{1}{2\pi i} \oint_{|\zeta-a|=\varepsilon} f(\zeta) \frac{d\zeta}{(\zeta - a)^n} \) for every function \( f(z) \) the following representation is valid (\( n = 0, \pm 1, \pm 2, \ldots \))

\[ f(x) = \sum_{n=-\infty}^{+\infty} c_n (z - a)^n = \begin{cases} 
0 + c_1(z - a) + c_2(z - a)^2 + \cdots + c_n(z - a)^n + \cdots \\
+ c_{-1}(z - a)^{-1} + c_{-2}(z - a)^{-2} + \cdots \\
+ c_{-n}(z - a)^{-n} + \cdots 
\end{cases} \]

Tip: the right-hand side of the equation can be created with an array environment consisting of only one column. What is its formatting argument?

### 5.4.4 Lines above and below formulas

The commands

\texttt{\overline{sub_form}} \textit{and} \texttt{\underline{sub_form}}

can be used to draw lines over or under a formula or sub-formula. They may be nested to any level:

\[ \overline{a^2 + xy + \overline{z}} \]

\[ \overline{\underline{a}^2 + \overline{\underline{xy}}} + \overline{\underline{z}} \]

The command \texttt{\underline} may also be employed in normal text mode to underline text, whereas \texttt{\overline} is allowed only in math mode.

Exactly analogous to these are the two commands

\texttt{\overbrace{sub_form}} \textit{and} \texttt{\underbrace{sub_form}}

which put horizontal curly braces above or below the sub-formula.

\[ \overbrace{a + b + c + d} \overbrace{a + \underbrace{b+c} + d} \]

In displayed formulas, these commands may have exponents or indices attached to them. The (raised) exponent is set above the \texttt{overbrace} while the (lowered) index is placed below the \texttt{underbrace}. 
Exercise 5.16: The total number of permutations of $n$ elements taken $m$ at a time (symbol $P_m^n$) is

$$\prod_{i=0}^{m-1} (n-i) = n(n-1)(n-2)\ldots(n-m+1) = \frac{n!}{(n-m)!}$$

5.4.5 Stacked symbols

The command

$$\stackrel{upper \ sym}{lower \ sym}$$

places the symbol $upper \ sym$ centered on top of $lower \ sym$, whereby the symbol on top is set in a smaller typeface.

$$\vec{x} \stackrel{\text{def}}{=} (x_1, \ldots, x_n) \quad \text{vec}\{x\} \ \stackrel{\text{def}}{=} \{x_1, \ldots\}$$

$$A \stackrel{\alpha'}{\longrightarrow} B \stackrel{\beta'}{\longleftarrow} C \quad A \ \stackrel{\alpha'}{\longrightarrow} B \ \stackrel{\beta'}{\longleftarrow} C$$

By making use of math font size commands (Section 5.5.2) it is possible to construct new symbols with this command. For example, some authors prefer to have the $\leq$ symbol appear as $\leq$ instead of $\leq$. This is achieved by combining $<$ and $=$ with $\stackrel{\textstyle<}{=}$. If the command $\text{textstyle}$ were omitted, the symbol would be printed as $\leq$.

5.4.6 Additional \TeX commands for math

The \TeX math commands $\atop$ and $\choose$ are useful additions to the set of commands and may be applied within any \LaTeX document. (In fact, all \TeX math commands except $\eqalign$, $\eqalignno$, and $\leqalignno$ may be used in a \LaTeX manuscript.) Their syntax is

\begin{verbatim}
{top \atop bottom}
{top \choose bottom}
\end{verbatim}

Both commands produce a structure that looks like a fraction without the dividing line. With the $\choose$ command, this structure is also enclosed within round brackets (in mathematics this is called a binomial coefficient).

$$\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}$$
\[
\prod_{j \geq 0} \left( \sum_{k \geq 0} a_{jk}z^k \right) = \sum_{n \geq 0} z^n \left( \sum_{k_0, k_1, \ldots \geq 0} \begin{array}{c} k_0 \\ k_0 + k_1 + \cdots = 0 \end{array} a_{0k_0} a_{1k_1} \ldots \right)
\]

Similar structures can be generated with the \LaTeX environments

\begin{array}{c}
upper line \\
lower line \end{array}
(atop)
\left(\begin{array}{c}
upper \\
lower \end{array}\right)(choose)

The above \LaTeX commands may also be employed to produce small matrices within text formulas, such as \( \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \) or \( \begin{pmatrix} a & b & c \\ l & m & n \end{pmatrix} \). Here the first matrix was typed in with

\$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\$

and the second with

\$\left(\begin{pmatrix} a & \atop \atop \atop l \end{pmatrix} \begin{pmatrix} b & \atop \atop \atop m \end{pmatrix} \begin{pmatrix} c & \atop \atop \atop n \end{pmatrix}\right)\$

The syntax of these Plain \LaTeX commands is radically different from that normally used by \LaTeX. See page 189 for a way to correct this.

5.4.7 Multiline equations

A multiline equation is one that is developed over several lines in which the relation symbols (for example, = or \(\leq\)) in each line are all vertically aligned with each other. For this purpose, the environments

\begin{eqnarray*}
\begin{array}{c}
line_1 \\\ \ldots \\\ line_n \end{array} 
\end{eqnarray*}

\begin{eqnarray*}
\begin{array}{c}
line_1 \\\ \ldots \\\ line_n \end{array} \end{eqnarray*}

are used to set several lines of formulas or equations in displayed math mode. The individual lines of the equation or formula are separated from one another by \(\backslash\). Each entry line has the form
5.4. Additional elements

When printed, all the left formulas appear right justified in a left column, the right formulas left justified in a right column, and the mid formulas centered in between. The column separation character & designates the various parts of the formula. Normally the mid formula is a single math character, the relation operator mentioned above. The individual lines thus have the same behavior as they would in a \begin{array} \{rc|l\} \end{array} environment.

The difference between the array and eqnarray environments is that in the latter the lines are set as displayed formulas. This means that for those symbols listed in Section 5.3.7 the larger form will be selected, and that the numerator and denominator of fractions will be in normal size. On the other hand, for the array environment the column entries will be set as text formulas, the smaller form of these symbols will be chosen and the parts of the fraction will appear in a smaller type size.

The standard form of the eqnarray environment adds an automatic sequential equation number, which is missing in the *-form. To suppress the equation number for a single line in the standard form, add the command \nonumber just before the line termination \.

The equation numbers may be referred to in the text with the command \ref\{name\} once the keyword name has been assigned to the equation number with the \label\{name\} command somewhere in the line of the equation. See Section 9.2.1 for more details.

Examples:

$$(x + y)(x - y) = x^2 - xy + xy - y^2$$

$$(x + y)^2 = x^2 + 2xy + y^2$$

\begin{eqnarray}
(x+y)(x-y) & = & x^2-xy+xy-y^2 \nonumber\ \\
& = & x^2 - y^2 \ \\
(x+y)^2 & = & x^2 + 2xy + y^2
\end{eqnarray}

\begin{eqnarray*}
x_{n+1} + \cdots + x_{n+t-1}u_t & = & x_nu_1 + (ax_n + c)u_2 + \cdots \\
& & + \left(a^{t-1}x_n + c(a^{t-2} + \cdots + 1)\right)u_t \\
& = & (u_1 + au_2 + \cdots + a^{t-1}u_t)x_n + h(u_1,\ldots,u_t)
\end{eqnarray*}
The second example requires some explanation. In the second line there is a \( \left( \ldots \right) \) pair for the automatic sizing of the (. Such a pair may only appear within a single line; that is, it may not be broken by the \( \backslash \) line terminator! If automatic bracket sizing is to occur in a multiline equation, it may only be found within a single line.

If bracket pairs must appear on different lines, one can try using the construction \( \left( \cdots \right. \backslash \left. \cdots \right) \). In the first line, the \( \left( \) is paired with the invisible bracket \( \right. \), while the second begins with the invisible \( \left. \) which is paired with the closing \( \right) \). However, this will only work satisfactorily if both parts of the equation have roughly the same height so that the two automatic sizings yield much the same results. Section 5.5.3 describes how to select bracket sizes manually in the event that the automatic method fails.

The + sign at the beginning of the second line also requires a remark. The signs + and − have two meanings in mathematics: between two math quantities they act as a coupling (binary operator), but coming before a math symbol they serve as a sign designation (positive or negative). \LaTeX{} stresses this difference by inserting different spacing in the two cases (for example, compare \( +b \) with \( a + b \)).

\[
y = a + b + c + d + e + f + g + h + i + j
\]

If a long formula is broken into several lines and one of them begins with + or −, \LaTeX{} regards it as a sign designation and moves it closer to the next character.

The solution is to introduce an invisible character of zero width at the beginning of such a line. This may be the empty structure \{\}. Compare the effects of \&+e+f+g and \&{h+i+j} in the above equation.

Since the extra spacing is always inserted between + and an opening parenthesis (, it was not necessary to give the \{\} at the beginning of the second line of the eqnarray* example on the previous page.

Occasionally it is better to break long multiline equations as follows:

\[
\begin{eqnarray*}
\lefteqn{w+x+y+z =} \\
& & a + b + c + d + e + f + g + h + i + j + k + l
\end{eqnarray*}
\]

The command \( \texttt{\lefteqn{w+x+y+z =}} \backslash \) in the first line has the effect that the contents of the argument are indeed printed out, but that \LaTeX{} considers them to have zero width. The left-hand column then contains only intercolumn spacing, which produces the indentation for the rest of the lines.

The indentation depth may be altered by inserting \texttt{\hspace{\textit{depth}}} between the \texttt{\lefteqn{...}} and the \( \backslash \) line termination. A positive value for \textit{depth} increases the indentation, a negative value decreases it.
Exercise 5.17: The following equations are to be broken as shown:

\[
\begin{align*}
\arcsin x &= -\arcsin(-x) = \frac{\pi}{2} - \arccos x = \left[ \arccos \sqrt{1-x^2} \right] \\
&= \arctan \frac{x}{\sqrt{1-x^2}} = \left[ \arccot \sqrt{1-x^2} \right] \\
\end{align*}
\]

\[f(x+h, y+k) = f(x, y) + \left\{ \frac{\partial f(x, y)}{\partial x} h + \frac{\partial f(x, y)}{\partial y} k \right\} + \frac{1}{2} \left\{ \frac{\partial^2 f(x, y)}{\partial x^2} h^2 + 2 \frac{\partial^2 f(x, y)}{\partial x \partial y} kh + \frac{\partial^2 f(x, y)}{\partial y^2} k^2 \right\} + \frac{1}{6} \{ \cdots \} + \cdots + \frac{1}{n!} \{ \cdots \} + R_n \]

Note on possible error messages:

Long formulas with many sets of logical brackets, especially deeply nested ones, will almost always contain errors at first. The cause is often brackets that have been incorrectly ordered or overlooked.

If \LaTeX\ produces error messages during formula processing, which the beginner is not able to interpret correctly (error messages are described in detail in Appendix C), he or she should check the bracket pairing very carefully in the formula text. Some text editors can do the search for the matching bracket, which greatly simplifies the task.

If the error is not found in this way, one can try pressing the return key at the error message in order to proceed with the processing. The resulting printout can indicate where things may have gone wrong.

Exercise 5.18: The eqnarray environment inserts additional spacing where the column separation character & appears. This is undesirable when an equation is to be broken and aligned on + or − within a long summation. For example:

The inverse function of the polynomial expansion \( y = f(x) = ax + bx^2 + cx^3 + dx^4 + ex^5 + fx^6 + \cdots \) \((a \neq 0)\) begins with the elements

\[
\begin{align*}
x &= \varphi(y) = \frac{1}{a} y - \frac{b}{a^3} y^2 + \frac{1}{a^5} (2b^2 - ac) y^3 \\
&\quad + \frac{1}{a^7} (5abc - z^2 d - fb^3) y^4 \\
&\quad + \frac{1}{a^9} (6a^2 bd + 3a^2 c^2 + 14b^4 - a^3 e - 21ab^2 c) y^5 \\
&\quad + \frac{1}{a^{11}} (7a^3 be + 7a^3 cd + 84ab^3 c - a^4 f - 28a^2 b^2 d - 28a^2 bc^2 - 43b^5) y^6 + \cdots
\end{align*}
\]

Select a value for the declaration \texttt{\setlength{\arraycolsep}{(Section 4.8.2)}} such that the distance between the + and − and the break points are as near as possible to those in the rest of the formula.
5.4.8 Framed or side-by-side formulas

Displayed formulas or equations may be put into vertical boxes of appropriate width, that is, in a \parbox command or minipage environment. Within the vertical box, the formulas are horizontally centered or left justified with indentation \mathindent according to the selected document class option.

Vertical boxes may be positioned relative to one another just like single characters (Sections 4.7.3 and 4.7.7). In this way the user may place displayed formulas or equations side by side.

\[ \alpha = f(z) \quad (5.8) \]
\[ \beta = f(z^2) \quad (5.9) \]
\[ \gamma = f(z^3) \quad (5.10) \]
\[ x = \alpha^2 - \beta^2 \]
\[ y = 2\alpha\beta \]

The left-hand set of equations is set in a \parbox of width 4 cm, the right-hand set in one of width 2.5 cm, while this text is inside a minipage of width 4.5 cm.

The actual set of equations is produced here in the eqnarray* environment, within a vertical box of width 10 cm, followed by an empty eqnarray environment in a box of width 1 cm that generates the equation number. Both boxes are vertically aligned along their center lines.

Emphasizing formulas by framing requires no new construction elements. It is sufficient to put them in an \fbox (Section 4.7.7). Text formulas [a + b] are simply framed with \fbox{$a+b$}. For displayed formulas, \displaystyle (Section 5.5.2) must be issued, else they are set as text formulas.

\[ \int_0^\infty f(x) \, dx \approx \sum_{i=1}^n w_i e^{x_i} f(x_i) \]

An alternative method to frame displayed equations is with the \texttt{AMS-LaTeX boxed} command, page 270.
5.4. Additional elements

5.4.9 Chemical formulas and bold face in math formulas

In mathematics it is sometimes necessary to set individual characters or parts of the formula in bold face. This can be achieved simply with the math alphabet command \textbf{mathbf} that we met in Section 5.4.2:

\[
\mathbf{S^{-1}TS = dg(\omega_1,\ldots,\omega_n) = \Lambda}
\]

produces $S^{-1}TS = dg(\omega_1,\ldots,\omega_n) = \Lambda$.

In this example, the entire formula has been set as the argument of \textbf{mathbf} so that everything should be set in bold face. In fact, only numbers, lower and upper case Latin letters, and upper case Greek letters are set in \textbf{bold Roman} with \textbf{mathbf}. Lower case Greek letters and other math symbols appear in the normal \textbf{math} font.

If only part of the formula is to be set in bold face, that part must be given as the argument of the \textbf{mathbf} command.

\[
\mathbf{2\sqrt{x}/y = z}
\]

\[
\oint\limits_C V\,\mathrm{d}\tau = \oint\limits_{\Sigma} \nabla\times V\,d\sigma
\]

The \textbf{boldmath} command will set all characters in bold face, with the following exceptions:

• raised and lowered symbols (exponents and indices)
• the characters + : ; ! ? ( ) [ ]
• symbols that exist in two sizes (Section 5.3.7)

The \textbf{boldmath} declaration may not appear in math mode. It must be called before switching to math mode or within a parbox or minipage. The countercommand \textbf{unboldmath} resets the math fonts back to the normal ones.

\[
\textbf{boldmath}\{\oint\limits_C V\,\mathrm{d}\tau = \oint\limits_{\Sigma} \nabla\times V\,d\sigma\}
\]

If \textbf{boldmath} has been turned on outside the math mode, it may be temporarily turned off inside with \textbf{mbox{unboldmath$\ldots$}}.

\[
\textbf{boldmath}\{( P = \textbf{mbox{unboldmath$\ldots$}} b\})\textbf{unboldmath}
\]
yields: $P = mb$. Similarly, \textbf{boldmath} can be temporarily turned on within math mode with the structure \textbf{mbox{boldmath$\ldots$}}: $W_r = \int M\,d\varphi = r^2m\omega^2/2$

\[
\textbf{int}\textbf{mbox{boldmath$\ldots$}}  \textbf{mathrm{d}}\textbf{varphi} = \ldots
\]

An alternative method of printing single symbols in bold face is provided by the \textbf{bm} package described on page 394.

**Chemical formulas** are normally set in Roman type, not in italics as for mathematical formulas. This may be brought about by setting the formula as the argument of the font command \textbf{mathrm}:

\[
\textbf{mathrm{Fe_2^{2+}Cr_2O_4}} \quad \text{Fe}_2^{2+}\text{Cr}_2\text{O}_4
\]
5.4.10 International typesetting standards

The International Standards Organization (ISO) has established the recognized conventions for typesetting mathematics, the essential elements of which are presented in an article by Beccari (1997), along with a description of how they may be realized with \LaTeX. Some of these rules have already been mentioned and demonstrated in the examples. Here are the major points.

1. Simple variables are represented by italic letters, as $a\ b\ c\ x\ y\ z$.
2. Vectors are written in bold face italic, as $\mathbf{B}\ \mathbf{v}\ \mathbf{w}$.
3. Tensors of 2nd order and matrices may appear in a sans serif font, as $\mathsf{M}\ \mathsf{D}\ \mathsf{I}$.
4. The special numbers $e, i, \pi$, as well as the differential operator $d$, are to be written in an upright font to emphasize that they are not variables.
5. A measurement consisting of a number plus a dimension is an indivisible unit, with a smaller than normal space between them, as $5.3\ \text{km}$ and $62\ \text{kg}$. The dimension is in an upright font.

Point 1 is fulfilled automatically by \LaTeX. Point 5 is easily achieved by inserting the small space `\,` command between the number and dimension, as $5.3\,\text{km}$ and $62\,\text{kg}$. Using the protected space `\` instead is very common practice among \LaTeX users, which ensures that the two parts are not split, but with regular, not small spacing: $5.3\ \text{km}$ and $62\ \text{kg}$.

Point 2 is not satisfied with the `\vec` command, which produces $\vec{B}$. Nor does `\mathbf` help, for this yields $\mathbf{B}$, in an upright font. The best solution is to use the `\boldsymbol` command from the `amssymb` package (Section 12.2.1) or the `\bm` command from the tools package `bm` (Section B.5.4). Otherwise one must resort to defining

\begin{verbatim}
\renewcommand{\vec}[1]{\mbox{\boldmath$#1$}}
\end{verbatim}

for a revised `\vec` command.

Similarly point 3 can be met with the math alphabet command `\mathsf`. However, Beccari does point out that even tensor variables should probably be italic. This is more difficult, but it can be accomplished. See example 5 on page 191.

Point 4 is the one that is most often violated, especially for the differential $d$. We have demonstrated it in the examples in this book and have shown how it may be achieved with the `\mathrm` math alphabet command. However, to simplify the application, it is recommended to create some user-defined commands, such as

\begin{verbatim}
\newcommand{\me}{\mathrm{e}} \quad \text{for \textit{math} e}
\newcommand{\mi}{\mathrm{i}} \quad \text{for \textit{math} i}
\newcommand{\dif}{\mathrm{d}} \quad \text{for \textit{ differential operator} d}
\end{verbatim}

An upright $\pi$ is not so easy since this is not provided in the usual math fonts.

With these new commands, the equation on page 119 is more conveniently set with

\begin{verbatim}
\int_0^{\infty} f(x)\,\mathrm{d}x \approx \sum_{i=1}^{n_\text{w}} w_i \ e^{x_i} f(x_i)
\end{verbatim}
5.5 Fine-tuning mathematics

5.5.1 Horizontal spacing

Even though $\LaTeX$ has a thorough knowledge of the rules of mathematical typesetting, it cannot hope to understand the mathematical meaning. For example, $y\, dx$ normally means the joining of the variable $y$ with the differential operator $dx$, and this joining is designated by a small space between the two. However, $\LaTeX$ removes the blank in the entry $y\, dx$ and prints $ydx$, the product of three variables $y$, $d$, and $x$. At this point, $\LaTeX$ needs some fine-tuning assistance. (And the $d$ should really be upright, the $\mathrm{d}$ or $\mathrm{d}$ of the previous section.)

Small amounts of horizontal spacing can be added in math mode with the commands

\begin{itemize}
\item \textbackslash{}, small space \hspace{3/18 of a quad}
\item \textbackslash:, medium space \hspace{4/18 of a quad}
\item \textbackslash; large space \hspace{5/18 of a quad}
\item \textbackslash{}! negative space \hspace{−3/18 of a quad}
\end{itemize}

In the following examples, the third column contains the results without the additional horizontal spacing command.

\begin{itemize}
\item $\sqrt{\,2\,}\, x$ \quad $\sqrt{2} x$
\item $\sqrt{\,\log x\,}$ \quad $\sqrt{\log x}$
\item $0\left(1/\sqrt{n}\right)$ \quad $O\left(1/\sqrt{n}\right)$
\item $[0,1)$ \quad $[0,1)$
\item $\log n(\log \log n)^2$ \quad $\log n(\log \log n)^2$
\item $x^2/2$ \quad $x^2/2$
\item $n/\log n$ \quad $n/\log n$
\item $\Gamma_2 + \Delta^2$ \quad $\Gamma_2 + \Delta^2$
\item $R_i{}^j{}_{kl}$ \quad $R_i^j_{kl}$
\item $\int_0^x\!\int_0^y\! dF(u,\nu)$ \quad $\int_0^x\!\int_0^y\! dF(u,\nu)$
\item $\\\\\\\mathrm{d}F(u,\nu)$
\end{itemize}

Note: In the third last example $R_i{}^j{}_{kl}$ is so constructed that an invisible character of zero width comes after the index of $R_i$, and it is this dummy character that receives the following exponent. The result is $R_i^j_{kl}$, instead of $R_i{}^j{}_{kl}$ which is produced by $R_i{}^j{}_{kl}$.

There are no hard and fast rules for applying the math spacing commands. Some candidates are the differential operator, small root signs in text formulas followed by a variable, the dividing sign $/$, and multiple integral signs. The above examples illustrate many suitable situations.
5.5.2 Selecting font size in formulas

It is possible to alter the font sizes that \LaTeX{} selects for the various parts of the formula. First we must explain what sizes are available in math mode and what \LaTeX{}'s selection rules are.

In math mode there are four font sizes that may be chosen, their actual sizes being relative to the basic font size of the document class:

\begin{itemize}
  \item \texttt{\textbackslash displaystyle} $D$ Normal size for displayed formulas
  \item \texttt{\textbackslash textstyle} $T$ Normal size for text formulas
  \item \texttt{\textbackslash scriptstyle} $S$ Normal size for first sub-, superscript
  \item \texttt{\textbackslash scriptscriptstyle} $SS$ Normal size for later sub-, superscripts
\end{itemize}

From now on we shall make use of the symbolic abbreviations $D, T, S,$ and $SS$. When math mode is switched on, the active font size becomes $D$ for displayed and $T$ for text formulas. Their only difference lies in those symbols that appear in two sizes, plus the corresponding style of those superscripts and subscripts (Section 5.3.7). The larger symbols belong to $D$, the smaller to $T$.

Starting from these base sizes, various math elements will be set in other sizes. Once another size has been chosen for an element, it remains the active size within that element.

The table below shows the selection rules:

<table>
<thead>
<tr>
<th>Active size</th>
<th>Fractions upper</th>
<th>Fractions lower</th>
<th>Super-, subscripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D$</td>
<td>$T$</td>
<td>$T$</td>
<td>$S$</td>
</tr>
<tr>
<td>$T$</td>
<td>$S$</td>
<td>$S$</td>
<td>$S$</td>
</tr>
<tr>
<td>$S$</td>
<td>$SS$</td>
<td>$SS$</td>
<td>$SS$</td>
</tr>
<tr>
<td>$SS$</td>
<td>$SS$</td>
<td>$SS$</td>
<td>$SS$</td>
</tr>
</tbody>
</table>

If the active size is $D$, size $T$ is selected for both parts of the fraction. That is, for both numerator and denominator $T$ becomes the active size. If they in turn contain further fractions, these will be set in $S$. If the active size is $D$ or $T$, superscripts and subscripts (exponents and indices) appear in $S$; within them $S$ is the active size and any fractions or shiftings inside them will be set in $SS$.

The \LaTeX{} elements \texttt{\textbackslash atop} and \texttt{\textbackslash choose} are treated as fractions.

The active font size inside an array environment is $T$.

The smallest available math font size is $SS$. Once it has been reached, no further reduction is possible, so that all subsequent superscripts and subscripts appear in $SS$ as well.

From the table one easily sees that

\[
\left[ \quad a_0 \quad + \quad \frac{1}{a_1 \quad + \quad \frac{1}{a_2 \quad + \quad \frac{1}{a_3 \quad + \quad \frac{1}{a_4}}}} \right].
\]

must appear as shown at the right.

Such a math structure, called a \textit{continued fraction}, is normally printed as
By explicitly giving the font size within each element, one makes that size active rather than relying on the internally selected size. In this example, \( D \) is chosen for each denominator. Thus the next fraction acts as though it were outermost. The \texttt{\displaystyle} command may be omitted in the last fraction. (Can you see why?)

The selection table allows one to predict precisely which math font size will be applied at any part of the formula so that an explicit specification may be made if necessary. The effects of such a choice may also be calculated for the following math elements.

In the examples below, the right-hand column shows how the formulas would appear if the math font size is not explicitly specified.

\[
\frac{a}{x-y} + \frac{b}{x+y} \quad 1 + \frac{a-b}{a+b}
\]

\[
e^{-\frac{x_i-x_j}{n^i+n^j}}
\]

\[
\left(\begin{array}{cc}
\frac{a}{b} & \frac{e+f}{g-h} \\
\frac{c}{d} & \frac{i}{j} \\
0 & \frac{k}{l}
\end{array}\right)
\]

If an explicit size specification is to be given frequently within a document—say within every entry in an \texttt{array} environment—then considerable writing effort may be avoided by adding to the preamble

\[
\newcommand{\D}{\displaystyle}\newcommand{\T}{\textstyle}...
\]

In this way the size command may be given simply by typing \texttt{\D} or \texttt{T}, etc.

The math font size selection rules given above are in fact a simplification. For those readers who wish to know the exact details, we now present the complete description.
For each of the four math font sizes \( D \), \( T \), \( S \), \( SS \) there is a modified version \( D' \), \( T' \), \( S' \), \( SS' \). The difference is that with \( D \), \( T \), \( S \), \( SS \) the superscripts (exponents) are somewhat higher than they are with \( D' \), \( T' \), \( S' \), \( SS' \). Compare the positions of the exponent 2 above and below the line: \( \frac{x^2}{x^2} \). Otherwise the primed and unprimed font sizes are identical. The true selection rules are given in the table at the right.

When the font size is explicitly stated within a numerator or superscript, the unprimed version is selected; and in the denominators and subscripts, the primed fonts.

### 5.5.3 Manual sizing of bracket symbols

The \left...\right commands preceding one of the 22 bracket symbols listed in Section 5.4.1 adjust the symbol size automatically according to the height of the enclosed formula text. However, it is possible to select a size explicitly by placing one of the \TeX commands \texttt{\textbackslash big}, \texttt{\textbackslash Big}, \texttt{\textbackslash bigg}, or \texttt{\textbackslash Bigg} before the bracket symbol.

```
none ( ) [ ] { } \{ \} \langle \rangle / \backslash \| \uparrow \downarrow \uparrow \downarrow
\texttt{\textbackslash big} ( ) [ ] { } \{ \} \langle \rangle / \backslash \| \uparrow \downarrow \uparrow \downarrow
\texttt{\textbackslash Big} ( ) [ ] { } \{ \} \langle \rangle / \backslash \| \uparrow \downarrow \uparrow \downarrow \uparrow \downarrow
\texttt{\textbackslash bigg} ( ) [ ] { } \{ \} \langle \rangle / \backslash \| \uparrow \downarrow \uparrow \downarrow \uparrow \downarrow \uparrow
\texttt{\textbackslash Bigg} ( ) [ ] { } \{ \} \langle \rangle / \backslash \| \uparrow \downarrow \uparrow \downarrow \uparrow \downarrow \uparrow \downarrow \uparrow
```

In contrast to the \left...\right pairs, the explicit bracket size commands do not have to be contained within one line of a multiline formula. The opening and closing brackets may appear on different lines. This applies also to the commands described in the next paragraph.

### 5.5.4 Math style parameters

The mathematical style parameters listed below are set to standard values by \LaTeX. The user may alter them at any time with the \setlength command in the usual manner.
5.5. Fine-tuning mathematics

\arraycolsep
Half the width of the intercolumn spacing for the array environment (see also Section 4.8.2).

\jot
Extra vertical spacing that appears between the rows of multiline equations in the eqnarray and eqnarray* environments.

\mathindent
The amount of indentation for math formulas when the document class option fleqn has been selected.

\abovedisplayskip
The vertical spacing above displayed formulas when the left side of the formula is closer to the left margin than the end of the preceding line of text. Such a formula is designated long.

\belowdisplayskip
The vertical spacing inserted below a long displayed formula.

\abovedisplayshortskip
Vertical spacing added above a short displayed formula. This is one in which the left edge is to the right of the end of the preceding line of text.

\belowdisplayshortskip
Vertical spacing inserted after a short displayed formula.

\topsep
The above four spacings are not used with the document class option fleqn where instead \topsep is inserted above and below displayed formulas (see Section 4.4.2).

All the above parameters, except \jot, should be rubber lengths (Section 2.4.2).

5.5.5 Some further advice

Sometimes authors desire horizontal and vertical alignments of their formulas that just cannot be achieved using the means described so far. They should consider placing their formulas inside horizontal or vertical boxes which may then be positioned wherever they please.

Similarly the array environment together with explicit size declarations and the table construction and style elements from Sections 4.8.1 and 4.8.2 should be able to accomplish just about any horizontal and vertical alignment.

Exercise 5.19: Generate this continued fraction.
Note: in contrast to the example on page 146, the 1 here in the numerator appears left justified.
Hint: do you remember the command \hfill?

\[
a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4}}}}
\]

Exercise 5.20: Produce the following set of equations with the array environment.

\[
\begin{align*}
\sin 2\alpha &= 2 \sin \alpha \cos \alpha, & \cos 2\alpha &= \cos^2 \alpha - \sin^2 \alpha \\
\sin 3\alpha &= 3 \sin \alpha - 4 \sin^3 \alpha, & \cos 3\alpha &= 3 \cos^3 \alpha - 3 \cos \alpha \\
\sin 4\alpha &= 8 \cos^3 \alpha \sin \alpha - 4 \cos \alpha \sin \alpha, & \cos 4\alpha &= 8 \cos^4 \alpha - 8 \cos^2 \alpha + 1
\end{align*}
\]
Chapter 5. Mathematical Formulas

Hint: recall that @{...} expressions in the formatting field of the array environment can be used to insert horizontal spacing and/or mathematical text between the columns (see the first example in Section 5.4.3).

Exercise 5.21: Create the following output with the array environment.

**Equations for the tangential plane and surface normal**

<table>
<thead>
<tr>
<th>Equation for the surface</th>
<th>Tangential plane</th>
<th>Surface normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F(x, y, z) = 0$</td>
<td>$\frac{\partial F}{\partial x}(X - x) + \frac{\partial F}{\partial y}(Y - y) + \frac{\partial F}{\partial z}(Z - z) = 0$</td>
<td>$X - x = Y - y = Z - z$</td>
</tr>
<tr>
<td>$z = f(x, y)$</td>
<td>$Z - z = p(X - x) + q(Y - y)$</td>
<td>$p = \frac{X - x}{q} = \frac{Z - z}{-1}$</td>
</tr>
<tr>
<td>$x = x(u, v)$</td>
<td>$\begin{array}{lll}X - x &amp; Y - y &amp; Z - z \ \frac{\partial x}{\partial u} &amp; \frac{\partial y}{\partial u} &amp; \frac{\partial z}{\partial u} \end{array} = \begin{array}{lll}Y - y &amp; Z - z \end{array}$</td>
<td>$\begin{array}{lll}\frac{\partial y}{\partial u} &amp; \frac{\partial z}{\partial u} &amp; \frac{\partial z}{\partial v} \end{array}$</td>
</tr>
<tr>
<td>$y = y(u, v)$</td>
<td>$\begin{array}{lll}X - x &amp; Y - y &amp; Z - z \ \frac{\partial x}{\partial v} &amp; \frac{\partial y}{\partial v} &amp; \frac{\partial z}{\partial v} \end{array} = \begin{array}{lll}Y - y &amp; Z - z \end{array}$</td>
<td>$\begin{array}{lll}\frac{\partial y}{\partial v} &amp; \frac{\partial z}{\partial v} &amp; \frac{\partial z}{\partial v} \end{array}$</td>
</tr>
<tr>
<td>$z = z(u, v)$</td>
<td>$\begin{array}{lll}X - x &amp; Y - y &amp; Z - z \ \frac{\partial x}{\partial v} &amp; \frac{\partial y}{\partial v} &amp; \frac{\partial z}{\partial v} \end{array} = \begin{array}{lll}Y - y &amp; Z - z \end{array}$</td>
<td>$\begin{array}{lll}\frac{\partial y}{\partial v} &amp; \frac{\partial z}{\partial v} &amp; \frac{\partial z}{\partial v} \end{array}$</td>
</tr>
<tr>
<td>$r = r(u, v)$</td>
<td>$(R - r)(r_1 \times r_2) = 0$ or $(R - r)N = 0$</td>
<td>$R = r + \lambda(r_1 \times r_2)$ or $R = r + \lambda N$</td>
</tr>
</tbody>
</table>

In this Table, $x, y, z$ and $r$ are the coordinates and the radius vector of a fixed point $M$ on the curve; $X, Y, Z$ and $R$ are the coordinates and radius vector of a point on the tangential plane or surface normal with reference to $M$; furthermore $p = \frac{\partial z}{\partial x}$, $q = \frac{\partial z}{\partial y}$ and $r_1 = \partial r/\partial u$, $r_2 = \partial r/\partial v$.

**Note:** If you succeed in reproducing this mathematical table you should have no more problems with positioning formulas and their parts!

Hints for the solution:

1. Define abbreviations such as \(\texttt{D}\) for \texttt{\hspace{1em}displaystyle} and \texttt{bm for \hspace{1em}boldmath} and possibly even \texttt{ba and \hspace{1em}ea for \hspace{1em}begin{array} and \hspace{1em}end{array}}.

2. Build the table in stages. Start with the head and only proceed further when it looks reasonable. Watch out that normal text within the array environment must be included inside an \texttt{mbox}.

3. Continue with the first mathematical row. Here the first entry in the second column is also an array environment that is aligned with the rest of the row by means of the \texttt{[\hspace{1em}]} positioning argument. Do not forget to activate the size \(\texttt{D}\) at the necessary places in the inner structures. Insert a possible strut (Section 4.7.6) to generate the right distance from the head. The distance to the next row can be adjusted with a length specification added to the row terminating command \texttt{\hspace{1em}[]}.
4. After this row has been completed successfully, the next should pose no difficulties.

5. In the third mathematical row, both the first column and the left side of the second column consist of array environments. The third column contains three fractions whose denominators may be produced either with array or with the TeX \{...\atop...\} command; the fractions themselves are placed in two rows with another array environment.

6. The second and third columns of the last mathematical row again contain array environments. Some of the sub-formulas in this row appear in \textbf{boldmath}. Recall that this command may only be invoked within text mode, that is, within an \texttt{mbox}.

7. In the last row, all three columns of the outer array environment are merged together and the text is set within a parbox of appropriate width:

\begin{verbatim}
\begin{multicolumn}{|c|}{\parbox{..}{......}}
\end{verbatim}

\section{Beyond standard \LaTeX}

The typesetting elements presented in this chapter should be able to handle many of the everyday problems of mathematical composition. The American Mathematical Society, \texttt{AMS}, which has supported TeX from its beginning, has developed a set of packages to assist authors in composing even more complicated formulas with \TeX. Known as \texttt{AMS-LaTeX}, it includes many extra commands, further environments to supplement eqnarray, and many additional symbols. This system is presented in Chapter 12.

In any event, typesetting mathematics is a very complicated issue. It was for this reason that Donald Knuth selected the $\$ sign as the switch to math mode: in the old days of movable lead type, setting mathematical formulas was an expensive business because of the extra workload it entailed. Even today, electronic typesetting is far more involved for math than for simple text.
The two topics of this chapter have one thing in common: neither is handled by \TeX at all, and thus neither by \LaTeX. Outputting graphics files produced by other programs, or printing in color, are functions of the output device, that is, the printer, the monitor, the electronic output file, and these are controlled by the DVI driver program that converts the DVI output of \LaTeX into the final output form. For each output type, a different set of commands is needed to switch color or to send an image.

What \TeX does have is a way of sending special driver-specific commands to the driver by placing them in the .dvi file; the command to do this is even named \texttt{\special}. \LaTeX exploits this feature to pass driver commands to the .dvi file, but to do so, it needs to know which driver is going to be used for the further processing.

With the advent of \LaTeX\$\varepsilon$, a standardized syntax was introduced, replacing the driver-specific commands that abounded with \LaTeX 2.09 and which caused much confusion. At last the importation of external images had become a basic part of \LaTeX usage, and was no longer an exotic, magical art. Well . . . , nearly.

It is still necessary to tell \LaTeX which driver will be used afterwards, as an option to the graphics and color packages, but once that is done, all the driver-specific special commands are loaded from a corresponding .def files, which the user does not have to worry about. The top-level syntax remains the same for all types.

It is also important to realize that not all image formats can be handled by every driver. PostScript drivers for example will only accept PostScript images, and then only as \textit{encapsulated} PostScript. Image format conversion may be required.

6.1 The \texttt{graphics} packages

Strictly speaking, these packages, referred to as the \textit{graphics collection}, are extensions to the basic \LaTeX installation. We describe them here
because of their indispensability in producing professional documents and camera-ready copy. Provided by Sebastian Rahtz and David Carlisle and many other contributors, they can be expected to be part of every \LaTeX\ installation. A basic manual is provided in a file named \texttt{grfguide.tex}, which may already be preprocessed to \texttt{.dvi} or some other form. Even more details on these packages and on advanced applications of PostScript to \LaTeX\ are to be found in the book \textit{The \LaTeX\ Graphics Companion} (Goossens \textit{et al.}, 1997).

The commands defined by these packages are the building blocks for other packages that either emulate the older driver-specific ones or provide a more comfortable syntax for these functions. As long as these other packages are based on \texttt{graphics} and \texttt{color}, they should be equally compatible with all the supported drivers.

These features are not limited to PostScript drivers. As long as a driver can support the inclusion and manipulation (scaling, rotation) of graphics, and/or the use of color, a \texttt{.def} file can be written to enable it to make use of the standardized graphics and color commands.

Driver names that may be used as options are

\begin{verbatim}
dvipdf  dviwin ▼ □ ⇑ pctexhp ▼ □ ⇑ tcidvi ▼
dvipdfm emtex ▼ □ ⇑ pctexps textures
dvips oztex pctexwin ▼ □ ⇑ truetex ▼
dvipsone pctex32 pdftex vtex
\end{verbatim}

(Limited functionality: ▼ no color support; □ no scaling; ⇑ no rotation.)

The \texttt{xdvi} previewer (and its Windows equivalent \texttt{windvi}) work with PostScript graphics; although there is also an \texttt{xdvi} option, it is only an alias for \texttt{dvips}.

A driver option \textit{must} be specified when loading the \texttt{graphics} and \texttt{color} packages, for example, as

\begin{verbatim}
\usepackage[dvips]{graphics,color}
\end{verbatim}

However, it is possible to establish a default for the local configuration, as described in Section 6.1.7. It is conceivable that your installation has already done this, so it is worth doing an experiment to see if a default driver option already exists.

6.1.1 Importing external graphics

We wish to have a graphics file produced by some other program included in the document, possibly scaled to a desired size or rotated by 90°. One wants to do by computer what used to be done with scissors and glue.

There are two packages available for importing and manipulating external graphics files: the basic \texttt{graphics} package and the more extended \texttt{graphicx} one. They both offer identical functionality, differing only in their syntax.
In addition to the driver names, there are some other options that may be invoked when the packages \texttt{graphics} or \texttt{graphicx} are loaded:

- \texttt{draft} does not import but places a framed box where the graphic would appear, with the file name printed inside; this speeds up the processing considerably when one is only working on the text;

- \texttt{final} counteracts \texttt{draft}; needed when the \texttt{draft} option has been issued globally in \texttt{\documentclass};

- \texttt{hidescale} leaves blank space where scaled text should be;

- \texttt{hiderotate} leaves blank space where rotated text should be; this and the option \texttt{hidescale} are useful if the previewer cannot handle scaling or rotation;

- \texttt{hiresbb} look for bounding box values in \texttt{\%HiResBoundingBox} instead of the normal \texttt{\%BoundingBox} line.

### 6.1.2 Importing with the \texttt{graphics} package

The basic importing command with the \texttt{graphics} package is

\begin{verbatim}
\includegraphics[llx,lly][urx,ury]{file_name}
\end{verbatim}

where \texttt{llx}, \texttt{lly} are the coordinates of the lower left corner, and \texttt{urx}, \texttt{ury} those of the upper right corner of the bounding box containing the part of the picture that is to be included. In other words, they say where the scissors are to be applied. Units may be specified (like \texttt{[3cm, 2in]}) but if they are omitted, big points (bp, 72 per inch, 28.3464... per cm) are assumed. If only one optional argument is given, it is the upper right corner, and the lower left is assumed to be \texttt{[0,0]}.

If no bounding box coordinates are given, the driver will obtain them some other way, depending on the type of graphics file. For example, for the very common \texttt{encapsulated PostScript} files with extension \texttt{.eps}, the bounding box information is extracted from the graphics file itself. The figure at the right is stored in such a file and is included simply with the command

\begin{verbatim}
\includegraphics{clock}
\end{verbatim}

(It is not necessary, nor recommended, to include the file extension \texttt{.eps} in the file name designation; \LaTeX{} will automatically test for all the extensions allowed by the selected driver option, something that permits greater flexibility in the source text.)
With \includegraphics*, the figure is clipped, so that any drawing outside the specified bounding box is suppressed. This is useful if only part of a figure is to be reproduced. It is also vital with some perverse figures that paint the whole page white!

**Scaling**

The graphics file in the above commands is transferred to the document in its original size. In order to rescale it, there are two commands available:

\scalebox{\textit{h\_scale}}{\v\_scale}{\textit{text}}

which applies horizontal and vertical scale factors to the contents \textit{text}; if \textit{v\_scale} is omitted, it is the same as \textit{h\_scale};

\resizebox{\textit{h\_length}}{\v\_length}{\textit{text}}

adjusts the figure to fit into the specified horizontal and vertical sizes; if either length is given as !, the one scale factor is used for both dimensions. A *-form allows \textit{v\_length} to refer to the height + depth of the box, rather than just to the height. In both cases, the contents \textit{text} may be an \includegraphics command, but it may also be any arbitrary text.

**Reflection**

The contents of a box may be reflected horizontally with

\reflectbox{\textit{text}}

**Rotation**

Rotation of a box about the left-hand end of its baseline is done with

\rotatebox{\textit{angle}}{\textit{text}}

where \textit{angle} is in degrees, and the rotation is counterclockwise.

To illustrate this, we have scaled the previous clock figure to a height of 2 cm and then rotated it by 30°, using the commands

\rotatebox{30}{\resizebox{!}{2cm}{\includegraphics{clock}}}

For demonstration purposes, we have added framed boxes around the figure before and after the rotation. It is the inner (tilted) box that is 2 cm high, while the overall outer box is somewhat higher and broader due to its inclined contents. Without these frames the figure seems to have extraneous space above and to the left, something that can be very
puzzling even to experienced users. (The frames would not normally appear with the above commands.)

**Exercise 6.1:** Copy the lines at the right to a file named demo.eps and then include it in a *LATEX* document, with some normal text above and below it. The result should appear as:

```latex
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 169 158 233 242
220 200 moveto
200 200 20 0 360 arc
170 170 moveto
230 220 lineto
170 210 lineto
225 160 lineto
205 240 lineto
170 170 lineto
stroke
showpage
```

**Note:** This file can be copied from the enclosed CD in the directory `\books\Kopka_and_Daly\`, where it is also available in PDF format.

### 6.1.3 Importing, scaling, rotating with the graphicx package

**Package:** graphicx

If one selects the graphicx rather than the graphics package, a different interface is available for both importing and rotation, one making use of keys and values:

```
\includegraphics[key=value,...]{file.name}
```

The keys are of two types: those that take a numerical value, and those that are flags with the values `true` or `false`. Simply giving the name of a flag without a value is equivalent to setting it to `true`. Possible keys and their values are:

- `scale= number`; enters the number by which the figure size should be magnified over its natural size;
- `width= length`; specifies the width to which the figure should be scaled; if `height` not given, it is scaled with the same factor as the width;
- `height= length`; specifies the height to which the figure should be scaled; if `width` is not given, it is scaled with the same factor as the height;
- `totalheight= length`; like `height` but specifies the height plus depth of the figure; should always be used in place of `height` if the figure has been rotated;
- `keepaspectratio (=true|false)`; if both `height` and `width` are specified, this flag ensures that the original height/width ratio remains unchanged; the figure will not exceed either of the given dimensions;
angle = number; the angle by which the figure is to be rotated counterclockwise, in degrees; any height or width specifications coming before this key are also rotated, so that the height becomes the width, while the width becomes either the height (positive angle) or depth (negative angle);

origin = loc; determines the point about which the rotation occurs; default is bl for bottom left corner; also possible are c for center, t for top, r for right, and B for baseline; any sensible combination, such as tr, is allowed;

draft (=true|false); like the draft package option but applied to the one graphics file; the figure is not imported, but rather a framed box of the correct size is printed containing the name of the file;

clip (=true|false); suppresses the printing of any graphic outside the bounding box;

bb = llx lly urx ury; enters the coordinates of the bounding box manually, if they are missing or incorrect in the graphics file, or to be deliberately altered; the specifications are four lengths separated by blanks; units may be given, but if omitted, big points (bp) are assumed; this option should never be used with pdfTEX;

viewport = llx lly urx ury; specifies the bounding box but relative to the lower left corner of the existing one; this is the preferred method for correcting the bounding box, or (with clip) to select only a portion of the whole figure; with pdfTEX, this option must be used rather than bb;

trim = dllx dlly durx dury; reduces the existing bounding box by the amounts specified;

hiresbb (=true|false); like the hiresbb package option but applied to the one graphics file; reads bounding box information from the $$\text{HiResBoundingBox}$$ line in the graphics file.

The keys are all optional; they are included as needed. Their order is not important other than that angle interchanges any previous height and width meanings. The sets of key/values are separated from each other by commas. It is often best to set width relative to the current line width \linewidth, say as width=0.8\linewidth.

With the key/value syntax, the tilted, scaled graphic on page 156 is produced with

\includegraphics[height=2cm,angle=30]{clock}
For compatibility with the `graphics` package, there also exists an `\includegraphics*` version that clips the imported figure; this is equivalent to including the key `clip`.

With the `graphicx` package, the `\rotatebox` command is similarly redefined to accept the optional key `origin`.

**Exercise 6.2:** Include the graphics file `demo.eps` from Exercise 6.1 scaled by a factor of 2 and rotated by $45^\circ$ using the `\includegraphics` command with the `graphicx` package. Experiment with various keys, in particular with `height` and `width` together.

### 6.1.4 Additional graphics packages

#### Package: epsfig

Sebastian Rahtz has provided a package `epsfig` that not only updates the earlier (EPiX 2.09) version, but also re-implements Rokicki’s `epsf` package by means of the `graphicx` commands. This is helpful for users who are accustomed to those syntaxes. For `epsf`, this is

\[
\text{\epsfysize=y\_size \ or \ \epsfxsize=x\_size}
\]
\[
\text{\epsf[llx lly urx ury]{file\_name}}
\]

The `epsfig` package also defines an importing command that makes use of the regular keys and values to enter its parameters:

\[
\text{\epsfig{file=file\_name, key=value, ...}}
\]

For compatibility with some older versions, there is a `\psfig` command which is synonymous with the above.

The `epsfig` package is included in the bundle of `graphics` packages and drivers.

#### Package: lanscape

Another extra package in the `graphics` bundle is `lanscape`, by David Carlisle. This defines a `landscape` environment that prints its contents rotated $90^\circ$ on a page for itself. Head and footlines remain as normal. This is intended primarily for inserting figures that are in landscape mode, that is, wider than they are high.

The `rotating` package by Sebastian Rahtz and Leonor Barroca tries to make the interface for rotation somewhat simpler. It defines

\[
\begin{sideways}
text \end{sideways}
\begin{turn}{angle} text \end{turn}
\begin{rotate}{angle} text \end{rotate}
\text{\turnbox{angle}{text}}
\]

where `sideways` rotates `text` by $90^\circ$, `turn` by an arbitrary angle. The environment `rotate` and command `turnbox` are equivalent: they rotate but in a box of zero size, so that the contents overlap the surroundings.

This package is not part of the `graphics` bundle and must be obtained separately.
6.1.5 Superimposing images

Sometimes it is desired to superimpose one image file over part of another, say to add an inset in a corner. Or one may want to add text elements, or even lines and arrows pointing to some feature in the image. One might even want to make some area of the image into a link to another document. The picture environment described in Section 13.1 can be used to allow exact placement of such objects.

For example, in the figure on this page, we embellish the TeXLive welcome image from Figure B.1 on page 383 by adding a blow-up of a small section, pointing a labeled arrow to it, and making the area around the text ‘TeXLive!’ into a link. We do this with the following input:

\begin{picture}(80,80)
\thicklines
\put(0,0){\includegraphics[width=.8\linewidth]{texlive}}
\put(57,26){\framebox(14.2,14.6)}
\drawline(57,26)(-10,36.7)
\end{picture}

Figure 6.1: An embellished image file

\begin{itemize}
\setlength{\unitlength}{0.01\textwidth}
\setlength{\fboxsep}{0pt}
\setlength{\fboxrule}{1.5pt}
\begin{picture}(80,80)
\thicklines
\put(0,0){\includegraphics[width=.8\textwidth]{texlive}}
\put(57,26){\framebox(14.2,14.6)}
\drawline(57,26)(-10,36.7)
\end{picture}
\end{itemize}
6.1. The graphics packages

[diagram]

Here the unit length for the picture placement has been set to 1/100 of the current line width \linewidth, so we can conveniently use 100 units for the full width. The picture environment reserves a space of 80×80 units; the image file texlive is placed at the lower left corner (location 0,0) and set to a width of 80% of \linewidth. We then draw a framed box around a portion of this image, and then add that portion as a second image (selected with viewport and clipped) as a blow-up. The \drawline commands (from the epic package) join the corners of the original and blown-up areas. The order here is important since later images overwrite the earlier ones. Then we add an arrow (\vector) and the text 'Blow-up' (with \makebox). Finally, the \hyperlink command from the hyperref package turns the 30×7 unit area around the text 'TEXLive!' into a active link to the target named info.

One can also place objects outside of the reserved picture area, so arrows could be added pointing from the main text to some part of an image. This is more appropriate for the production of slides (Chapter 15) where one wants rigid control over the relative locations of graphics and text.

6.1.6 How importation can go wrong

In order to understand better what can go wrong when a graphics file is imported, and to know what to do about it, it is important to realize how the interplay between \LaTeX and the driver program functions.

\LaTeX has no idea what is in the graphics file; for it, the figure is simply a box of a given height, width, and depth, as indeed are all the characters that \LaTeX processes. The information on the graphic's natural size is somehow obtained, either by information in the graphics file itself, or through the optional entries in the \includegraphics command or equivalent. After scaling and rotating, \LaTeX knows the final size that it must reserve in the output text for the figure.

What is then written to the .dvi file is the name of the graphics file and information on how it should be transformed. Just how this information is coded depends on the graphics driver selected. When the printer driver program processes the .dvi file, it interprets these special instructions, reads in the specified graphics file, performs the transformations, and places the result where \LaTeX has said it should go. The end result is that the area inside the designated bounding box coincides with the box that \LaTeX has reserved for it. If the bounding box information is incorrect, the figure is obviously going to be misplaced.
Chapter 6. Graphics Inclusion and Color

PostScript output

Graphics inclusion was originally developed for PostScript graphics adhering to the *encapsulated* specification that was designed to allow PostScript code to be included within another PostScript file. Tomas Rokicki and his dvips driver broke the ground in this field and thus established the standards. It is for that reason that we place so much emphasis on PostScript graphics here. Encapsulated PostScript files may not contain any commands that reset the whole graphics page, and furthermore, they must contain a comment line of the form

```plaintext
%%BoundingBox: llx lly urx ury
```

giving the coordinates of the lower left and upper right corners of the bounding box. The units are never specified, always being big points.

PDF output

Today PDF is an important alternative to PostScript, and the pdfTeX program (Section 10.2.3) is the most convenient way to produce it with \LaTeX. However, pdfTeX is not only a replacement for the classical \TeX program, producing identical output, it also combines the functionality of a DVI-to-PDF driver. Thus no intermediate .dvi file is generated at all, with the .pdf file being generated directly from the \LaTeX source file in one operation. This has consequences for graphics inclusion with pdfTeX. First of all, the number of allowed graphics formats is increased: images may be PNG (.png files), JPEG (.jpg or .jpeg files), TIFF (.tif or .tiff files) or even a single image in a one-page PDF file. Ironically, PostScript images are absolutely out! Secondly, the concept of bounding box does not really apply to these formats, since they have a natural size anyway. The viewport option should therefore be used to make a cutout from the original rather than bb.

An alternative method of obtaining PDF output is to employ the true DVI-to-PDF driver, dvipdfm, written by Mark A. Wicks. In this case, a .dvi file is produced as usual (with the graphics option dvipdfm), and then the PDF output is generated with this conversion program. The same set of graphics formats is allowed as for pdfTeX, for much the same reasons.

The third method would be to generate DVI, then PostScript, and then convert it to PDF. As far as the graphics are concerned, this is the same as PostScript.

Having pointed out how the importation takes place, we can now discuss what can go wrong along this chain of processes.

Problems with importation: pdfTeX

The single most common error when importing a graphics file with pdfTeX is that it does not exist in one of the permitted formats. Many users who are used to using classical \TeX with PostScript simply switch to pdfTeX and are astonished to get the error message:

```plaintext
! \LaTeX Error: Unknown graphics extension: .eps.
```

As explained above, pdfTeX cannot handle PostScript files, so the graphics must be converted to, or regenerated in, another format.
The above error message will appear only if one has specified the full name of the graphics file including the extension .eps. If the extension is absent, which we recommend, the error message will report that the file cannot be found. This means that it cannot be found with any of the allowed extensions. It might very well be that it exists with the extension .eps, but since that is not allowed for pdf\TeX, it will not be sought.

To maintain flexibility, one can have the same graphics file available both as an .eps and as a .pdf file. Leaving the extension off in the $\texttt{\includegraphics}$ command means one does not need to make any changes to the file name when switching between regular \TeX and pdf\TeX.

**Problems with importation: PostScript**

The most common problems encountered when importing encapsulated PostScript files are listed here.

**No bounding box.** If the bounding box information is totally missing from the graphics file, \LaTeX issues the error message

```
! \LaTeX Error: Cannot determine size of graphic
  in ... (no BoundingBox).
```

The solution is to determine the bounding box coordinates somehow (see next point) and to include them, either in the $\texttt{\includegraphics}$ command, or by editing the graphics file itself. However, if there really is no bounding box information in the file, it is unlikely to conform to the encapsulated standard and will cause other problems.

**The placement is incorrect.** Both \LaTeX and the driver process without error messages, but the figure is either displaced from the expected position, or is far too small.

Most likely the bounding box information is incorrect. Many applications that produce PostScript files are too lazy to calculate the true bounding box, or they think they are generating a whole page with a figure somewhere in the middle. In either case, the bounding box corresponds to the full page even though the printed figure occupies only a portion of it.

Find the true bounding box by one of the following methods:

1. Print the figure, mark the lower left and upper right corners of the box containing the figure, and measure their distances from the left and bottom edges. Enter these distances in the $\texttt{\includegraphics}$ command, or edit the PostScript file. In the latter case, convert to big points.

Difficulties with this are that some encapsulated PostScript files cannot be printed on their own, and that the left and bottom
edges of the paper need not be the exact lines from which the printer really measures.

2. Include the figure in a short \LaTeX file such as

\begin{verbatim}
\setlength{\fboxsep}{-\fboxrule}
\fbox{\includegraphics{test.eps}}
\end{verbatim}

and print the output. The apparent bounding box appears as a framed box. Measure the true bounding box relative to the left and bottom edges of the apparent one, and enter the values in the \includegraphics command with the viewport= key. If necessary, scale the figure down to fit on the page, but then remember to increase the measurements by the same scale factor.

3. Use the GhostView program to fix up the bounding box, either automatically or manually. This is the most convenient method if you have this utility for viewing and manipulating PostScript files.

**Immovable graphic.** It does not shift, nor scale nor rotate, no matter what is specified. In this case, it violates the encapsulated PostScript rules and contains some global plotting commands. Graphics produced by word processing programs are notorious for this. Often the offending command is \texttt{setpagedevice}.

There is little that can be done to correct this, other than trying to regenerate the graphics file with an option for encapsulated PostScript. Judicious editing can remove the troublesome lines, but this could result in the file becoming totally unreadable.

### 6.1.7 Configuring graphics importation

Although the graphics syntax has been standardized, and most of the driver-specific coding hidden in the .def files, there are still a number of items that must be set up for any particular installation and operating system. These are most conveniently placed in the local configuration file \texttt{graphics.cfg} which is read in if it is present.

Many installations come with a ready-made configuration file, so there may not be any need to make up your own. On the other hand, you may want to edit or replace it for your own purposes.

**Default driver**

The choice of driver option must always be given, but a local default can be specified in the configuration file. For example, if \texttt{dvips} is the standard graphics driver, you may add
6.1. The graphics packages

\ExecuteOptions{dvips}
to graphics.cfg and you do not need to give the option dvips when loading
graphics or graphicx. You may still specify any other graphics driver option if
you wish.

Many supplied configuration files go further and determine whether pdf\LaTeX
is activated or not; if so, the default graphics option is set to pdftex, otherwise
to dvips.

The rest of the configuring commands in this section can be issued either in
the graphics.cfg file, or in the document.

Search path for graphics files

One can specify the directories where \LaTeX is to look for graphics files with

\graphicspath{\textit{dir_list}}

where \textit{dir_list} is a list of directory names, each enclosed in brackets \{\}, with no
other separator. The syntax of the local operating system must be used. Without
this command, \LaTeX searches for graphics files in the same directories as for all
other \TeX files. Example,

\begin{verbatim}
\graphicspath{{figs/}{eps/}} for Unix, DOS, Windows
\graphicspath{{:/figs:}{:/eps:}} for Macintosh
\graphicspath{{:figs:}:eps:}} for VMS
\end{verbatim}

Note: The Unix syntax works even with DOS and Windows, since the normal
backslash in the directory names must be replaced by a slash / to avoid looking
like a command name.

Default extensions

A list of default extensions for the graphics files can be defined with

\DeclareGraphicsExtensions{\textit{ext_list}}

This means that only the root name of the file must be given and \LaTeX will
attempt to find it by attaching all the possible extensions. For PostScript drivers,
the \textit{ext_list} is usually set to .eps,.ps. At our installation, we also include the
non-standard extension .psc. Note that the above command does not add to the
list of extensions but rewrites it anew; if you wish to add to the list, you must
include all the allowed extensions in the one declaration.

Graphics types

Defining the extensions is only part of the task: one must also associate each
extension with a graphics type so that \LaTeX knows how to process it. PostScript
recognizes only one type, eps, encapsulated PostScript, but there do exist other
types such as bmp and pcx for other drivers. For the non-standard .psc extension
above, we must also give

\DeclareGraphicsRule{.psc}{eps}{}{}

to inform \LaTeX that this extension belongs to type eps. The other two (empty)
arguments specify that the bounding box information is to be read from the file
itself, and that no other program needs to be applied to the file.
Compressing graphics files

Since PostScript files are often extremely large, it makes sense to try to compress them with either the zip or gzip programs. In such a case, the .eps file is replaced by a file with extension .zip, or .eps.gz, or .eps-gz. Two problems now arise: first, \LaTeX cannot read such files to obtain the bounding box information, and secondly, the driver needs to unpack such a file to include it in the final output. This can be accomplished with, for example,

\texttt{\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{'gunzip -c #1}}

which establishes the graphics type as \texttt{eps}, with the bounding box information in the file of the same name and extension .eps.bb, and that the operating system command \texttt{gunzip -c} must be applied to the file (represented as \texttt{#1}). The single quote ' is required to indicate a system command. The \texttt{%%BoundingBox} line of the original file must be copied and stored in the .eps.bb file.

Such decompression rules are system dependent and thus need to be configured for the local installation. For example, under the VMS operating system, the gzip program produces files with extension .eps-gz and decompression is performed with gzip -d rather than with gunzip. The corresponding rule becomes

\texttt{\DeclareGraphicsRule{.eps-gz}{eps}{.bb}{'gzip -d -c #1}}

6.2 Adding color

The \texttt{color} package accepts the same driver options listed on page 154 for the \texttt{graphics} package. In addition, it also recognizes the options:

\texttt{monochrome}

to convert all color commands to black and white, for previewers that cannot handle color;

\texttt{dvipsnames}

makes the named color model of dvips (Section 6.2.1) available to other drivers;

\texttt{nondvipsnames}

disables the named model for dvips, to save memory;

\texttt{usenames}

loads all the named colors as defined ones; again, see Section 6.2.1 for details.

A local configuration file \texttt{color.cfg} can be set up in the same way as for the \texttt{graphics} package. The default driver option is specified in exactly the same way as in Section 6.1.7.

Colors are specified either by a defined name, or by the form

\texttt{[model]{specs}}
6.2. Adding color

where *model* is one of rgb (red, green, blue), cmyk (cyan, magenta, yellow, black), gray, or named. The *specs* is a list of numbers from 0 to 1 giving the strengths of the components in the model. Thus [rgb]{1,0,0} defines red, [cmyk]{0,0,1,0} yellow. The gray model takes only one number. The named model accesses colors by internal names that were originally built into the dvips driver, but which may now be used by some other drivers too. This model is described in Section 6.2.1.

A color can be defined with

```
definecolor{name}{model}{specs}
```

and then the *name* may be used in all the following color commands. Certain colors are automatically predefined for all drivers: red, green, blue, yellow, cyan, magenta, black, white.

In the following color commands, *col_spec* is either the name of a defined color, like {blue}, or [model]{spec}, like [rgb]{0,1,0}.

```
\pagecolor{col_spec}
\color{col_spec}
\textcolor{col_spec}{text}
\colorbox{col_spec}{text}
\fcolorbox{col_spec1}{col_spec2}{text}
\normalcolor
```

*\pagecolor{col_spec}* sets the background color for the current and following pages;

*\color{col_spec}* is a declaration to switch to setting text in the given color;

*\textcolor{col_spec}{text}* sets the text of its argument in the given color;

*\colorbox{col_spec}{text}* sets its argument in a box with the given color as background;

*\fcolorbox{col_spec1}{col_spec2}{text}* like *\colorbox*, with a frame of *col_spec1* around a box of background color *col_spec2*; the two specifications must either both be defined ones, or both use the same model, which is given only once; for example, *\fcolorbox{red}{green}{Text}* sets 'Text' in the current text color on a green background with a red frame;

*\normalcolor* switches to the color that was active at the end of the preamble. Thus placing a *\color* command in the preamble can change the standard color for the whole document. This is the equivalent to *\normalfont* for font selection.

Normally one would try to define all the colors needed as names for the *col_spec* entries. This simplifies changing the color definition everywhere should fine-tuning be required after the initial printed results are seen. The same color definition can produce quite different effects on different printers. Even the display on the monitor is no reliable guide as to how the output will appear on paper.
6.2.1 The named color model

One very useful color model is called named and is based on the 68 predefined internal colors of the dvips PostScript driver. Sample names are BurntOrange or DarkOrchid. This model can be activated for other drivers with the option dvipsnames, in which case one can define colors as, for example

\definecolor{titlecol}{named}{DarkOrchid}

The color titlecol can then be used as \texttt{col_spec} in the various color commands.

The named colors can be defined with their own names if one invokes the option usenames, which effectively declares

\definecolor{BurntOrange}{named}{BurntOrange}

and so on, for all 68 colors.

It is possible to generate a palette of the named colors by processing the following short \LaTeX file and sending the output to the desired printer.

\documentclass[12pt,a4paper]{article}
\usepackage[dvipsnames]{color}
\usepackage{multicol}
\pagestyle{empty}
\setlength{\oddsidemargin}{0pt}
\setlength{\textwidth}{16cm}
\setlength{\textwidth}{22cm}
\setlength{\parindent}{0pt}
\setlength{\parskip}{0pt}
\begin{document}
\renewcommand*{\DefineNamedColor}[4]{% 
\textcolor{named}{#2}{\rule{7mm}{7mm}}\quad \texttt{#2}\strut\}
\begin{center}\Large Named colors in \texttt{dvipsnam.def}\end{center}
\begin{multicols}{3}
\input{dvipsnam.def}
\end{multicols}
\end{document}

Remember, each printer can reproduce the colors differently, so it is important to test this table with every color printer that might be used.

Exercise 6.3: Copy the above lines to a file named \texttt{palette.tex} or copy it from the enclosed CD from \texttt{books/Kopka_and_Daly}. Process it and view the output on a color monitor or send it to a color printer.
Floating tables and figures

Whether a figure is imported with \includegraphics (Chapter 6) or produced with the picture environment of Chapter 13, it is inserted in the text where the drawing or importing commands are issued, coming between the previous and following texts. This can present the same difficulties as for tables, described in Section 4.8.5: if the figure is so high that it no longer fits on the current page, that page is prematurely terminated and the figure is placed at the top of the next page, with too much empty space left on the original page.

In Section 4.8.5 we mention how this problem is solved for tables. Here we now give a complete description of the float procedures that apply to both figures and tables.

7.1 Float placement

\LaTeX does make it possible to float figures and tables, together with their headlines and captions, to an appropriate location without interrupting the text. This is invoked with the environments

\begin{figure}[where] figure \end{figure}
\begin{figure*}[where] figure \end{figure*}
\begin{table}[where] table \end{table}
\begin{table*}[where] table \end{table*}

The *-forms apply only to the two-column page format and insert the figure or table across both columns instead of the normal single column. They function exactly the same as the standard forms when the page format is single column.

In the above syntax, figure and table are the texts for the contents of the float, either a picture or tabular environment, or an \includegraphics command, together with a possible \caption command, as described later in Section 7.4.
The argument *where* specifies the allowed locations for the figure or table. There are several possibilities so that *where* consists of from zero to four letters with the following meanings:

h *here*: the float may appear at that point in the text where the environment is typed in; this is not permitted for the *-form; see also the *here* package and the H parameter, page 178;

t *top*: the float may appear at the top of the current page, provided there is enough room for both it and the previous text; if this is not the case, it is added at the top of the next page; the subsequent text continues on the current page until the next normal page break; (for two-column format, read *column* in place of *page*);

b *bottom*: the float may be placed at the bottom of the page; the subsequent text continues until the room left on the current page is just enough for the float; if there is already insufficient room, the float will be put at the bottom of the next page; this is not permitted for the *-form;

p *page*: the float may be put on a special page (or column) containing only figures and/or tables;

! used together with any combination of the other letters, suspends all the spacing and number restrictions described in Section 7.3.

The argument values may be combined to allow several possibilities. If none is given, **\LaTeX** assumes the standard combination *tbp*.

The placement arguments permit certain possibilities for locating the float, but the actual insertion takes place *at the earliest possible point* in accordance with the following rules:

- no float appears on a page prior to that where it is defined;
- figures and tables are output in the order in which they were defined in the text, so that no float appears before a previously defined float of the same type; figures and tables may, however, be mixed in their output sequence; in two-column format, the double column *-floats may also appear out of sequence;
- floats will be located only at one of the allowed positions given by the placement argument *where*; without an argument, the standard combination *tbp* is used;
- unless the ! is included in *where*, the positioning obeys the limitations of the style parameters described in Section 7.3;
- for the combination *ht*, the argument h takes priority; the float is inserted at the point of its definition even if there is enough room for it at the top of the page.
When any of the commands `\clearpage`, `\cleardoublepage`, or `\end{document}` is given, all floats that have not yet been output will be printed on a separate page or column regardless of their placement arguments.

Sometimes \LaTeX gets stuck on a float which holds up the entire queue until the end of the document. One way to clear the queue is to issue `\clearpage` right after the troublesome float. However, that would insert a new page at that point, something that may not be desired. The package `afterpage` in the tools collection (Section B.5.4) provides the command `\afterpage` which executes its argument at the end of the current page. Thus `\afterpage{\clearpage}` solves this problem by delaying the `\clearpage` command until the actual page break.

### 7.2 Postponing floats

Occasionally one wants to prevent floats from appearing on a certain page, for example at the top of a title page. (\LaTeX automatically corrects that case.) However, there are other situations where a float should be suppressed temporarily. One might want it at the top of a page, but not before the start of the section that refers to it. The command

\begin{verbatim}
\suppressfloats[loc]
\end{verbatim}

sees to it that for the current page only no further floats of the specified placement `loc` should appear. If the optional `loc` is omitted, all floats are suppressed; otherwise `loc` may be either `t` or `b`, but not both.

Note that `\suppressfloats` does not suppress all floats for the current page, but only further ones that come between the issuing of this command and the end of the page. Thus it is still possible for floats from a previous section to appear on the page.

Alternatively, the package `flafter` (Section B.5.3) may be loaded to ensure that all floats appear only after their position in the text.

The command `\suppressfloats` and the location parameter `!` are attempts to give the author more control over the sometimes capricious actions of float placement.

### 7.3 Style parameters for floats

There are a number of style parameters that influence the placement of floats, which may be altered by the user as desired.

- **topnumber**
  The maximum number of floats that may appear at the top of a page.

- **bottomnumber**
  The maximum number of floats that may appear at the bottom of a page.
Chapter 7. Floating tables and figures

**totalnumber**
The maximum number of floats that may appear on any page regardless of position.

**dbltopnumber**
The same as `topnumber` but for floats that extend over both columns in two-column page format.

The above parameters are all **counters** and may be reset to new values with the command `\setcounter{ctr}{num}`, where `ctr` is the name of the counter and `num` the new value that it is to take on.

**\topfraction**
A decimal number that specifies what fraction of the page may be used for floats at the top.

**\bottomfraction**
A decimal number that specifies what fraction of the page may be used for floats at the bottom.

**\textfraction**
The fraction of a page that must be filled with text. This is a minimum, so that the fraction available for floats, whether top or bottom, can never be more than $1 - \textfraction$.

**\floatpagefraction**
The smallest fraction of a float page that is to be filled with floats before a new page is called.

**\dbltopfraction**
The same as `\topfraction` but for double column floats in two-column page format.

**\dblfloatpagefraction**
The same as `\floatpagefraction` but for double column floats in two-column page format.

These style parameters are altered with `\renewcommand{cmd}{frac}` where `cmd` stands for the parameter name and `frac` for the new decimal value, which in every case must be less than 1.

**\floatsep**
The vertical spacing between floats appearing either at the top or at the bottom of a page.

**\textfloatsep**
The vertical spacing between floats and text, for both top and bottom floats.

**\intextsep**
The vertical spacing above and below a float that appears in the middle of a text page with the `h` placement argument.

**\dblfloatsep**
The same as `\floatsep` but for double column floats in two-column page format.

**\dbltextfloatsep**
The same as `\textfloatsep` but for double column floats in two-column page format.
This group of style declarations are rubber lengths that may be changed with the \setlength command (Section 2.4.2).

\topfigrule
A command that is executed after a float at the top of a page. It may be used to add a rule to separate the float from the main text. Whatever it adds must have zero height.

\botfigrule
Similar to \topfigrule, but is executed before a float that appears at the bottom of a page.

\dblfigrule
Similar to \topfigrule, but for double column floats.

These three commands normally do nothing, but they may be redefined if necessary. For example, to add a rule of thickness 0.4 pt below a top float,

\renewcommand{\topfigrule}{\vspace*{-3pt}\rule{\columnwidth}{0.4pt}\vspace{2.6pt} }

Because of the negative argument in \vspace*, the total vertical spacing is zero, as required.

All the one-column style parameters also function within the two-column page format, but they apply only to floats that fill up one column.

If the style parameters are set to new values within the preamble, they apply from the first page onwards. However, if they are changed within a document, they do not take effect until the next page.

## 7.4 Float captions

A figure caption or table title is produced with the command

\caption[short_title]{caption_text}

inside the figure or table environment. The caption_text is the text that is printed with the float and may be fairly long. The short_title is optional and is the text that appears in the list of figures or tables (Section 3.4.4). If it is missing, it is set equal to caption_text. The short_title should be given if the caption_text is longer than about 300 characters, or if it is more than one line long.

In the table environment, the \caption command generates a title of the form ‘Table n: caption_text’, and in the figure environment ‘Figure n: caption_text’, where n is a sequential number that is automatically incremented. In document class article, the figures and tables are numbered from 1 through to the end of the document. For the report and book classes, they are numbered within each chapter in the form c.n, where c is the current chapter number and n is the sequential number reset to 1 at the start of each chapter. Figures and tables are numbered independently of one another.
Chapter 7. Floating tables and figures

Table 7.1: Computer Center Budget for 2004

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Item</th>
<th>51505</th>
<th>52201</th>
<th>53998</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Maintenance</td>
<td>130 000</td>
<td></td>
<td>15 000</td>
<td>145 000</td>
</tr>
<tr>
<td>1.2</td>
<td>Network costs</td>
<td>5 000</td>
<td>23 000</td>
<td></td>
<td>28 000</td>
</tr>
<tr>
<td>1.3</td>
<td>Repairs</td>
<td>25 000</td>
<td>6 000</td>
<td></td>
<td>31 000</td>
</tr>
<tr>
<td>1.4</td>
<td>Expendables</td>
<td></td>
<td>68 000</td>
<td></td>
<td>68 000</td>
</tr>
<tr>
<td>1.</td>
<td>Total</td>
<td>160 000</td>
<td>74 000</td>
<td>38 000</td>
<td>272 000</td>
</tr>
</tbody>
</table>

The \caption command may be omitted if numbering is unwanted, since any text included in the float environment will accompany the contents. The advantages of \caption over simple text are the automatic numbering and entries in the lists of figures and tables. However, a manual entry in these lists may be made as described in Section 3.4.4 using

\texttt{\textbackslash addcontentsline and \textbackslash addtocontents}

A title, or headline, is produced with \caption when the command comes at the beginning of the material in the float environment: the number and text are printed above the table or figure. A caption is added below the object if the command comes after all the other float commands. In other words, the \caption is just another item within the float and whether its text appears at the top (as title) or below (as caption) depends on how the user places it.

The \texttt{caption.text} will be centered if it is shorter than one line, otherwise it is set as a normal paragraph. The total width may be adjusted to that of the table or figure by placing the command inside a parbox or minipage. For example,

\texttt{\parbox{width}{\caption{caption.text}}}

The following demonstrations contain further examples of how text, table, and picture commands may be combined in floats.

7.5 Float examples

The first two tables are produced with the following texts:

\texttt{\begin{table} \caption{Computer Center Budget for 2004} \begin{tabular}{|l|l||r|r|r|r|} ... ... \end{tabular} \end{table}}

(This text was typed in before the last paragraph of the previous section. The second table is entered here in the current text.)
7.5. Float examples

Table 7.2: Estimates for 2004. A continuation of the previous budget is no longer practical since, with the installation of the new computing system in 2003, the operating conditions have been completely overhauled.

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Item</th>
<th>51505</th>
<th>52201</th>
<th>53998</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Maintenance</td>
<td>240 000</td>
<td></td>
<td></td>
<td>240 000</td>
</tr>
<tr>
<td>1.2</td>
<td>Line costs</td>
<td>12 000</td>
<td>8 000</td>
<td>36 000</td>
<td>56 000</td>
</tr>
<tr>
<td>1.3</td>
<td>Training</td>
<td></td>
<td></td>
<td>50 000</td>
<td>50 000</td>
</tr>
<tr>
<td>1.4</td>
<td>Expansion</td>
<td>80 000</td>
<td>3 000</td>
<td></td>
<td>83 000</td>
</tr>
<tr>
<td>1.5</td>
<td>Expendables</td>
<td></td>
<td>42 000</td>
<td></td>
<td>42 000</td>
</tr>
<tr>
<td>1.</td>
<td>Total</td>
<td>332 000</td>
<td>53 000</td>
<td>86 000</td>
<td>471 000</td>
</tr>
</tbody>
</table>

Because the placement argument is missing from the \textbf{table} environments, the standard values \texttt{tbp} are used. The first was placed at the top on the facing page because it was typed in early enough (during the last section) that there was still room for it there. Then the command \texttt{\suppressthfloats} (Section 7.2) was issued to prevent any more floats appearing on the same page. The second table is therefore forced to float to the top of this page.

Narrow figures or tables may be set beside each other, as shown in the following example (which appears at the bottom of this page). (The \texttt{picture} environment is explained in Chapter 13.)

\begin{figure}[b]
\setlength{\unitlength}{1cm}
\begin{minipage}[t]{5.0cm}
\begin{picture}(5.0,2.5) \end{picture}
\caption{Left}
\end{minipage}
\begin{minipage}[t]{6.0cm}
\begin{picture}(6.0,3.0) \end{picture}
\caption{Right}
\end{minipage}
\end{figure}
The two figures along with their captions are each set in a `minipage` environment of widths 5 and 6 cm. The minipages are separated from one another by an `\hfill` space. The positioning argument \texttt{t} has the effect that the minipages are aligned along their first lines (Section 4.7.3). The entire structure within the `figure` environment floats as a single entity.

The question might now arise as to why, since the two figures have unequal heights and are supposedly aligned vertically along their top lines, their bottom edges are at the same level. The explanation is that a `picture` environment establishes an LR box (Section 4.7.1) to contain all the picture commands, and that is viewed by \LaTeX{} as a single line of output text with the baseline at the bottom edge of the picture. In both minipages, the `picture` environment is the first entry and is therefore the first logical line of text. It is these baselines that are taken for the vertical alignment of the minipages.

If the two pictures were to be aligned along their top edges, it would be necessary to include a dummy first line (Section 4.7.4) in each minipage before the `picture` environments. This could be something like `\mbox{}`, for example.

The application of box commands within a float permits completely free positioning. If the caption text is to appear, say, beside the table or figure, instead of above or below it, the objects may be put into minipages or parboxes with suitable alignment arguments. Here is an example:

\begin{table}[b]
\centerline{\textbf{Results and Seat Distribution of the…}}
\mbox{\small}
\begin{minipage}[b]{7.7cm}
\begin{tabular}{ll}
Pros & 140 \\
Cons & 186 \\
\hline
Ups & 60 \\
Downs & 42
\end{tabular}
\end{minipage}
\end{table}

**Results and Seat Distribution of the 2003 General Election**

<table>
<thead>
<tr>
<th>Party</th>
<th>Votes</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pros</td>
<td>15 031 287</td>
<td>37.7</td>
</tr>
<tr>
<td>Cons</td>
<td>12 637 418</td>
<td>31.7</td>
</tr>
<tr>
<td>Ups</td>
<td>6 499 210</td>
<td>16.3</td>
</tr>
<tr>
<td>Downs</td>
<td>4 486 514</td>
<td>11.3</td>
</tr>
<tr>
<td>Others</td>
<td>1 201 078</td>
<td>3.0</td>
</tr>
</tbody>
</table>

**Seat Distribution in the Assembly of 2004–2006**

**Results:** 2003 General Election
7.6 References to figures and tables in text

The automatic numbering of tables and figures has the consequence that the author does not know the numbers at the time of writing. Since he or she would like to be able to refer to these objects by number, as ‘see Figure 3’ or ‘Table 5 illustrates’, another means of referencing must be found. It is not sufficient to keep track of the number of \caption calls that have been made, since the document may not be written in the order in which it finally appears, and new figures or tables may be inserted during revision, or some removed.

These problems are solved with the \LaTeX cross-reference system, described in more detail in Section 9.2.1. The basic commands are

\begin{verbatim}
\label{name} \ref{name}
\end{verbatim}

which assign a keyword name to the figure or table number that may be used as reference in the text. The keyword name may be any combination of letters, numbers or symbols. The assignment is made with the \label command given anywhere within the caption text of the \caption command; in the main text, the command \ref inserts the number that is associated with its keyword.

This is best shown with an example. The budget table on page 174 was actually written as
so that Table \ref{budget04} produces ‘Table 7.1’ when used in the main text.

There is a second referencing command \pageref to generate the page number where the referenced object is to be found. For example, a few lines up, the input text on page \pageref{budget04} was used to create the text ‘... on page 174’.

\section{Some float packages}

\subsection{Placing a float right here}

The float placement parameter h, described on page 170, only permits a float to be placed at the current location, it does not force it to appear there. With Frank Mittelbach’s package \texttt{here}, an additional parameter H is provided which places the float immediately where it appears in the source text. If there is not enough room for it on the current page, a \texttt{pagebreak} command is first issued. This will most likely lead to an ugly page break.

\subsection{Flowing text around a float}

The package \texttt{floatflt} by Mats Dahlgren (based on the earlier and now obsolete \texttt{floatfig} by Thomas Kneser) permits one to mix text and floats side by side, with the text flowing around the float at the end, as in the next paragraph.

A figure is placed within the text as shown in the box at the right. Here \texttt{pos} determines on which side the figure is to appear; it may be r (right), l (left), p (right for odd, left for even pages), or v (like p unless a package option is given, in which case, it is used). The mandatory argument \texttt{wth} is the width of the figure itself; in fact, some additional space will be included for a border between text and figure. It is advisable to include a \texttt{centering} command at the start of the figure material.

The material in the \texttt{floatingfigure} environment need not actually be a figure. In the above example, some boxed text in a \texttt{minipage} was used for demonstration purposes. However, if the \texttt{caption} command is included in the environment, then a figure number will be printed, in sequence with those of the regular \texttt{figure} environment.

Rather than giving the \texttt{pos} parameter each time, one can specify a default value as an option when loading the package with \texttt{usepackage:}
7.7. Some float packages

Some float packages \begin{rflt} (right), \begin{lflt} (left), or \begin{vflt} (variable, i.e. p). If no option is given, the default is \begin{vflt}. To include a table with text, the usage is slightly different, as shown at the left. That is, the entire table appears in place of the \texttt{with} for a figure, and only the caption in placed in the environment itself.

Both these environments are to be placed at the start of the paragraph that is to flow around the figure or table. The flowing cannot start in the middle of a paragraph.

There is some very tricky coding to get the flowing text to work properly, and needless to say, there are many things that can go wrong. Please read the accompanying manual for more details.

7.7.3 Defining new floats and styles

Package: \texttt{float}

For authors who are not satisfied with the standard float style, or who need additional floating environments, we recommend the \texttt{float} package by Anselm Lingnau. With this package, one defines a new float type with

\begin{verbatim}
\newfloat{name}{loc}{ext}[within]
\end{verbatim}

where \texttt{name} is the name of the new float environment, \texttt{loc} is the default placement letters (selection of \texttt{tbpH}), \texttt{ext} is the extension of the file containing the list of this float type, and the optional \texttt{within} is the name of a counter (e.g., \texttt{chapter}) within which the numbering is reset.

For example, to define a \texttt{plate} environment that behaves just like \texttt{figure} but which is numbered separately as \texttt{Plate xxx}, with the numbers reset within each chapter, give

\begin{verbatim}
\newfloat{plate}{p}{lop}[chapter]
\floatname{plate}{Plate}
\end{verbatim}

The first line defines the new environment, with default placement on a float page. The second states that the captions associated with this float are to start with the word 'Plate'; without this, they would be labeled as 'plate', the same as the environment name itself.

The information for the list of plates is stored in a file with extension \texttt{.lop}, in the same way that the list of figures is in a file with extension \texttt{.lof} (Section 3.4.4). This list is printed by reading in this file with the command

\begin{verbatim}
\listof{plate}{List of Plates}
\end{verbatim}

where 'List of Plates' is the list title.

The default placement parameters for any float environment, including the standard \texttt{figure} and \texttt{table} ones, can be redefined with
\floatplacement\{name\}\{loc\}

The float package also defines 4 float styles:

plain much like standard \LaTeX{} except the caption is always below the float contents;

plaintop like plain but with the caption always above the contents;

boxed in which the contents are in a framed box with the caption below it;

ruled with the caption above, and with rules above, below, and between caption and float contents.

One selects a float style by issuing \floatstyle\{style\}, where style is one of the above possibilities. The style then applies to all new float environments defined with \newfloat, until a new style is issued.

Note: the style selection affects the subsequent float definitions, not the float environments that are then used. To change the float style for the standard figure and table floats, issue

\floatstyle\{ruled\}
\restylefloat\{figure\}
\restylefloat\{table\}

The float package also contains the H placement parameter for ‘here and I mean it’, so it is not necessary to include the here package with it.
\LaTeX allows the user to define his or her own commands and environments. However, since these make extensive use of the \LaTeX counters and lengths, we will first present a more detailed discussion of these objects and how they may be manipulated.

8.1 Counters

8.1.1 \LaTeX counters

\LaTeX manages a number of counters by giving them initial values at the start and by changing these values when certain commands are called. Most of these counters have the same name as the commands that alter them:

\begin{verbatim}
part  chapter  paragraph  figure  enumi
section subparagraph table enumii
subsection page footnote enumiii
subsubsection equation mpfootnote enumiv
\end{verbatim}

The meanings of most of these counters are obvious from their names and need no further explanation. The counters \texttt{enumi}...\texttt{enumiv} refer to the four levels of the \texttt{enumerate} environment (Sections 4.3.4 and 4.3.5), while the counter \texttt{mpfootnote} controls the footnote numbering within the \texttt{minipage} environment (Section 4.10.4).

The value of a counter is an integer number, usually non-negative. A command may output several numbers at once: the current \texttt{\subsection} command outputs 8.1.1, which addresses three counters in all. For example, the \texttt{\subsection} command increments the value of the \texttt{subsection} counter by one, and prints the values of the \texttt{chapter}, \texttt{section}, and \texttt{subsection} counters, separated by periods. At the same time, this command sets the \texttt{subsubsection} counter to zero.
8.1.2 User-defined counters

The user may create new counters with the command
\newcounter{counter_name}[in_counter]

where counter_name is the name of the newly established counter. This may be any combination of letters that is not already the name of an existing counter. Thus none of the names of the \LaTeX counters listed above may be used as counter_name nor any name of a previously defined user counter.

The optional argument in_counter is the name of another counter that already exists (\LaTeX or user defined) and has the effect that the newly defined counter is reset to zero whenever in_counter is incremented by one of the commands \stepcounter or \refstepcounter (see below). For example, the subsection counter is reset to zero whenever the section counter is incremented.

The \newcounter command may not appear in any file that is read in with the \include command (Section 9.1.2). It is therefore best to put all \newcounter commands into the preamble.

8.1.3 Changing counter values

Every counter, whether \LaTeX or user defined, has an initial value of zero; this can be altered with the following commands:
\setcounter{counter}{num}
This command is self-explanatory: the specified counter is assigned the integer value num.

\addtocounter{counter}{num}
With this command, the value of the counter is increased by the integer num, which may be positive or negative.

\stepcounter{counter}
The value of the counter is increased by one, and at the same time all its sub-counters (that is, those that have this counter as their in_counter) are reset to zero (see above).

\refstepcounter{counter}
This command has the same effect as \stepcounter but also makes counter the current counter for the cross-referencing command \label (see Section 9.2.1).

This last command may be applied, for example, within the figure or table environments when the \caption command is missing and yet there is to be a reference in the text to the figure or table number using the \ref command. Then \refstepcounter{figure} or \refstepcounter{table} is given within that float environment to bring the corresponding
counter to the right value and to allow that value to be assigned a keyword with the \label command (Section 9.2.1).

The value of a counter may be treated as a number with the command

\value{counter}

which does not change the value at all. This command is mostly used in connection with \setcounter or \addtocounter. For example, if the user-defined counter mypage has been created, it may be set to the same value as the page counter page by giving \setcounter{mypage}{\value{page}}.

8.1.4 Printing counter values

The numerical value in a counter can be printed with the commands

\arabic{counter} as an Arabic number,
\Roman{counter} as a capital Roman numeral,
\roman{counter} as a lower case Roman numeral,
\alph{counter} as a lower case letter,
\Alph{counter} as a capital letter,
\fnsymbol{counter} as a footnote symbol.

For the commands \alph and \Alph, the numbers 1...26 correspond to the letters a...z and A...Z. It is up to the user to ensure that the counter value lies within this range. For \fnsymbol, the numbers 1...9 are output as the symbols *, †, ‡, §, ¶, ‖, **, ††, ‡‡. Here again, the user must take care that the counter does not reach a value of 10 or more.

For each counter, a command of the form

\thecounter

is also available, consisting of \the immediately followed by the name of that counter, such as \thepage. This type of command is initially identical to \arabic{counter}, but may be redefined to be composed of several counter commands. In the document classes book and report, for example, the command \thesection is defined in terms of both the chapter and section numbers: \arabic{chapter}.\arabic{section}. Here is the result of printing \thesection at this point: 8.1.

The automatic printing of counter values such as the page, equation, or sectioning numbers is accomplished by means of calls to the appropriate \thecounter commands. If a different format for some automatic numbering is desired, say alphabetical equation numbers, the definition of the corresponding \thecounter command can be altered using the methods described in Section 8.3.

\textbf{Exercise 8.1:} Take your standard exercise.tex file and print out the final values of the \LaTeX counters with \arabic{counter} commands at the end. Change some the values with the \setcounter and \addtocounter commands and print the values out once more.
8.2 Lengths

It has been constantly pointed out in all the descriptions of the length parameters such as \textwidth or \parskip that new values may be assigned with the \setlength command. Some of these parameters expect rubber length values that may stretch or shrink. These are mainly parameters that produce vertical spacing. The types of length units, both fixed and rubber, are described in detail in Section 2.4. This will not be repeated here, but rather additional commands for assigning and handling lengths are discussed in this section.

The standard \LaTeX method for assigning a value to a length parameter is with the command
\[
\setlength{\text{length cmd}}{\text{length spec}}
\]
where \text{length spec} may be a length specification (with units) or another length parameter. In the latter case, \text{length cmd} takes on the current value of that other parameter. Thus with \setlength{\text{rightmargin}}{\text{leftmargin}} the right-hand margin in a list environment is set to the same value as that of the left-hand margin.

Lengths may be increased with
\[
\setlength{\text{addtolength}}{\text{length cmd}}{\text{length spec}}
\] which adds \text{length spec} to the value of the length parameter \text{length cmd}. A negative value for \text{length spec} decreases \text{length cmd} by that amount. Once again, another length parameter may be used for \text{length spec}, with or without a preceding minus sign, and its value will be added or subtracted. A decimal number just before a length parameter multiplies its value by that quantity: 0.5\textwidth means half the width of the text column and 2\parskip twice the inter-paragraph spacing.

With the command
\[
\setwidth{\text{settowidth}}{\text{length cmd}}{\text{text}}
\]
the length parameter \text{length cmd} is set equal to the natural length of a piece of text.

Similarly the commands
\[
\settoheight{\text{settodepth}}{\text{length cmd}}{\text{text}}
\]
set the \text{length cmd} equal to the height and depth of the \text{text} above and below the baseline, respectively.

Finally, the command
\[
\setstretch{\text{decimal num}}
\]
yields a rubber length that is \text{decimal num} times as stretchable as \text{fill} (Section 2.4.2).

A user-defined length parameter is created with
8.3 User-defined commands

New commands may be defined or redefined under \LaTeX with the commands

\begin{verbatim}
\newcommand{\com_name}[narg][opt]{def}
\renewcommand{\com_name}[narg][opt]{def}
\end{verbatim}

The first version is used to define a command \texttt{\com_name} that does not yet exist. Its name may be any combination of letters that do not form the name of another command. The second version redefines an already existing command \texttt{\com_name}. In both cases, an error message is printed if the incorrect variant is called. The first optional argument \texttt{narg} is a number between 1 and 9 specifying how many arguments the new or altered command is to have. A second optional argument \texttt{opt} gives the default value for an optional argument that the new command may take. The actual definition of the command is contained in the text \texttt{def}.

8.3.1 Commands without arguments

We will first illustrate the use of the \texttt{\newcommand} without the optional argument [\texttt{narg}]. This form is applied when a fixed combination of \LaTeX or user commands is to be repeated frequently as a command with its own name. For example, the structure $x_1, \ldots, x_n$, called an x-vector, often occurs in mathematical formulas and is formed in math mode with $x_1, \ldots, x_n$. Typing

\begin{verbatim}
\newcommand{\xvec}{x_1, \ldots, x_n}
\end{verbatim}

creates a new command named \texttt{\xvec} that may be called and used just like any other command. When called, it inserts the sequence of text and commands, in this case $x_1, \ldots, x_n$, into the current text exactly as
if one had typed it oneself. In fact, this is precisely what happens: when \texttt{\textbackslash xvec} is called, it is its definition that goes into the \LaTeX processing.

Since the new command \texttt{\textbackslash xvec} contains a math command (the subscript command \texttt{\textbackslash _}), it may only be called within math mode. Thus $\texttt{\textbackslash xvec}$ is required to produce $x_1, \ldots, x_n$ in text mode. It might therefore seem a good idea to include the switching to math mode in the definition itself, as

\begin{verbatim}
\newcommand{\xvec}{${x_1}, \ldots, {x_n}$}
\end{verbatim}

Now \texttt{\textbackslash xvec} yields $x_1, \ldots, x_n$. However, this command may only be applied in text and never in math mode. There is a trick to enable the command to be called in both modes: define it as

\begin{verbatim}
\newcommand{\xvec}{\ensuremath{\texttt{x_1}, \ldots, \texttt{x_n}}}
\end{verbatim}

Now both \texttt{\textbackslash xvec} and $\texttt{\textbackslash xvec}$ are allowed, both with the same result.

The above text mode example was actually written as \texttt{\textbackslash xvec{}}, since \TeX treats it as a command without arguments, terminating its name with the first non-letter that it finds. If this character is a blank, it only ends the command name and does not insert interword spacing (Section 2.1). Thus \texttt{\textbackslash xvec and ...} produces \enquote{$x_1, \ldots, x_n$ and ...} without any spacing between. This problem is solved by adding a space command \texttt{\_} or the empty structure \texttt{\{} after the command name, in this case as \texttt{\textbackslash xvec\_} or \texttt{\textbackslash xvec{}).

It would also have been possible to include the blank in the definition of \texttt{\textbackslash xvec}, as \texttt{\{\ensuremath{\texttt{x_1}, \ldots, \texttt{x_n}} \}}. Now the blank following the command name is still removed, but the command itself inserts one to make up for it. However, this is not recommended practice since this \enquote{programmed} blank will always be present, even when some punctuation or other symbol directly follows the command.

A better solution is provided by the \texttt{xspace} package in the tools collection (Section B.5.4). With this package, one adds the command \texttt{\textbackslash xspace} at the end of a definition where a space may appear:

\begin{verbatim}
\newcommand{\xvec}{\ensuremath{\texttt{x_1}, \ldots, \texttt{x_n}}\texttt{\textbackslash xspace}}
\end{verbatim}

This command prints a blank unless it is followed by punctuation, in which case it does nothing.

The different versions of the above example were all illustrated with the command \texttt{\textbackslash newcommand}, although in fact this command may only be used once to initiate a user-defined command that does not already exist. Once \texttt{\textbackslash xvec} has been created, a revised definition can only be given with the command \texttt{\textbackslash renewcommand}. This was indeed done with the second and subsequent definitions of \texttt{\textbackslash xvec}.

Following this example, the user may employ \texttt{\textbackslash newcommand} (or alternatively \texttt{\textbackslash renewcommand}) to combine any set of commands and text to form a command under a new name that may then be called whenever
necessary. In this way, a considerable amount of typing can be avoided and the possibility of error reduced, especially for complex mathematical structures.

If it is not known whether or not a command with the chosen name already exists, one may use instead

\providecommand{\com_name}[narg][opt]{def}

which has the same syntax as \newcommand and \renewcommand. The difference is that if the command already exists, the new definition will be ignored. The opposite effect (overwriting the current definition of a command without knowing if it already exists) can be achieved by first calling \providecommand to ensure the command exists, and then issuing the true definition with \renewcommand. However, this should be done only with great care!

Exercise 8.2: Define the commands \iint, \iiint, and \idotsint, to make the multiple integrals shown at the right as displayed formulas, or as the following text formulas: \int\int, \int\int\int, \int \cdots \int

Exercise 8.3: Change the \thechapter, \thesection, and \thesubsection commands so that for the document classes book and report the chapter numbering is done with capital letters, such as B, the section numbering with capital Roman numerals after the chapter letter, in the form B–III, and the subsection numbering with lower case Roman numerals following a comma: B–III,v.

Hint: the original versions of these commands in book and report are defined as

\newcommand{\thechapter}{\arabic{chapter}}
\newcommand{\thesection}{\thechapter.\arabic{section}}
\newcommand{\thesubsection}{\thesection.\arabic{subsection}}

Now apply \renewcommand to make the required changes.

8.3.2 Commands with arguments

In addition to the vector \(x_1, \ldots, x_n\), there are equivalent vectors \(y_1, \ldots, y_n\) and \(z_1, \ldots, z_n\) in mathematics. It would be possible to define commands \yvec and \zvec following the pattern for \xvec. However, it is also possible to define a generalized vector command and to specify the variable part as an argument. In the present example, the variable part is the letter \(x, y,\) or \(z\). A command with one variable part is created with the optional argument [1]. For example,

$$\newcommand{\avec}[1]{\ensuremath{#1_1, \ldots, #1_n}}$$

defines the general vector command \avec{arg}. Calling \avec{x} yields \(x_1, \ldots, x_n\) while invoking \avec{y} prints out \(y_1, \ldots, y_n\). The character
#1 in the command definition is a dummy argument representing the text of \textit{arg} that replaces all occurrences of #1 when the command is called. By imagining an x or a y at each location of the #1 in the definition, the desired structures in each case can be recognized.

The digit 1 in the dummy argument #1 seems at first to be rather pointless. In fact, for a command with only one argument it really has no meaning. However, its function becomes more obvious for commands with \textit{multiple} arguments. Let us say, for example, that we want a command to generate structures such as \( u_1, \ldots, u_m \) as well as \( v_1, \ldots, v_n \). This requires two arguments, one to specify the letter \( u, v, \) etc., and a second to determine the last subscript \( n, m, \) and so on. Such a command is created with

\[
\texttt{\newcommand{\anvec}{[2]\{\ensuremath{#1_1, \ldots, #1_{#2}}}}
\]

and invoked as \( \anvec{u}{n} \) for \( u_1, \ldots, u_n \) and as \( \anvec{v}{m} \) for \( u_1, \ldots, u_m \). The optional argument [2] for \texttt{\newcommand} says that the command being defined contains two arguments; in the definition part, #1 is replaced by the first argument and #2 by the second. By imagining \( u \) or \( v \) in place of #1 and \( n \) or \( m \) where #2 stands, one can see how the command \( \anvec{\text{arg1}}{\text{arg2}} \) operates.

This pattern may be carried on for even more arguments. With

\[
\texttt{\newcommand{\subvec}{[3]\{\ensuremath{#1_{#2}, \ldots, #1_{#3}}}}
\]

da command \texttt{\subvec} is defined with three arguments. It should be clear from the definition that calling \( \subvec{a}{i}{j} \) produces \( a_i, \ldots, a_j \).

A command argument that consists of only a single character need not be put into curly brackets \{ \} but may be given directly. If it is the first argument, it must be separated from the command name with a blank, as usual. Thus the sequence \( \subvec a i k \) is the same as \( \subvec{a}{i}{k} \), and \( \subvec x i n \) produces the same structure \( x_1, \ldots, x_n \) as our first user-defined example \texttt{\xvec}.

Arguments must be enclosed in curly brackets \{ \} when they contain more than one character, since the brackets indicate that the contents are to be treated as a unit. Thus \( \subvec{A}{ij}{lk} \) prints out \( A_{ij}, \ldots, A_{lk} \). The three replacement arguments are A for #1, ij for #2, and lk for #3.

Why does \( \subvec{A}{ij}{lk} \) produce \( A_{ij}, \ldots, A_{lk} \) and not the expected result \( A_{ij}, \ldots, A_{lk} \)? The answer is that, although the arguments within curly brackets are set as units into the definition text, the brackets themselves are not. The command text after replacement is \texttt{\ensuremath{A_{ij}, \ldots, A_{lk}}} so that only the first characters following the subscript symbols _ are actually lowered. In order that both letters be lowered, they must be seen as a unit within the command text, that is as \( A_{\{ij\}, \ldots, A_{\{lk\}} \). This may be achieved with the \texttt{\subvec} command by including an extra set of brackets in the arguments: \( \subvec{A}{\{ij\}}{\{lk\}} \). A better solution, however, is to put the brackets in the definition to begin with:

\[
\texttt{\renewcommand{\subvec}{[3]\{\ensuremath{#1_{\{#2\}}, \ldots, #1_{\{#3\}}}}}\]

which will always produce the desired result with only a single set of brackets per argument: \( \subvec{A}{ij}{lk} \) prints \( A_{ij}, \ldots, A_{lk} \).
8.3.3 Commands with an optional argument

As we have seen, many \LaTeX commands may take optional arguments, including the command \verb|\newcommand| itself. It is also possible to define user commands with one optional argument. The advantage of this is that, although an argument is provided, in most applications it will usually take some standard value that need not be given explicitly.

As an example, the user-defined vector command \verb|\subvec| in the last section has three arguments, for the letter and for the first and last subscripts. However, it may be that the letter is normally \(x\), so it makes sense to include it as an optional argument, which is to be specified only for a different letter. This is accomplished with

\begin{verbatim}
\renewcommand{\subvec}[3][x]{\ensuremath{#1_{#2},\ldots,#1_{#3}}}
\end{verbatim}

The difference between this and the previous definition is the addition of \([x]\) after the \([3]\) argument. This states that the first of the three arguments is to be optional, and its standard value is \(x\). Now \verb|\subvec{i}{j}| prints \(x_i,\ldots,x_j\) while \verb|\subvec[a]{1}{n}| produces \(a_1,\ldots,a_n\).

There may only be one optional argument in the user-defined command, and it will always be the first one, the \#1 in the definition.

8.3.4 Additional examples of user-defined commands

In the above explanation of user-defined commands, a very simple case of a vector structure was taken as an example. We would now like to demonstrate some more complex situations, in which counters, lengths, and even some special \TeX commands are applied.

**Example 1:** In Section 5.4.6, the \TeX commands \verb|\atop| and \verb|\choose| were presented as useful mathematical commands even for \LaTeX applications. Unfortunately, the syntax of these commands deviates considerably from that of the similar \LaTeX command \verb|\frac|. However,

\begin{verbatim}
\newcommand{\latop}[2]{{#1\atop#2}}
\newcommand{\lchoose}[2]{{#1\choose#2}}
\end{verbatim}

define two commands \verb|\latop| and \verb|\lchoose| that yield the same results with a syntax like that of \LaTeX: \verb|\latop{upper}{lower}|.

**Example 2:** The command \verb|\defbox{sample\_text}| is to set a box width equal to the length of the text \textit{sample\_text}. A subsequent call to the command \verb|\textbox{text}| centers \textit{text} within a frame with the same width as \textit{sample\_text}.

\begin{verbatim}
\newlength{\wdth}
\newcommand{\defbox}[1]{\settowidth{\wdth}{#1}}
\newcommand{\textbox}[1]{\framebox[\wdth]{#1}}
\end{verbatim}
First, a new length parameter \textwidth is created, then \defbox is defined so that \textwidth is set equal to the length of its argument (Section 8.2), and finally \textbox makes a framed box of that same width containing its argument, centered. (Do not name the length parameter \textwidth, for this already exists, Section 4.7.5.)

\defbox{as wide as this text}\textbox{as wide as this text}\\
\textbox{text}\\
\textbox{longer text}

\begin{verbatim}
Example 3: A footnote command \myftnote is to be created that behaves as the normal command \footnote{text} in putting text into a footnote, but instead of using numbers as the marker, it should take the symbols ∗ † ‡ § ¶ ∥ ∗∗ †† ‡‡ one after the other, starting again with the symbol ∗ on each new page. First a new counter must be established that will be reset to zero every time the page counter is incremented. This is done with (see Section 8.1.2)
\newcounter{myfn}[page]
\renewcommand{\thefootnote}{\fnsymbol{footnote}}
\newcommand{\myftnote}[1]{\setcounter{footnote}{\value{myfn}}%\footnote{#1}\stepcounter{myfn}}
\end{verbatim}

Making a user-defined counter myfn that is set to zero every time the page counter is incremented.* Next the command
\newcommand{\myftnote}[1]{\setcounter{footnote}{\value{myfn}}%\footnote{#1}\stepcounter{myfn}}

redefines the footnote marker to be that symbol in the sequence given by the counter \footnote (Sections 4.10.2 and 8.1.4). Now the actual new footnote command can be constructed with
\newcommand{\myftnote}[1]{\setcounter{footnote}{\value{myfn}}%\footnote{#1}\stepcounter{myfn}}

yielding the desired results. The user-defined command \myftnote possesses one argument, which is passed to the \LaTeX \footnote command after the \LaTeX counter \footnote has been set equal to the value of the user counter myfn. Once the command \footnote has been executed, the counter myfn is then incremented by one with \stepcounter{myfn}. This counter, however, is reset to zero whenever the page counter is incremented, that is, whenever a new page begins.

The footnote on the previous page was generated with the command \myftnote as described. It is now used here† and again here‡, demonstrating how the symbols have been reset on a new page.

\begin{verbatim}
Example 4: A command \alpheqn is to be set up so that once it has been called, the subsequent equations will all have the same number but be
\alpheqn
\end{verbatim}

*Actually the page counter is not incremented exactly at the end of the page; \LaTeX reads in the whole paragraph before it decides if and where a page break should occur. This can cause problems with resetting the myfn counter near the top of a page.
†another footnote
‡and yet another footnote
followed by letters $a$, $b$, ..., separated by a hyphen `-' from the number. The command \reseteqn restores the numbering scheme to its original style. Thus a sequence of equation numbers could be 4, 5, 6-a, 6-b, 7.

\newcounter{saveeqn}
\newcommand{\alpheqn}{\setcounter{saveeqn}{\value{equation}}%
  \stepcounter{saveeqn}\setcounter{equation}{0}%
  \renewcommand{\theequation}{\mbox{\arabic{saveeqn}-\alph{equation}}}}
\newcommand{\reseteqn}{\setcounter{equation}{\value{saveeqn}}%
  \renewcommand{\theequation}{\arabic{equation}}}

The example should be easy to comprehend with the help of the commands and counters in Section 8.1. The current value of counter equation is saved in the counter saveeqn and then incremented, while equation itself is set to zero. The form of the equation marker, \theequation, is redefined using these two counters. The equation numbering routines will operate on equation as usual, leaving saveeqn unchanged. The resetting command \reseteqn puts the value of saveeqn back into equation and restores the definition of \theequation.

This example is only appropriate for the document class article. For report and book, the definition of \theequation is

\arabic{chapter}.\arabic{equation}

The necessary modifications are left as an exercise for the user.

The \mbox command in the first \renewcommand{\theequation} defining the combined equation number is necessary because the result will be printed in math mode, where the hyphen `-' is interpreted as a binary operator (minus sign) with extra spacing between it and its two `operands', \arabic{saveeqn} and \alph{equation}. Thus 6 – $a$ would be output instead of 6-a. The \mbox command causes a temporary switch out of math into text mode.

**Example 5:** In Section 5.4.10, the international standards for mathematical typesetting are outlined, in which tensor and matrix variables should be set in a sans serif typeface, preferably italic or slanted. The math alphabet command \mathsf accomplishes this only for an upright font. The solution is to create a new math alphabet \mathsfsl for use in a \tensor command.

\DeclareMathAlphabet{\mathsfsl}{OT1}{cmss}{m}{sl}
\newcommand{\tensor}[1]{\mathsfsl{#1}}

The NFSS system of font attributes is presented in Section A.1, and the \DeclareMathAlphabet command is explained in Section A.3.3.

The Computer Modern sans serif fonts do not possess an italic form which is why the slanted shape is taken instead. If one were to select italic with \{i\} in place of \{sl\}, \LaTeX would automatically substitute the slanted version.
In Section 5.4.10, several new commands are suggested to simplify typing math according to the ISO standard. These are:

\newcommand{\me}{\mathrm{e}}
\newcommand{\mi}{\mathrm{i}}
\newcommand{\dif}{\mathrm{d}}
\renewcommand{\vec}[1]{\boldsymbol{#1}}

These permit the constants \(e\), \(i\), and differential operator \(d\) to be printed upright, and not italic. The redefinition of the \(\vec\) command requires the \(\texttt{AMS}\) math package \texttt{amsbsy}, either directly or as part of the \texttt{amsmath} package (Chapter 12).

Here one could also include the abbreviation for the electron volt, mentioned on page 30, and its multiples:

\newcommand{\eV}{\mbox{e\hspace{-.12em}V}}
\newcommand{\keV}{\mbox{k\eV}}
\newcommand{\MeV}{\mbox{M\eV}}

If these commands are used frequently in many different documents, we recommend saving them in a separate file, say \texttt{isomath.tex}, to be input at the start of those documents, with \texttt{\input{isomath}}.

**Exercise 8.4:** Define \TeX\ commands \texttt{\Lbrack} and \texttt{\Lbrace} in the same manner as in Example 1 for \texttt{\latop} and \texttt{\lchoose} corresponding to the \TeX\ commands \texttt{\brack} and \texttt{\brace}. These \TeX\ commands behave as \texttt{\choose} from Section 5.4.6 except that they enclose their contents in square brackets \texttt{\Lbrack} or curly braces \texttt{\Lbrace}. (Note that the names \texttt{\lbrack} and \texttt{\lbrace} are already defined and so should not be used for these commands.)

**Exercise 8.5:** Generalize Example 4 with a command \texttt{\vareqn{num}{type}} to make the subsequent equation numbers have the value \texttt{num} followed by a running number in square brackets printed as \texttt{\alphph\ldots\Roman}, as given by the argument type. For example, \texttt{33[A], 33[B]}, would result from calling \texttt{\vareqn{33}{\Alph}}.

**Exercise 8.6:** Generalize the integral commands in Exercise 8.2 to include an argument to represent the area of integration set centered below the entire symbol. Thus \texttt{\iint{(D)}}, \texttt{\iiint{V}}, and \texttt{\idotsint{G}} should produce:

\[
\iint{(D)} \quad \iiint{V} \quad \idotsint{G}
\]

Hint: the second command can be made simply with a subscript on the middle integral (but see \texttt{\limits} in Section 5.2.5). For the other two, negative horizontal shifting of the sub scripted symbol is needed using \texttt{\hspace{-\ldots}}.

**Exercise 8.7:** The \texttt{\tensor} command defined in Example 5 above sets tensor variables in a slanted, sans serif font. If matrices are to be set instead in an upright, sans serif font, define a command \texttt{\mtrx} to produce this.

Note: do not name this command \texttt{\matrix} because that already exists to produce matrices, not their variable names.
8.3.5 Conditional text

Practiced \TeX users will be familiar with the conditional commands that are available, both for \TeX and \LaTeX. However, their usage is not always straightforward and often requires extensive knowledge of \TeX's deeper principles. Leslie Lamport has provided a package named ifthen, extended by David Carlisle, which not only simplifies their application, but also gives them a \LaTeX syntax.

The package is loaded as usual with the command

\usepackage{ifthen}

in the preamble. It then makes available the two commands \ifthenelse and \whiledo, which have the following syntaxes:

\ifthenelse{test}{then_text}{else_text}
\whiledo{test}{do_text}

In both cases, test is a logical statement (explained below); for the first command, then_text or else_text is inserted into the text depending on whether test is (true) or (false). For the second command, the do_text is inserted (executed) as long as test evaluates to (true). (The do_text must alter the inputs to test or it will never stop!) The texts may also contain commands, or even define or redefine commands.

There are four types of basic logical statements which may be combined to form more complicated ones.

Testing numbers

To compare two numbers or commands that evaluate to numbers, simply put one of the relational operators <, =, or > between them, which stand for less than, equals, and greater than, respectively. The value of a counter may be tested by putting its name as the argument of the \value command. Examples:

\newcommand{\three}{3}
\ifthenelse{\three = 3}{O.K.}{What?}
\ifthenelse{\value{page} < 100}{Page xx}{Page xxx}

The first case prints 'O.K.' since \three is equal to 3; in the second case Page xx is printed if the current page number is less than 100, else Page xxx. (The blanks above are added for clarity, since spaces between arguments are always ignored.)

Whether a number is even or odd can be tested with \isodd

\ifthenelse{\isodd{\value{page}}}{odd}{even}
Chapter 8. User Customizations

Testing text

To test if two commands evaluate to the same piece of text, or if a command is defined as a certain string of text, use

\equal{string1}{string2}

where string1 and string2 are texts or commands that reduce to text. For example, with

\ifthenelse {\equal{name}{Fred}} {Fred} {??}

the text Frederick is inserted if \name has been defined to be Fred (with \newcommand), otherwise two question marks are printed.

Testing lengths

Another logical statement compares two lengths.

\lengthtest{relation}

where relation consists of two lengths or length commands separated by a relational operator <, =, or >. For example,

\newlength{\horiz} \newlength{\vert}
\newlength{\min}
\ifthenelse {\lengthtest{\horiz > \vert}}
{\setlength{\min}{\vert}} {\setlength{\min}{\horiz}}

sets \min to be the smaller of \horiz and \vert.

Testing switches

A boolean switch is a parameter that is either ⟨true⟩ or ⟨false⟩, also called a flag. Three commands exist to handle them:

\newboolean{string} creates a new switch
\setboolean{string}{value} assigns a value true or false
\boolean{string} tests its value

The last of these is used as test in \ifthenelse and \whiledo.

There are a number of internal \TeX switches that may also be tested (but never reset!). The most useful of these are @twoside and @twocolumn for checking if two-side or two-column modes are active. Since they contain the character @, they may only be used inside a class or package file (Appendix D).
Combining logical statements

Any of the above logical statements may be combined to form a more complex statement by means of the logical operators

\and \or \not \( \)

which should be straightforward to anyone familiar with boolean logic. For example, to set \textwidth\ to 10 cm if two-column mode is active or if \paperwidth\ is greater than 15 cm and the page counter is below 100,

\ifthenelse {\lengthtest{\textwidth > 10cm} \or \( \lengthtest{\paperwidth > 15cm} \and \value{page} < 100 \) } {\setlength{\textwidth}{10cm}} {}

will accomplish this.

Such conditionals are fairly complicated, and they are most appropriate for defining new commands that may have alternative actions, or for including in package and class files (Appendix D).

A relatively simple example: suppose one is uncertain whether British or American spelling is wanted by the publisher. One can write the document with both included in the text, with a switch in the preamble to make the final selection.

\newboolean{US}
\setboolean{US}{true} %For American spelling
%\setboolean{US}{false} %For British spelling
\newcommand{\USUK}[2]{\ifthenelse{\boolean{US}}{#1}{#2}}

Now \USUK\ is a command that prints its first or second argument according to the setting of the flag US. Thus in the text one may write

... the \USUK{color}{colour} of music ...

which will yield American spelling if \setboolean{US}{true} has been specified, otherwise the British version. In fact, different sections of the work could have different spelling simply by changing the value of the switch at the appropriate point.

As an example of \whiledo: one wishes to write some text $n$ times, where both the text and $n$ are to be variable.

\newcounter{mycount}
\newcommand{\replicate}[2]{\setcounter{mycount}{#1} \whiledo{\value{mycount}>0}{#2\addtocounter{mycount}{-1}}}

Now \replicate{30}{?} will print 30 question marks.

8.4 User-defined environments

Environments may be created or changed with the commands
8.4.1 Environments without arguments

Just as for user-defined commands, environments without the optional argument \texttt{narg} will be illustrated first. A user-defined environment named \texttt{sitquote} is created with

\begin{verbatim}
\newenvironment{sitquote}{\begin{quote}\small \itshape}{\end{quote}}
\end{verbatim}

which sets the text appearing between \texttt{\begin{sitquote} text \end{sitquote}} in the typeface \texttt{\small \itshape}, and indented on both sides from the main margins, as demonstrated here.

In this case, \texttt{beg\_def} consists of the command sequence \texttt{\begin{quote} \small \itshape while \texttt{end\_def} is simply \texttt{\end{quote}. Now the call

\begin{verbatim}
\begin{sitquote} text \end{sitquote} is the same as \begin{quote} \small \itshape text \end{quote}
\end{verbatim}
which produces the desired result.

This example does not appear to be very practical since the same effect can be achieved with less typing by adding \texttt{\texttt{\texttt{\texttt{\texttt{\small\itshape}}}} at the beginning of the quote environment. However, it is more consistent with the concept of logical markup introduced in Section 1.2.2. This special quotation environment may be used throughout the document without worrying about its typographical details, which are specified in its definition located in the document preamble. Doing it this way not only ensures consistency, it also simplifies any substitution that may be demanded later by a typographical expert.

Let us expand the previous example somewhat as follows:

\newcounter{com}
\newenvironment{comment}{\noindent\slshape Comment:\begin{quote}\small\itshape}{\stepcounter{com}\hfill(\arabic{com})\end{quote}}

where now \texttt{beg\_def} contains the text and commands

\texttt{\noindent\slshape Comment:\begin{quote}\small\itshape
and \texttt{end\_def} is

\texttt{\stepcounter{com}\hfill(\arabic{com})\end{quote}}

where \texttt{com} is a user counter created by the \texttt{\newcounter} command. Now since the command \texttt{\begin{comment}} inserts the text \texttt{beg\_def} at the start of the environment and \texttt{\end{comment}} the text \texttt{end\_def} at the finish, it should be clear that

\begin{comment}
This is a comment.
Comments should ...
... in round parentheses.
\end{comment}

will generate the following type of structure:

\texttt{Comment:}

\textit{This is a comment. Comments should be preceded by the word Comment: the text being in a small, italic typeface, indented on both sides from the main margins. Each comment receives a running comment number at the lower right in round parentheses. (1)}

The user should examine the sequence of commands with the replacement text of this example to see precisely what the effect of this environment is. Two weaknesses should become apparent: what would happen if \texttt{\begin{comment}} were called in the middle of a line of text without a blank line before, and what would happen if the last line of the comment text were so long that there was no more room for the running comment number on the same line?

The following revision removes these two problems:
With the \begin{sloppypar} command, the call to the environment always starts a new paragraph in which no overfull line can occur upon line breaking. If the comment number does not fit on the last line of text, a new line begins with the number right justified because of the command \hspace*{\fill}. Again the user should examine carefully just what is inserted into the processing between the \begin{comment} and the \end{comment}.

### 8.4.2 Environments with arguments

Passing arguments over to an environment is carried out exactly as for commands. As an example, the comment environment will be modified so that the name of the person making the comment is added after the word Comment:, and this name will be an argument when the environment is invoked.

\renewenvironment{comment}[1] {
  \begin{sloppypar}\noindent\slshape Comment: #1
  \begin{quote}\small\itshape
  \stepcounter{com}\hspace*{\fill}\end{quote}\end{sloppypar}
}\end{quote}\end{sloppypar}

The text
\begin{comment}{Helmut Kopka} This is a modified ...
\end{comment}
... environment argument \end{comment}
now produces

**Comment: Helmut Kopka**

*This is a modified comment. Comments should be preceded by the word Comment:, followed by the name of the commenter, with the text of the comment being in a small, italic typeface, indented on both sides from the main margins. Each comment receives a running comment number at the lower right in round parentheses. The name of the commenter is transferred as an environment argument.*

This example will now be modified once again by interchanging the comment number and the name of the commenter. Placing the running number after the word Comment: is no problem, for it is simply necessary to insert those commands from {end_def} at the location of the #1 dummy argument. However, putting the symbol #1 where the comment number used to be will produce an error message during the \LaTeX processing since
this violates the syntax of the `\newenvironment` command: `No dummy arguments shall appear within the \{end_def}.` If the dummy argument comes after `\begin{quote}`, the name will be printed at the wrong place, at the beginning of the comment text.

There is a trick to solve this problem:

\begin{verbatim}
\newsavebox{\comname}
\renewenvironment{comment}[1]
  {\begin{sloppypar}\noindent\stepcounter{com}\slshape
    Comment \arabic{com}\sbox{\comname}{#1}
  \begin{quote}\small\itshape
  \begin{quote}\small\itshape
  {\hspace*{\fill}}\usebox{\comname}\end{quote}\end{sloppypar}}
\end{verbatim}

The commands `\newsavebox`, `\sbox`, and `\usebox` are all described in Section 4.7.1. Here `\comname` is the name of a box that has been created with `\newsavebox`, in which the first argument, the commenter's name, is stored. With this new definition, a comment now appears as follows:

```
Comment 3

In this form, every comment is assigned a sequential number after the word Comment. The comment text appears as before, while the name of the commenter is entered as the environment argument and is placed at the right of the last line.  Helmut Kopka
```

The implementation of more than one argument for environments is the same as for commands and needs no further explanation.

### 8.4.3 Environments with an optional argument

Similarly, environments may be defined with an optional argument, just as commands. To take our last example once more, if we feel that most comments will be made by Helmut Kopka, we could alter the first line of the definition to

\begin{verbatim}
\renewenvironment{comment}[1][Helmut Kopka]{..}{..}
\end{verbatim}

Now it is only necessary to specify the name of the commenter if it is someone else. With

```
\begin{comment}[Patrick W. Daly]More than ...
  ... appropriate.\end{comment}
```

we obtain

```
Comment 4

More than one person may want to make a comment, but perhaps one person makes more than others do. An optional argument for the name is then appropriate.  Patrick W. Daly
```
Exercise 8.8: Extend the definition of the comment environment so that a page break cannot occur either between the heading ‘Comment n’ and the comment text or between the comment text and the commenter’s name.

Exercise 8.9: Create a new environment making use of the \texttt{minipage} environment, to be named \texttt{varpage} and possessing one argument that is a sample text to determine the width of the minipage. The call

\begin{varbox}{'As wide as this sample text'} \end{varbox}

should pack the enclosed text into a minipage that has a width equal to that of the text ‘As wide as this sample text’.

Hint: a user-defined length parameter must first be established, say \texttt{varwidth}. See Section 8.2 for details on assigning lengths to text widths.

Exercise 8.10: Generate an environment named \texttt{varlist} with two arguments that behaves as a generalization of the sample list in Section 4.4.3. The first argument is to be the item word that is printed on each call to \texttt{item}; the second is the numbering style of the item numeration. For example, with the call

\begin{varlist}{Sample}{\Alph} \end{varlist}

every \texttt{item} command within the environment should produce the sequence ‘Sample A’, ‘Sample B’, . . . . The indentation should be 1 cm larger than the width of the item word, which itself should be left justified within the label box.

Hint: once again a user-defined length, say \texttt{itemwidth}, is necessary for this solution. After the length of the item word has been stored with \texttt{settowidth}, the indentation may be set with

\setlength{\leftmargin}{\itemwidth}

to be equal to the width of the item word, and then with

\addtolength{\leftmargin}{1cm}

to be 1 cm larger. Length assignments for \texttt{labelwidth} and \texttt{labels} may be similarly set to appropriate values.

All further details may be obtained from Section 4.4.

8.5 Some comments on user-defined structures

8.5.1 Reusing sets of definitions

User definitions are created to alter page formats, to reorganize the document layout, to invent new structures, and to define shortcuts. That is, they either directly specify typographical elements like the text width, or indicate how new or existing logical elements are to be rendered typographically. As such, they belong in the preamble, where they can be conveniently redefined as needed, effecting the entire document.

Once a set of user definitions has been established, what is the best way to reuse them? There are several possibilities:
1. Write a new class file based on an existing one, incorporating the your own changes. This should be reserved for major changes to the layout or document organization, which will be used very frequently.

2. Write your definitions to a package file with the extension .sty to be loaded with the `\usepackage` command. You can take advantage of special package features, like internal commands and options.

3. Write the definitions to a file with the extension .tex and load it with `\input{filename}`. No extra features are possible, and the file can be loaded anywhere within the source file.

4. Just copy the definitions from one source file to the next.

Methods 1 and 2 are more sophisticated, and should be considered for user definitions that have a wide generality. They are explained in Appendix D. Method 3 is perfectly good for a personal collection of shortcut commands, or common (re)definitions that are applied to many documents. By having one depository for all such customizations, it is easier to maintain them centrally.

Finally method 4 is applicable for a short list of uncomplicated definitions.

### 8.5.2 Unwanted spaces

Occasionally user-defined structures generate spacing where none was expected, or more spacing than was desired. This is almost always due to blanks or new lines in the definition, included only to improve the legibility of the input text but interpreted as spacing when the structure is invoked.

For example, if the `%` character had been left out of the first line of the `\myftnote` definition on page 190, a new line would have been added to the command text at that point, and converted into a blank. This blank would be inserted between the previous word which should receive the footnote marker and the call to the `\footnote` command that actually generates the marker, with the result that it would be displaced from that word (for example, wrong * instead of right*).

At this point we should like to point out that many \LaTeX commands are invisible, in that they do not produce any text at the point where they are called. If such an invisible command is given separated by blanks from the surrounding text, it is possible that two blanks will appear.

For example `\rule{0pt}{0pt}` produces ‘For example produces’ twice the interword spacing between ‘example’ and ‘produces’. Invisible commands without arguments do not present this problem since the trailing blank acts solely as a command name terminator and disappears. Furthermore, the following \LaTeX commands and environments always remove the subsequent blanks even when arguments are present.
8.5.3 Identical command and counter names

In the previous examples a number of counters were introduced for application to specific commands or environments, such as myfn for the command \myfootnote on page 190, or com for the environment comment on page 197. In these cases, the counter and the corresponding command or environment were given different names. This was not necessary: counters may have a name that is identical to that of a command or environment. \LaTeX knows from the context whether the name refers to a counter or to a command/environment.

The different names were chosen in the examples in order to avoid confusion for the beginner. In fact, it is reasonable to give a counter the same name as the command or environment with which it is coupled, a practice that \LaTeX itself constantly employs (Section 8.1.1). In the above-mentioned examples, the counters could just as easily have been named myfootnote and comment to emphasize their interdependence with the command and environment of the same names.

8.5.4 Scope of user definitions

User structures that are defined within the preamble are valid for the entire document. Command and environment definitions that are made inside an environment remain in effect only until its end. Even their names are unknown to \LaTeX outside the environment in which they were defined. Thus if they are to be defined once again in another environment, it is \newcommand or \newenvironment that must be used and not the \renew versions.

For command and environment names that have been globally defined, that is, in the preamble, all further new definitions of these names must be made with the \renew version. However, these subsequent definitions will apply only locally within the environment in which they are declared; outside, the previous global definitions will be valid once more.

The same applies to structure definitions within nested environments. A definition in the outer environment is effective within all inner ones, but a new definition must be made at the deeper levels with \renewcommand or \renewenvironment. On leaving the inner environment, the new definition will no longer be operative, but rather the old one will be re-established.

Warning: Structures that have been created with \newsavebox or with \newcounter are globally defined. If they are given within an environment, their definitions remain effective even outside that environment. Similarly \setcounter functions globally, although \savebox does not.
Local definitions are really not to be recommended. It is far better to place all user definitions at the beginning, in the preamble, where they are most easily visible. A local redefinition of a standard command can lead to puzzling effects that are hard for others, or even for the author, to trace down. Of course, some special effects might just be wanted.

8.5.5 Order of definitions

User-defined structures may be nested within one another. If one user definition contains another user-defined structure, it is often the case that the inner one has already been defined. However, this is not necessary. *User definitions may contain other user structures that are defined afterwards.* What is important is that the other structure is defined before the first command is invoked.

For example, a normal sequence of definitions would be

```
\newcommand{\A}{defa}
\newcommand{\B}{defb}
\newcommand{\C}{\A \B}
```

where \(\C\) is defined in terms of \(\A\) and \(\B\). However, it is also permissible to write

```
\newcommand{\C}{\A \B}
\newcommand{\B}{defb}
\newcommand{\A}{defa}
```

further text with any number of calls to \(\A\), \(\B\) and \(\C\)

8.5.6 Nested definitions

User definitions may be nested inside one another. A structure such as

```
\newcommand{\outer}{{\newcommand{\inner}...}}
```

is permissible. The command that is defined with the name \(\inner\) is valid and known only within the command \(\outer\), according to the remarks on the scope of definitions on the facing page. Although the \TeX\ macros make copious use of nested command definitions to limit the lifetime of temporary commands, it is not recommended to nest \TeX\ definitions excessively since it is too easy to lose track of the bracketing. A forgotten bracket pair will produce an error message on the second call to the outer command, since the inner definition still exists as a leftover from the first call. Nevertheless, here is an example:

```
\newcommand{\twentylove}
{{\newcommand{\fivelove}
{\newcommand{\onelove}
{I love \LaTeX!}
\onelove \onelove \onelove \onelove \onelove}}}
\fivelove\fivelove\fivelove\fivelove\fivelove
```
The entry My opinion of \LaTeX: \twentylove now produces:

My opinion of \LaTeX:

I love \LaTeX! I love \LaTeX! I love \LaTeX! I love \LaTeX! I love \LaTeX! I love \LaTeX! I love \LaTeX! I love \LaTeX! I love \LaTeX!

Indenting the lines of the definition, as shown above, can help to keep track of the nesting levels; each line of the same level starts in the same column, ignoring braces.

If both the inner and outer definitions are to be provided with arguments, the symbols for the inner and outer dummy arguments must be distinguished. The symbols for the inner definition are ##1 ... ##9, while those for the outer one are the normal #1 ... #9. For example:

\begin{verbatim}
\newcommand{\thing}[1]{{\newcommand{\color}[2]{The ##1 is ##2.} \\
\color{#1}{red} \color{#1}{green} \color{#1}{blue}}}\\
The entry The colors of the objects are\ \\
\thing{dress}\ \\
\thing{book}\ \\
\thing{car} produces
\end{verbatim}

The colors of the objects are

The dress is red. The dress is green. The dress is blue.
The book is red. The book is green. The book is blue.
The car is red. The car is green. The car is blue.

The separate definition and calling as in Section 8.5.5 is easier to follow and would be:

\begin{verbatim}
\newcommand{\thing[1]}{\color{#1}{red} \color{#1}{green} \\
\color{#1}{blue} \\
\newcommand{\color}[2]{The #1 is #2.}\\
\end{verbatim}

If one were ever to go to a third level of definitions, the dummy arguments are given as ####1 ... ####9. And for the fourth level one writes 8 # signs!
Part II

Beyond the Basics
This chapter describes those features of \LaTeX{} that justify the subtitle of Leslie Lamport's original books, 'A Document Preparation System'. Whereas in the previous chapters we have concentrated more on markup, logical and typographical, we now present topics that are essential for producing large, complex documents in an efficient manner. The subjects included here are the splitting of a document into several files, selective processing of parts of a document, cross-references to sections, figures, and equations, automated production of bibliographies, indices, and glossaries.

9.1 Processing parts of a document

As often pointed out, a \LaTeX{} document consists of a preamble and the actual text part. Short documents, such as those that a beginner might have, are written to a single file by means of a text editor, and might be corrected after the first trial printing. As the user gathers experience and confidence, the \LaTeX{} documents will rapidly expand in length until one is faced with the task of producing an entire book of more than a hundred pages.

Such long documents could theoretically be kept in one file, although that would make the whole operation increasingly clumsy. The file editor functions less efficiently on longer files and the \LaTeX{} processing takes correspondingly more time. A better idea is to split the work into several files which \LaTeX{} then merges during the processing.

9.1.1 The \texttt{\input} command

The contents of another file may be read into a \LaTeX{} document with the command

\begin{verbatim}
\input{filename}
\end{verbatim}
where the name of the other file is \texttt{filename.tex}. It is only necessary to specify the full name of the file if the extension is something other than \texttt{.tex}. During the \LaTeX processing, the text contained in this second file is read in at that location in the first file where the command is given.

The result of the \texttt{\input} command is the same as if the contents of the file \texttt{filename.tex} had been typed into the document file at that position. The command may be given anywhere in the document, either in the preamble or within the text part.

Since the \texttt{\input} command may be given in the preamble, it is possible to put the whole preamble text itself into a separate file. The actual \LaTeX processing file could even be reduced to simply \texttt{\begin{document} ... \end{document}} with a number of \texttt{\input} commands. A preamble file makes sense if one has a series of documents all of the same type requiring a common preamble. This also simplifies any later change to the specifications that must be made in \textit{all} the documents. Different preamble files may be prepared for various types of processing and may then be selected with \texttt{\input{proc.type}}.

A file that is read in by means of the \texttt{\input} command may also contain further \texttt{\input} commands. The nesting depth is limited only by the capacities of the computer.

In order to obtain a listing of all extra files read in, give the command

\texttt{\listfiles}

in the preamble. The list appears both on the computer monitor and in the transcript file, at the end of the processing run. The version numbers and any other loading information are also printed. This provides a check as to which files have been input. See Section D.2.9 for a demonstration.

\textit{Exercise 9.1}: \textit{Put the preamble of your standard exercise file \texttt{exercise.tex} into a separate file \texttt{preamble.tex}. Split the text part into three files \texttt{exer1.tex}, \texttt{exer2.tex}, and \texttt{exer3.tex}. What should the main file now contain to ensure that \LaTeX processes the whole exercise text?}

\subsection*{9.1.2 The \texttt{\include} command}

Splitting the document into several files may be practical for writing and editing, but when the files are merged with the \texttt{\input} commands, it is still the entire document that is processed. Even if only one file contains a small correction, all files will be read in and processed once again. It would therefore be desirable to be able to reprocess only the one corrected file.

One rough-and-ready method is to write a temporary main file containing only the preamble (which may be read in) and an \texttt{\input} command to read in that specific file. The disadvantage is that all automatic numbering of page numbers, sections, figures, equations, etc. will start from 1, since
all the information from the previous files will be missing. Furthermore, all cross-references from other files will be absent.

A much better method is to employ the \LaTeX command

\begin{verbatim}
\include{filename}
\end{verbatim}

which is only allowed within the text part of the document, together with

\begin{verbatim}
\includeonly{file_list}
\end{verbatim}

in the preamble, containing a list file_list of those files that are to be read in. The file names are separated by commas and the extension .tex is left off.

If filename is in the file_list, or if \includeonly is missing from the preamble, the command \include{filename} is identical to

\begin{verbatim}
\clearpage \input{filename} \clearpage
\end{verbatim}

However, if filename is not contained within the file_list, \include is equivalent to \clearpage and the file contents are not read in. However, an auxiliary .aux file is read in that sets all the counters and cross-referencing information for that file.

Since \include always begins a new page, the document must be split into files at those points where a new page occurs, such as between chapters. Furthermore, \include commands may not be nested: they may only appear in the main processing file. However, an \input command may be given within a file that is \included.

The great advantage of the \include command is that the additional information about page, section, and equation numbers will be supplied by the .aux files that are read in in place of the .tex files so that the selective processing takes place with the correct values of these counters. Cross-reference information from the other files is also available so that the \ref and \pageref commands (Section 9.2.1) yield the correct results. All these values will have been determined during a previous processing of the entire document.

If the changes in the file that is being selectively processed lead to an increase or reduction in the number of pages, the following files will also have to be reprocessed to correct their page numbers. The same is true if sections are added or removed, or if the number of equations, footnotes, figures, etc. is altered.

For example, suppose file_3 ends on page 17, but after the selective processing it now extends to page 22. The following file_4 still begins on page 18, and all further files also have their original starting page numbers. If file_4 is now selectively processed, it will receive the correct number for its first page, 23, based on the stored information in the revised file_3. So far so good. However, if instead file_6 were to be selectively processed right after file_3, it would receive its starting page number from file_5, which has not yet been corrected and would be in error by 5. The same
applies to all other structure counters. Their correct values can only be
guaranteed when the files have been reprocessed in their proper order.

In spite of these restrictions, the \include command is extremely use-
ful for large documents, saving considerable computation time. Longer
documents are normally written and edited in many stages. The \include
command permits one to reprocess selective alterations in a short time,
even if the numbering systems temporarily go awry. This can be re-
paired later on with a complete reprocessing of the entire document, by
deactivating the \includeonly command in the preamble.

A file that is read in with \include may not contain any \newcounter
declarations. This is not much of a restriction, since they should normally
be given in the preamble.

For example, each chapter of a book might be written to a separate
file with names chap1.tex, chap2.tex, .... The processing file itself
contains the text

\documentclass{book}
\includeonly{...}
\begin{document}
\frontmatter
\include{toc}
\mainmatter
\include{chap1}
\include{chap2}
\include{chap3}
\backmatter
\include{back}
\end{document}

where the file toc.tex contains only commands for the title page, table
of contents, and other listings:

\begin{titlepage} ... \end{titlepage}
\tableofcontents \listoftables \listoffigures

and back.tex those for bibliography and index:

\bibliography{..}
\printindex

The chapters to be processed are selected by adding them to the argument
of \includeonly, not by changing the \include commands, which must
always be present to ensure that all the .aux are input every time. For
example, by giving \includeonly{toc,chap3} one processes the table
of contents and chapter 3.
9.1.3 Monitor input and output

There are times when it is desirable that \LaTeX write a message to the computer monitor during processing. This can be achieved with the command

\texttt{\textbackslash typeout\{message\}}

where \textit{message} stands for the text that is to appear on the monitor. This text is printed when the \LaTeX\ processing reaches this command. It is also written to the transcript file, but does not appear in the processed output.

If \textit{message} contains a user-defined command, it will be interpreted and its translation appears on the monitor. The same thing applies to \LaTeX\ commands. This could have dire consequences if the commands, either user or \LaTeX, are not really printable. To print the command name literally, precede it with the command \texttt{\textbackslash protect}.

The command

\texttt{\textbackslash typein[\textbackslash com\_name]\{message\}}

also writes the \textit{message} text to the monitor, but then it waits for the user to enter a line of text from the keyboard, terminated by typing the \texttt{\langle\textbackslash return\rangle} key. If the optional argument \texttt{\textbackslash com\_name} is missing, the line of text is inserted directly into the processing. In this way, one could, for example, reuse the same text for a letter for several addressees, entering the name each time from the keyboard. Suppose the text contains

\begin{verbatim}
Dear \texttt{\textbackslash typein\{Name\}:} \ldots
\end{verbatim}

then what appears on the monitor is:

At this point, one enters the name of the recipient. If on successive processings ‘George’, ‘Fred’, and ‘Mary’ are entered, the result will be a set of identical letters differing only in their salutations as ‘Dear George’, ‘Dear Fred’, and ‘Dear Mary’.

If the \texttt{\textbackslash typein} command contains the optional argument \texttt{\textbackslash com\_name}, this is treated the same as

\texttt{\textbackslash typeout\{message\} \newcommand\{\textbackslash com\_name\}\{entered\ definition\}}

In this case the definition is stored under the command name \texttt{\textbackslash com\_name} interactively and may be invoked and executed in the rest of the document like any other \LaTeX\ command.

With some experience in \LaTeX\ methodology, it soon becomes obvious that \textit{interactive} processing with the \texttt{\textbackslash typein} command can be very practical. For example, if the preamble contains

\begin{verbatim}
\texttt{\textbackslash typein[\textbackslash files]\{Which files?\}}
\texttt{\includeonly[\textbackslash files]}
\end{verbatim}
the following appears on the monitor:

```
Which files?
\files=
```

Now \LaTeX{} waits for the user to type in the names of one or more files (separated by commas) to be processed. This avoids having to modify the main processing file with an editor every time.

A similar procedure may be employed when a form letter is to be sent to various recipients. One can enter the name and address and even the salutation interactively. Complete forms may be processed by \LaTeX{} in this way, with the entries being made from the keyboard.

**Warning:** The `\texttt{\typein}` command may not be used as the argument of another \LaTeX{} command! It may, however, be given in environments such as `\texttt{minipage}`.

**Exercise 9.2:** Change the main file from Exercise 9.1 so that the files `exer1.tex`, `exer2.tex`, and `exer3.tex` may be read in with the `\texttt{\include}` command. Arrange that you can determine interactively which of the files is to be processed.

**Exercise 9.3:** Generate output of the form

```
Certificate
Olympic Spring Games
Walterville 1992
Finger Wrestling

Gold   A. T. Glitter  AUR  7999.9 Points
Silver  S. Lining    ARG  7777.7 Points
Bronze  H. D. Tarnish CUP  7250.0 Points
```

so that the following enquiries appear on the monitor one after the other

```
Message       Command            Input
Sport:        \@typein= Finger Wrestling
Unit:         \texttt{unit}  = Points
Gold:         \@typein= A. T. Glitter
Country:      \@typein= AUR
Value:        \@typein= 7999.9
Silver:       \@typein= S. Lining
\ldots\ldots\ldots\ldots\ldots = \ldots\ldots
```

and the appropriate entries are made interactively. The third column contains the answers necessary to produce the above sample output. Repeat the program with different entries. Let your imagination run wild.
9.2 In-text references

In a longer text, one often wants to refer to other chapters, sections, tables, and figures, or to pages where some other description has been given. One also needs an index of certain keywords that occur throughout the document. In the days before electronic text processing, such cross-references and indices meant an enormous amount of extra work for the author or secretary. Today the computer can alleviate much of this burden.

In the old days, referring to earlier parts of the text by page number was tedious but feasible. Referencing future parts that had not yet been written had to be limited to section numbers since the page numbers were not yet known, or space had to be left for the page number to be entered at a later date.

The production of a book is usually a progressive and constantly shifting process. The manuscript may not necessarily be written in the order in which it is to appear, and even once the initial draft version is finished, there will be major changes due to new considerations by the author or to knowledgeable advice from reviewers. Revisions, deletions, and insertions of entire sections or even chapters are commonplace, not to mention possible reordering of parts of the text.

\LaTeX\ relegates all the problems associated with such major alterations to the past. No matter what changes the author makes, the information necessary for the cross-references and keyword index is stored for use at any point in the text.

9.2.1 Cross-references

As already pointed out in several places, the command

\texttt{\label\{marker\}}

is used to store the current value of the relevant counter (section, equation, etc.) at that point in the text, which may then be referred to at other places. The value is assigned to the key \textit{marker}, which may be any combination of letters, numbers, and characters, even those that are normally special symbols.

The page number on which the \texttt{\label} command was issued can be printed with

\texttt{\pageref\{marker\}}

at any point earlier or later in the document.

If the \texttt{\label} command is given after a sectioning command, or within an \texttt{equation}, \texttt{eqnarray}, or \texttt{enumerate} environment, or inside the argument of a \texttt{\caption} within the \texttt{figure} or \texttt{table} environments, the command
\ref{marker}

prints the number of that section, equation, figure, table, or enumeration that was current when \textit{marker} was defined, and in the correct format. For \texttt{enumerate}, the same number is printed as is generated by the \texttt{\textbackslash item} command where the \texttt{\textbackslash label} appeared. Reference may also be made to theorem-like structures created by a \texttt{\textbackslash newtheorem} command if the \texttt{\textbackslash label} is given within the text of that theorem command. For example, a \texttt{\textbackslash label} with marker text \texttt{bo-wei} was issued in the text of the Bolzano–Weierstrass theorem on page 80:

\begin{theorem}[Bolzano--Weierstrass]
  \label{bo-wei}...
\end{theorem}

so that the input text

```
Theorem \ref{bo-wei} on page \pageref{bo-wei}
```

generates the output `Theorem 1 on page 80'. Similarly, the input text

```
for Table \ref{budget04} on page \pageref{budget04}, see also Section \ref{sec:figref}
```

produces `for Table 7.1 on page 174, see also Section 7.6', since that section contains the marker \texttt{\textbackslash label{sec:figref}}.

Note that the counter whose value is stored by the \texttt{\textbackslash label} command depends on the context in which the command is given. This is normally fairly self-evident. Within regular text, it will be the counter and its value for the last issued sectioning command, which might not be immediately visible. Therefore, when reference to a section number is desired, it is best to place the \texttt{\textbackslash label} command immediately after the appropriate sectioning command, or even within its argument, as part of the title text. This is in fact the recommended method. Put the \texttt{\textbackslash label} command within the body of the text only when you want to refer to the page number at that point.

A list of all label markers, their translations, and page numbers may be produced by processing the file \texttt{lablst.tex}, provided with the \LaTeX\ installation.

### 9.2.2 How cross-referencing works

It is useful to know how the cross-referencing information is managed. What the \texttt{\textbackslash label} command actually does is to write the key name together with the current value of the appropriate counter(s) and the current page number to an auxiliary file with the root name of the document file plus the extension \texttt{.aux}. The \texttt{\ref} and \texttt{\pageref} commands obtain their information from this \texttt{.aux} file, which is read in at the beginning of the next processing by the \texttt{\begin{document}} command. The same situation as for the table of contents (Section 3.4.2) also applies here: on the first run, the \texttt{.aux} file does not exist so that no cross-reference information
9.2. In-text references

can be output; instead the information is gathered and written to a new .aux file
at the end of the run. A warning message is printed at the end if the auxiliary
information has changed and a new run is necessary.

If selective processing is being done with \includeonly and a set of \include
commands, then an .aux file exists for each included files, as well as for the main
source file. These .aux files are read in on every processing run, whether or not
the corresponding .tex file is input. In this way, counter and cross-referencing
information for the entire document is maintained even when only parts of it are
processed.

9.2.3 Referencing parts of other documents

Package: \texttt{xr}

It is possible to refer to the marker keys in another \LaTeX{} document by
means of David Carlisle's \texttt{xr} package, which is part of the tools collection
(Section B.5.4). With

\begin{verbatim}
\externaldocument{filename}
\end{verbatim}

all the keys in the specified file become available in the current docu-
ment, to be printed with \texttt{\ref{key}} as usual. Difficulties arise if there
are duplicate key names in any of the external or current files. To cir-
cumvent this problem, an optional argument can be given, for example
as \texttt{\externaldocument[x-]{filename}}, in which case all the markers in
the external file will be prefixed with x-. Thus the external key intro is
referred to with \texttt{\ref{x-intro}}.

9.2.4 Smart referencing: the \texttt{varioref} package

Package: \texttt{varioref}

When the object being referenced is so close that you know it will be
on the same or adjacent page, you very likely will want to alter the text
accordingly. However, it might later switch from ‘same’ to ‘adjacent’ after
another revision and you will have to change that text once more. The
\texttt{varioref} package by Frank Mittelbach does this for you. It defines
a command \texttt{\vref{key}} which is almost the same as \texttt{\ref{key} on page
\pageref{key}}. However, if \texttt{\label{key}} is on the current, previous,
or next page, it prints appropriate text. Thus depending on the relative
locations, Fig. \texttt{\vref{f1}} automatically produces one of

\begin{verbatim}
Fig. 5  \hspace{1cm} \textit{when \label on same page,}
Fig. 5 on the preceding page \textit{on previous page,}
Fig. 5 on the next page \textit{on following page,}
Fig. 5 on the facing page \textit{on opposite page with twoside option,}
Fig. 5 on page 24 \textit{when two or more pages away.}
\end{verbatim}

The command \texttt{\vpageref} prints only the page part of the text, or ‘on
this page’ if \texttt{\label{key}} is on the current page. It may also take one or
two optional arguments: the texts for the current page and non-current
pages. For example, the example \texttt{\vpageref[above]{f1}} prints one of
the example above \( \text{when label on same page,} \)
the example on the previous page \( \text{on following page,} \)
the example on page 24 \( \text{when two of more pages away,} \)

while the \texttt{\vpageref[above example][example]{f1}} outputs one of:

the above example \( \text{when label on same page,} \)
the example on the next page \( \text{on the following page,} \)
the example on page 24 \( \text{when two of more pages away.} \)

A range of numbers and pages can be produced with the command \texttt{\vrefrange}, which takes two mandatory arguments, both label keys. It prints text like ‘5 to 7 on pages 212–214’, or ‘5 to 7 on the next page’, and so on. It takes one optional argument, the text to print when both labels are on the current page.

The page part of \texttt{\vrefrange} only is printed with \texttt{\vpagerefrange}, which also takes two mandatory (label key) arguments and one optional (text) one. If both labels are on the same page, it behaves like \texttt{\vpageref}; the optional argument is printed if both are on the current page.

Since English text is automatically printed by the above commands, they would not be very useful for documents, say, in Italian. The varioref package therefore accepts all the language options recognized by the babel system (Chapter 11) to reprogram the texts. With the option \texttt{italian}, one will get texts like \textit{in questa pagina} for ‘on this page’ and \textit{nella pagina precedente} for ‘on the preceding page’, and so on.

All the texts are stored in special commands which can be redefined by the user:

\begin{verbatim}
\reftextbefore \text{previous but non-facing page}
\reftextfacebefore \text{previous but facing page}
\reftextafter \text{next but non-facing page}
\reftextfaceafter \text{next but facing page}
\reftextfaraway\{key\} \text{more than one page away}
\reftextpagerange\{key1\}\{key2\} \text{page range of 2 labels}
\reftextlabeledrange\{key1\}\{key2\} \text{range of 2 different labels.}
\end{verbatim}

The last three must be redefined with 1 or 2 arguments, as

\begin{verbatim}
\renewcommand{\reftextfaraway}[1]\{on page \pageref{#1}\}
\renewcommand{\reftextpagerange}[2]\{on pages \pageref{#1} -- \pageref{#2}\}
\renewcommand{\reftextlabeledrange}[2]\{\ref{#1} to \ref{#2}\}
\end{verbatim}

There is also \texttt{\vreftextvario\{text1\}\{text2\}} to provide altering texts for the above commands, so that the same text is not printed every time.

\section{Bibliographies}

Academic publications normally include a list of references, or bibliography, containing the names of other works that are cited within the text
by means of a running number. This is the system supported by standard \TeX, described in the next section. The alternative citation method, with author name and year of publication, can be achieved with various extension packages, as described in Section 9.3.4).

Often the bibliography has not been finalized before the main text is started. Especially for numerical bibliographies, it would be a great nuisance to have to go through the whole text and change all the numbering every time something was added to the bibliography. Therefore \TeX is programmed not only to format the bibliography but also to keep track of its alterations and additions in order to modify the references in the text automatically.

The bibliography is generated most conveniently and efficiently from a general database with the help of the \TeX program that should be part of the \TeX installation. How to prepare such databases is explained in Chapter 14. The advantages of a database is that its entries can be reused again and again for all future documents, while the formatting style in each bibliography listing can vary according to the requirements of each document. One can, however, write it by hand without \TeX.

\section*{9.3.1 Bibliography by hand}

The actual bibliography, whether created by \TeX or by hand, is placed inside the environment

\begin{verbatim}
\begin{thebibliography}{sample_label}
  entries
\end{thebibliography}
\end{verbatim}

The individual entries in the bibliography each begin with the command

\begin{verbatim}
\bibitem[label]{key} entry_text
\end{verbatim}

Without the optional argument \textit{label}, \texttt{\bibitem} produces a running number in square brackets as the label for the reference in the text. With \textit{label}, one can give whatever indicator one wishes, such as an abbreviation of the author's name, or an arbitrary reference number. The mandatory argument \textit{key} is a reference key, much like that for the cross-referencing \texttt{\label} command (page 213), which is used to make the actual citation in the main text. The key name can be made up of any combination of letters, numbers, and symbols except commas.

The bibliography information itself is contained in \textit{entry_text}, such as ‘author, title, publisher, year, edition, page numbers’, possibly in various typefaces. The output text will be indented after the first line by a width equal to that of the \texttt{sample_label}, so this should be as large as the longest label in the bibliography. For the standard application with running numbers, \texttt{sample_label} should be a dummy number with as many digits
Chapter 9. Document Management

as the largest label (for example 99 if there are more than 10 but less than 100 entries).

For bibliographies with author–year citations (Section 9.3.4) the entries in the thebibliography are the same, except that the optional label must be present, taking a special form that will transfer the author and year texts to the citation commands.

A sample (numerical) thebibliography environment could look as follows:

\begin{thebibliography}{99}
\bibitem[6a]{knuth:a} Vol A: \textit{The \TeX{}book}, 1986
\end{thebibliography}

Here lamport, knuth, and knuth:a have been chosen as keys. The sample label is given as 99 since a two-digit number produces sufficient indentation for the standard form of \bibitem. The entry with the key knuth is the sixth in the list and thus it automatically receives the label [6]; in order for its sub-entries knuth:a ... knuth:e to be printed as [6a] ... [6e], it is necessary to set their optional label arguments to 6a ... 6e. (Yes, this does violate the automatic labeling system.)

The bibliography is normally printed near the end of the document by placing the thebibliography environment at the desired location. For document classes book and report, the word Bibliography appears as an unnumbered chapter title at the beginning, whereas for article the word References is written as an unnumbered section heading. The above sample bibliography appears as:

[6a] Vol A: \textit{The \TeX{}book}, 1986

Exercise 9.4: Produce a bibliography with the thebibliography environment, with a label consisting of the first three letters of the author’s name followed
by the last digits of the year of publication. If the same author has more than 
one work in a given year, add a running letter to distinguish them, for example 
knu86c, knu86d. With such labels, it is appropriate to make the reference keys 
the same as the labels. The indentation depth should be 1.5 cm. 
Note: The indentation depth is determined by the argument sample.label in the 
thebibliography environment. This is usually a dummy text with the right 
width. This width can be given precisely with \hspace{width}.

Exercise 9.5: Copy the thebibliography environment from the above exercise 
to the end of your standard exercise file (but ahead of the \end{document} 
command). Refer to the entries in the bibliography by inserting \cite commands 
into your text. Make sure that the key in the \cite command is written exactly 
as in the \bibitem command in the thebibliography environment.

9.3.2 Bibliography with Bib\TeX

Bibliographic databases and the Bib\TeX program are fully described in 
Chapter 14. Here we just review the essential aspects for the \LaTeX source 
file.

- All the information for the literature references is placed in one or 
  more databases. Each entry is given an identifying key (the key in 
  the \bibitem argument) which is the handle to refer to it.

- In the source text, citations to the references are made with various 
  commands like \cite, \citet, \citep, taking the key as argument.

- The bibliography style must be specified somewhere in the document 
  with \bibliographystyle{bib_style}; the bib_style determines how 
  the bibliography is to be formatted; the basic style is plain for 
  numerical citations, or plainnat with the natbib package.

- Give the command \bibliography{database} where the bibliogra- 
  phy is to appear; database is a list of all databases to be searched.

- Process the source file with \LaTeX; run Bib\TeX and then \LaTeX twice.

Bib\TeX writes a bibliography following the formatting specifications of 
bib_style into a thebibliography environment, storing it in a file with 
extension .bbl; this file is input by the \bibliography command.

9.3.3 Numerical citations

Standard \LaTeX can only handle numerical bibliographies and citations. In 
this case, the citation in the text is made with the command

\cite[extra]{key}
where \textit{key} is the reference key that appears in the \texttt{\bibitem} command. With the sample bibliography on page 218, the source text

\begin{verbatim}
For additional information about \LaTeX\ and \TeX\ see \cite{lamport} and \cite{knuth,knuth:a}.
\end{verbatim}

yields: For additional information about \LaTeX\ and \TeX\ see [1] and [6, 6a].

If the optional argument \textit{extra} is included in the \texttt{\cite} command, this text is added after the label(s) but still inside the square brackets.

The creation of a bibliographic database is described in \cite[Appendix B]{lamport}, while the program \texttt{\BibTeX}\ itself is explained in \cite[pages 74, 75]{lamport}.

The creation of a bibliographic database is described in [1, Appendix B], while the program \texttt{\BibTeX}\ itself is explained in [1, pages 74, 75].

\subsection*{9.3.4 Author–year bibliographies with \texttt{natbib}}

An alternative bibliographic citation style, used in this book and in many scientific journals, makes reference to other published works by citing the author's name and year of publication. In this case, the entries in the bibliographic listing are not numbered. The citation itself may be either parenthetical [\textit{Jones et al.}, 1999] or textual, as shown by \textit{Jones et al.} [1999]. There may be some variations within this style: for example, in this book parentheses are used in place of square brackets and the name is left in normal type. None of this is supported by standard \LaTeX, but rather by additional packages.

There are a number of packages developed to deal with this situation, each requiring a different syntax for the optional \textit{label} argument in the \texttt{\bibitem} command in order to transfer the author and year information to the citation commands. And each has its own bibliography style (.bst) file to allow \texttt{\BibTeX} to accomplish this.

The \texttt{natbib} package by Patrick W. Daly is the most universal of these, being compatible with the \texttt{\bibitem} syntaxes, and thus with the .bst files, of other author–year packages such as \texttt{apalike}, \texttt{chicago}, and \texttt{harvard}, to name just a few. Furthermore, \texttt{natbib} can even produce numerical or superscript citations as well, and will also function with the standard (numerical) bibliography styles and \texttt{\bibitem} syntax, albeit only with numerical citations.

Furthermore, \texttt{natbib} is integrated with the \texttt{hyperref} package (Section 10.2.4) to make citations into automatic links to the bibliographic entry.

\textbf{The author–year \texttt{\bibitem}}

The required \texttt{natbib} form of the 'optional' \textit{label} argument to \texttt{\bibitem} within the \texttt{thebibliography} environment is
\bibitem {short(year)long} {key} ...

where \textit{short} is the short author list (e.g. Jones \textit{et al.}) and \textit{long} is an optional full author list. A sample bibliography might be:

\begin{thebibliography}{99}
\bibitem[James et al.(2001)James, Jones, and Smith]{JAM01}
P. R. James, F. J. Jones, and R. T. Smith, ... 2001.
\bibitem[Jones et al.(1999)Jones, Baker, and Toms]{JON99}
F. J. Jones, H. P. Baker, and W. V. Toms, ... 1999.
\bibitem[Jones et al.(2000a)Jones, Toms, and Baker]{JON00}
F. J. Jones, W. V. Toms, and H. P. Baker, ... 2000a.
\bibitem[Jones et al.(2000b)Jones, Toms, and Baker]{JON00b}
F. J. Jones, W. V. Toms, and H. P. Baker, ... 2000b
\end{thebibliography}

Such bibliographies will be automatically produced by \LaTeX{} when a bibliography style is invoked with the \texttt{\bibliographystyle} command that is compatible with natbib. Three such .bst files are provided with natbib, as substitutes for 3 of the 4 basic \LaTeX{} styles: plainnat, unsrtnat, and abbrvnat (Section 14.1). However, many others also exist as contributions for specific publishers and journals.

\textbf{Author-year citations}

There is a flexible citation syntax permitting parenthetical (\texttt{\citep}) and textual (\texttt{\citet}) citations, with the abbreviated or full author list, with optional notes both before and after the citations. For example,

\begin{verbatim}
\citep{JON99} \Rightarrow Jones et al., (1999)
\citep[pg.\textsuperscript{22}]{JON99} \Rightarrow Jones et al., (1999, pg. 22)
\citet{JON99} \Rightarrow (Jones et al., 1999)
\citet[pg.\textsuperscript{22}]{JON99} \Rightarrow (Jones et al., 1999, pg. 22)
\citet[e.g.]{JON99} \Rightarrow (e.g. Jones et al., 1999)
\citet*[pg.\textsuperscript{22}]{JON99} \Rightarrow (e.g. Jones et al., 1999, pg. 22)
\citet*{JON99} \Rightarrow Jones, Baker, and Toms (1999)
\citet*{JON00} \Rightarrow Jones, Toms, and Baker (2000)
\end{verbatim}

The citation punctuation (type of brackets, points between multiple citations, dates, and so on) can be selected by options when loaded, specified by declarations, or programmed to be associated with the bibliography style. See below.

Multiple citations are available as usual, with compression of repeated authors:

\begin{verbatim}
\citep{JON99,JAM01} \Rightarrow Jones et al. (1999); James et al. (2001)
\citet{JON99,JAM01} \Rightarrow (Jones et al., 1999; James et al., 2001)
\citet{JON99,JON00} \Rightarrow (Jones et al., 1999, 2000a)
\citet{JON00,JON00b} \Rightarrow (Jones et al., 2000a,b)
\end{verbatim}
In numerical citation mode, the command \citep behaves like the standard \cite command, printing the reference's number in brackets, while \citet gives the authors' names followed by the reference's number. This is textual citation in numerical mode, allowing the author to use the same syntax for both citation modes.

\citep{JON99} ⇒ Jones et al. [21]
\citet{JON99} ⇒ Jones et al. [21, pg. 22]
\citep{JON99} ⇒ [21]
\citet{JON99} ⇒ [21, pg. 22]
\citep[e.g.]{JON99} ⇒ [e.g. 21]
\citet[e.g.]{JON99} ⇒ [e.g. 21, pg. 22]
\citep{JON00,JON00b} ⇒ [21, 32]
\citet{JON00,JON00b} ⇒ [21, 32]

The commands \citeauthor{key} and \citeyear{key} print just the author names and year, respectively; a starred version \citeauthor prints the full author list.

There also exist capitalized variants of these citation commands, \Citet, etc., for use at the beginning of a sentence, when the first author name begins with something like 'von'.

\citet{delaM89} has shown ... ⇒ de la Mar has shown ...
\Citet{delaM89} has shown ... ⇒ De la Mar has shown ...

Package options

Many aspects of formatting the citations can be controlled by options specified when loading \texttt{natbib} with the \texttt{usepackage} command:

round citations in round parentheses (default);
square citations in square brackets;
curly citations in curly braces;
angle citations in angle brackets;
colon to separate multiple citations with semicolons (default);
comma to use commas as separators;
authoryear for author–year citations (default);
numbers for numerical citations;
super for superscripted numerical citations;
sectionbib
   puts the bibliography in a section instead a chapter; for use with the chapterbib package (Section 9.3.5);

sort
   orders multiple citations into the sequence in which they appear in the list of references;

sort&compress
   as sort but in addition multiple numerical citations are compressed if possible (as 3–6, 15);

longnamesfirst
   makes the first citation of any reference the equivalent of the starred variant (full author list) and subsequent citations normal (abbreviated list).

Note: natbib defaults to author–year citations, but if any single \bibitem command does not conform to any of the recognized author–year syntaxes, it will switch automatically to numerical citations.

Further customizations

There are several formatting commands in natbib that may be (re)defined by the author to achieve special effects in the bibliography, such as font or spacing between entries. In standard \LaTeX, these are all fixed.

\bibsection
   can be redefined to the desired sectioning command for introducing the list of references. This is normally \section* or \chapter*.

\bibpreamble
   is printed after the bibliography heading but before the actual list of references; is normally undefined, but can be defined to insert any desired leading text.

\bibfont
   can be a font declaration, e.g. \small to apply to the whole list of references; is normally undefined.

\citenumfont
   can be a font declaration or command like \itshape or \textit for formatting numerical citations; normally undefined.

\bibnumfmt
   is a command with an argument to format the numbers in the list of references; the default definition is [#1].
\bibhang is the indentation after the first line of each reference; change this with the \setlength command.

\bibsep is the vertical spacing between references; change this with the \setlength command.

There are many other customization possibilities and specialized features to \natbib which can be found in the supplied documentation (process \natbib.dtx if it not already provided as a .dvi or .pdf file) or in the short summary \natnotes.tex.

9.3.5 Multiple bibliographies

Package: chapterbib

In standard \LaTeX, with or without \BibTeX, there can only be one bibliography per document. However, it is not uncommon for a book to consist of independent chapters, each with its own bibliography and local citations.

The \chapterbib package by Donald Arseneau solves this problem. It requires that the document be broken up into individual source files that are read into the main file with the \include command of Section 9.1.2. Each such file must contain its own thebibliography environment, or its own \bibliographystyle and \bibliography commands if \BibTeX is to be used. The citation commands will work locally within each file. Even if the same reference appears in several bibliographies, with a different number each time, the citations will refer to the one in the local file.

One must process \BibTeX individually on each of the separate files, creating a series of .bbl files. Then the document is reprocessed with \LaTeX, at least twice; the .bbl files are read in by the individual \bibliography commands.

If you cannot make use of the \include commands because you do not want each file to begin on a new page, you can still localize the bibliographies and their citations with the cbunit environment. This does not work with \BibTeX, however. In that case, you must still put the bibliographic units into separate files, and then read them in with the \cbinput{file} command. Set the package option draft when loading \chapterbib, process with \LaTeX, process each file with \BibTeX, remove the draft option, process with \LaTeX at least twice.

With classes book and report, the bibliography is placed in an unnumbered chapter; this is undesirable if each chapter is to have its own bibliography. In this case, add the package option sectionbib to change the bibliography to an unnumbered section instead.

The \chapterbib package is compatible with \natbib except for the sectionbib option. The two packages clash on their redefinitions of bibliography. In this case, add the option to \natbib instead of to \chapterbib.
9.4 Keyword index

\LaTeX itself does not produce a keyword index automatically as it does a table of contents, but it can prepare input data for the MakeIndex program (Section 9.4.3) that does generate such an index, in a form that \LaTeX can use on a later run.

9.4.1 The index environment

A keyword index is formatted in the environment

\begin{theindex} \index{index_entry} \end{theindex}

which switches to a two-column page format with a running head INDEX. The first page of the index carries the heading Index in the same size as a chapter heading for the book and report document classes, and as a section heading for the article class. (More precisely, the actual word printed is contained in the command \indexname which may be redefined for other languages.) The individual entries are made with commands

\item \subitem \subsubitem and \indexspace

followed by the keywords and their page numbers. For example,

\begin{itemize}
  \item commands, 18
  \item arguments, 19, 101
  \item multiple, 103, 104
  \item replacement symbol, 20
  \item as environments, 42
  \item used as arguments for sectioning commands, 41, 42
\end{itemize}

displayed text, 21–32

If the text entry is too long for one line, it is broken and continued on the next line, indented deeper than all other lines, as in the above example ‘used as argument for sectioning commands, 41, 42’. The command \indexspace leaves a blank line in the index.

9.4.2 Preparing the index entries

The theindex environment only sets up a suitable format for the index. The entries themselves, as well as their page numbers, are generated by the MakeIndex program described in Section 9.4.3. This program requires input information from the \LaTeX file in the form of unsorted keywords and page numbers. The author enters this information in the text file with the command

\index{index_entry}
where \texttt{index\_entry} is the text to be entered into the index. It may contain any combination of letters, numbers, and symbols \textit{including even command characters and blanks}. This means that even commands may be included in the \texttt{index\_entry}, such as \	exttt{\LaTeX\ \logo}. Even the one command that may otherwise never be used as an argument, \texttt{\verb}, may be included. However, if \texttt{index\_entry} does contain a command, \texttt{\index} may not be used as an argument of another command.

Normally all the \texttt{\index} commands are ignored by \LaTeX and do absolutely nothing. They are activated only when the preamble contains the command

\texttt{\makeindex}

in which case, a file with the document's root name plus the extension \texttt{.idx} is opened. Now the \texttt{\index} commands write \texttt{index\_entry} and the current page number to this file in the form

\texttt{\indexentry{index\_entry}{pagenumber}}

It is this \texttt{.idx} file that is the input to the MakeIndex program. There are special symbols that may be used in the \texttt{index\_entry} to indicate main and subitems and other refinements.

The MakeIndex program expects the \texttt{\index} entries to be of one of the three forms:

\texttt{\index{main\_entry}}
\texttt{\index{main\_entry!sub\_entry}}
\texttt{\index{main\_entry!sub\_entry!sub\_sub\_entry}}

The individual main and sub-entries may contain any combination of characters with the exceptions of \texttt{!, @, and \textbar}. The exclamation point is the divider between the various entry fields. If the \texttt{\index} command contains only a main entry, this will become the text for an \texttt{\item} command. The main entries will be in alphabetical order.

If the \texttt{\index} command contains a main and a sub-entry, the text \texttt{sub\_entry} will be assigned to a \texttt{\subitem} command underneath the corresponding \texttt{main\_entry}. The texts for the \texttt{\subitem} will also be in alphabetical order. Similarly the text of a \texttt{sub\_sub\_entry} will appear following a \texttt{\subsubitem} command, alphabetically ordered, below the appropriate \texttt{sub\_entry} text.

The main and sub-entries may also contain special characters and even \LaTeX commands that are to be neglected during alphabetization. This is indicated by an entry of the form \texttt{lex\_entry@print\_entry}, in which \texttt{lex\_entry} is used for the alphabetical ordering while \texttt{print\_entry} is the actual text to be output. For example, in the keyword index in this book, which was generated with MakeIndex, all the command names appear as input text and are ordered without the preceding \texttt{\ character}. These entries were made in the text as \	exttt{\index{put@\verb=\put=}}. 


An entry may be terminated with the character sequences |( or |}) to designate the beginning and end of a range of page numbers. For example,

```
\index{table|(} on page 95
\index{table|)} on page 110
```

produces the entry 'table' with page numbers given as 95-110.

Instead of having the page number printed after the entry, a reference may be made to another index entry. For example, with

```
\index{space|see{blank}}
\index{table|seealso{array}}
```

the entries 'space, see blank' and 'table, see also array' are made in the index.

The three characters !, @, and | therefore have special functions for the MakeIndex program. In order to print these characters literally as text without their functions, the quote character " must precede them. For example, "! represents a literal exclamation point and not the entry divider.

The quote character itself is thus a fourth special symbol and must be entered literally as "". However, there is a special rule in MakeIndex syntax that says a quote character preceded by a backslash will be interpreted as part of a command: thus \" may be used to put a German umlaut in an entry (as in \index{Knappen, J\"org}). This special rule can lead to additional problems at times: in this book the index entry \! had to be typed as "\!".

It is possible to specify varying fonts for the page number. For example, in the index of this book, page numbers are set in bold face to indicate the page where a command is defined or first explained. This is achieved with an entry of the form

```
\index{blank} on page 11, and
\index{blank|textbf} on page 22
```

which in the second case puts the page number for this entry as the argument of a command \textbf. The line in the theindex environment becomes

```
\item blank, 11, \textbf{22}
```

**Note:** The vertical bar in the \index entry is not a typing error, but must replace the backslash as the \LaTeX command symbol under these circumstances.

There is a file idx.tex in the \LaTeX installation, which can improve the readability of the .idx files. By processing the idx.tex file (that is, calling latex idx), the user is prompted for the name of the .idx file to list:
After the root name of the .idx file has been entered from the keyboard, a two-column page format is output containing the index entry texts listed under page headings of the form Page n. No further information is obtained through such a formatted listing, but it is far easier to work with than a straightforward output of the .idx file alone.

Even if the \index commands have no effect without the \makeindex command in the preamble, it is a good idea to include them in the text from the initial development of the document. The \makeindex command can be included later when the final version of the document is ready. At this point, the MakeIndex program can be run to produce the theindex environment from the data in the .idx file just as B\TeX produces the bibliography environments. This tool is described in the next section.

The L\TeX installation also contains the package showidx, which prints the index entries from the \index commands as marginal notes, beginning at the top of the page on which they occur. This is useful for going through a preliminary version of the document to check that all index entries are really on the proper page or whether additional entries have to be made.

If showidx is used, it is a good idea to increase the width of the marginal note box with the declaration \marginparwidth (Section 4.10.6) in the preamble. Unfortunately, the present book format is not suitable for such a demonstration.

### 9.4.3 Running MakeIndex

The tiresome drudgery of making up the theindex environment for a keyword index can be dispensed with if the program MakeIndex is available. This program was written by Pehong Chen with the support of Leslie Lamport. We will give only an abbreviated description of its use here. More details are in the documentation accompanying the program package.

The program MakeIndex processes the .idx file, producing as output another file with the root name of the document and the extension .ind, containing the complete theindex environment. It is run by clicking the appropriate icon in the editor shell, or from a command line with

```
makeindex root.name.idx
```

A subsequent L\TeX processing outputs the index at the location of the \printindex command which, together with the \see command, is defined in the package makeidx. Thus the production of an index in this way requires the package file makeidx to be loaded with the \usepackage command.

The alphabetical ordering of MakeIndex normally follows the standard ASCII code, first symbols, then digits, and finally letters, where upper case comes before lower case. Blanks are included as symbols. There are a
number of options that allow these rules to be changed. How the options are invoked when the program is called depends on the computer type, but we assume here a hyphen preceding the option letter, as in

```
makeindex -g -l root_name
```

The most important options are:

- `-l` Letter ordering: blanks are ignored when sorting;
- `-c` Compress blanks: multiple and leading blanks are removed as they are in normal \LaTeX;
- `-g` German ordering: following the German rules in which symbols precede letters (lower before upper case) which precede numbers; the sequences "a," "o," "u," and "s" (the codes for \¨a, \¨o, \¨u, and \ss in the usual \LaTeX adaptations for German) are treated as though they were ae, oe, ue, and ss, which is standard German practice;
- `-s` Style specification: allows the name of an index formatting file to be included to redefine the functioning of MakeIndex.

The `-s` option reads in an index style file containing commands to define both the input and output of the MakeIndex program. For example, it is possible to change the special symbols !, @, |, and " so that different characters execute their functions and they themselves revert to being pure text.

The style-defining file consists of a list of pairs in the form `keyword attribute`. The attribute is either one character in single quotes (for example, `'z'`), or a character string in double quotes (for example, "a_string"). The most important keywords, together with their default definitions, are:

- `quote` `'` defines the quote symbol;
- `level` `'!'` defines the entry separation symbol;
- `actual` `'@'` defines the lexical switch symbol;
- `encap` `'|'` defines the dummy command symbol for page formatting.

There are many more keywords for defining complicated output structures. These are described in the documentation that should be included with the MakeIndex program package.

The file `makeindex.tex` contains a short manual by Leslie Lamport (but without mentioning style definitions).
9.4.4 Glossary

A 'glossary' is a special index of terms and phrases alphabetically ordered together with their explanations. To help set up a glossary, LaTeX offers the commands

\makeglossary \hspace{1cm} in the preamble and
\glossary{glossary_entry} \hspace{1cm} in the text part

which function just like the commands for making up a keyword index. The entries are written to a file with extension .glo after the command \makeglossary has been given in the preamble. The form of these file entries from each \glossary command is

\glossaryentry{glossary_entry}{pagenumber}

The information the .glo file can be used to establish a glossary. However, there is no equivalent to the theindex environment for a glossary, but a recommended structure is the description environment (Section 4.3.3) or a special list environment (Section 4.4).
PostScript and PDF

PostScript is a printing and plotting language developed by the Adobe Systems Inc. Unlike the printing instructions for other types of printers, the PostScript code may be written to an ASCII file, viewed on a computer monitor, put online, downloaded, included in emails. It therefore represents a type of electronic paper.

It is also a programming language, and thus offers enormous possibilities for including special effects that cannot easily be inserted with other output formats. Several packages exist to exploit these, packages which are then limited to PostScript output. However, the PostScript advantages are so attractive that this can be justified.

Portable Document Format, PDF, is another Adobe product, a successor to PostScript, inheriting many of its properties. However, PDF includes many features to make it applicable to true electronic documents: internal and external links, bookmarks, animation, encryption, etc., features that have no correspondence in paper. It represents a revolutionary new type of information medium.

PostScript has played a very important role in the development of \LaTeX, setting the standards for graphics inclusion, and offering additional font families over the standard Computer Modern set. Now PDF carries on that tradition. With the pdf\LaTeX program, PDF is actually married to \TeX and thus to \LaTeX.

In the next sections we describe the generation of PostScript output with \LaTeX, the usage of PostScript (type 1) fonts, and then the production of PDF output with all its bells and whistles.

10.1 \LaTeX and PostScript

PostScript output is best produced with the dvips driver program written by Tomas Rokicki, available with most \TeX installations. It is this very powerful and flexible program that has set the standard for DVI conversion programs.
Its two most significant features are the inclusion of graphics as *encapsulated* PostScript images, as described in Chapter 6, and the capability of making use of PostScript fonts in place of the Computer Modern ones (Section G.5). There are also enhanced drawing capabilities, as demonstrated in Sections 13.2.2 and 13.3.2.

### 10.1.1 The dvips driver

The `dvips` driver program operates on the DVI result from \TeX, with or without \LaTeX, to generate PostScript output, which may be written to a file or sent directly to a printer. One either clicks the appropriate icon in the editor shell, or issues the command from a command line prompt:

\[ \texttt{dvips myfile} \]

This command will process `myfile.dvi` and generate `myfile.ps`. (The \LaTeX source text was of course in `myfile.tex`.) The PostScript file may be viewed with a PostScript viewer like GhostView or sent to a PostScript printer, or included in an email.

There are many options that may be included in the command line, appearing between `dvips` and the file name. For example, to output five pages only starting at the 10th, sending them to a file named `p5-15.ps`, give

\[ \texttt{dvips -p 10 -n 5 -o p5-15 myfile} \]

You may also specify the first (\(-p\)) and last (\(-l\)) page numbers, whether only odd (\(-A\)) or even (\(-B\)) pages are to be printed, or in reverse order (\(-r\)). The page numbers with -p and -l refer to the actual page numbers in the document, or rather to its first occurrence. Hence -p 8 might start with page `viii` rather than page 8. Specifying -p =8 will start with the 8th page, regardless of its actual number. The list of options can be printed to the monitor with `dvips --help`.

One useful option is \texttt{t landscape} to rotate the output into landscape mode, which should be done if the document has been processed with the \texttt{landscape} option in \texttt{\documentclass}. However, `dvips` has another way of accomplishing this: add the driver-specific command

\[ \texttt{\special{landscape}} \]

in the source text itself and `dvips` will automatically print in landscape mode.

The output is written to a file with the same root name as the input, but with extension `.ps`, unless the option `-o filename` is given. To send the output directly to a printer, specify that printer as the output `file`, as `-o \lpt1`. Beware, if that printer is not a PostScript printer, it will simply output the PostScript code directly as text, something that normally uses a considerable amount of paper.
A full manual for dvips should be available as dvips.ps or dvips.pdf; otherwise the source file dvips.tex can be processed (with \TeX, not \LaTeX!).

The program is configured to the local installation by means of a file config.ps that should be located in texmf\dvips\config. This sets up certain parameters for the default printer, as well as various paper formats, indicating which is to be the default (European A4 or American letter). This can be overridden with -t a4 or -t letter. Another important configuration item is a list of font mapping files (Section 10.1.3), to associate \TeX font names with actual PostScript fonts and their files.

Additional configuration files are usually provided, for specific printers or other font mappings, all with the name config plus a different extension. They may be invoked on the command line with the option -P plus extension. For example, config.cx contains lines for a generic PostScript printer with resolution of 300 dpi; with -P cx, this file is input, overriding the default values in config.ps. Another example for adding PostScript fonts is given in Section 10.1.3.

Of course, the dvips driver manages not only PostScript fonts but also the standard \TeX pixel fonts generated by the METAFONT program (Section G.3). These bitmaps are stored in .pk files that need to be generated for the desired printer, size, and resolution. Originally, the user had to ensure that they were present before the driver program could address them. Fortunately, dvips has revolutionized this process by calling METAFONT itself if the needed .pk files are missing.

However, it is the use of PostScript fonts that make dvips especially interesting. We describe in some detail how they are integrated into the \TeX/\LaTeX system in the next sections since almost all of this applies equally well to PDF output.

### 10.1.2 Invoking PostScript fonts

Under the New Font Selection Scheme of Appendix A, it is relatively simple to invoke the PostScript fonts, provided that the font definition .fd files are provided. These are normally to be found in the directory psnfss in the standard installations. It also contains packages for the 35 standard PostScript fonts, plus those for a number of other commercially available fonts. These packages are actually quite simple; for example, the times.sty file contains only four lines, which are listed on page 379. All these packages do is redefine the three font families, \texttt{\rmdefault}, \texttt{\sfdefault}, \texttt{\ttdefault} and the bold face default attribute \texttt{\bfdefault}.

The packages for the 35 standard fonts, and the three assignments that they make, are listed in Table 10.1 on the next page. (The 35 fonts include italic and bold variants of these.)

These packages are only meaningful when the resulting DVI output is to be transformed with dvips to PostScript or PDF files, or when the \LaTeX source is to be processed with pdf\LaTeX to generate PDF directly. Other
Table 10.1: The psnfss packages and their fonts

<table>
<thead>
<tr>
<th>Package</th>
<th>\texttt{\textsf{rmfamily}}</th>
<th>\texttt{\textsf{sffamily}}</th>
<th>\texttt{\textsf{ttfamily}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>times.sty</td>
<td>Times-Roman</td>
<td>Helvetica</td>
<td>Courier</td>
</tr>
<tr>
<td>palatino.sty</td>
<td>Palatino</td>
<td>Helvetica</td>
<td>Courier</td>
</tr>
<tr>
<td>newcent.sty</td>
<td>NewCenturySchlbk</td>
<td>AvantGarde</td>
<td>Courier</td>
</tr>
<tr>
<td>bookman.sty</td>
<td>Bookman</td>
<td>AvantGarde</td>
<td>Courier</td>
</tr>
<tr>
<td>avant.sty</td>
<td></td>
<td>AvantGarde</td>
<td></td>
</tr>
<tr>
<td>helvet.sty</td>
<td></td>
<td>Helvetica</td>
<td></td>
</tr>
</tbody>
</table>

drivers can only handle the $\TeX$ pixel fonts described in Appendix G (also known as type 3 fonts). Some drivers, in particular the monitor previewers $\texttt{xdvi}$ and $\texttt{windvi}$, will also invoke programs to generate the bitmap .pk files from the PostScript .pfb source files.

Why bother to use these packages when $\texttt{dvips}$ is capable of including the pixel fonts just as well? The answer is that the standard PostScript fonts are well known from other applications, and there are additional commercially available fonts in PostScript format; they therefore offer a wider choice than simply the traditional Computer Modern fonts that are limited to $\TeX$ and $\LaTeX$.

Furthermore, although the pixel fonts can be included in PDF output, the results viewed on the monitor are simply atrocious. They are slow to draw and appear very fuzzy. (They do print all right, however.) PDF should make use of PostScript fonts exclusively. In Section G.6 we explain how the standard Computer Modern and Extended Modern fonts are now available in PostScript encoding, especially for this need.

10.1.3 Installing PostScript fonts

The packages listed in the previous section invoke the PostScript fonts by means of the NFSS system, provided they have been properly included in the installation. This means that the following files must be present:

- .tfm   the font metric files for both the virtual and raw fonts;
- .fd    the NFSS font definition files associating the font attributes to precise virtual font names; these and the .tfm files are the only ones read by $\LaTeX$ itself;
- .vf    virtual font files that instruct the DVI driver how to produce the characters; usually this refers to characters in a real font with different encoding;
- .map   mapping files that tell the $\texttt{dvips}$ driver the true internal names of the real fonts, plus any re-encodings or distortions that must be undertaken.
The .map files are a feature of dvips that has been taken over by pdflatex and other programs that make use of PostScript fonts. It translates the \LaTeX{} names for the fonts into the internal names used by PostScript itself. For example, the line

\verbatim{ptmr8r Times-Roman "TeXBase1Encoding ReEncodeFont" <8r.enc}

says that the font known to \LaTeX{} as ptm8r is really the Times-Roman font. Furthermore, the text in quotes is also inserted into the PostScript output; in this case, it is a command to state that the symbol assignments (the encoding) should be that of the TeXBase1Encoding vector. The definition of this encoding vector is to be found in the file 8r.enc, which must also be inserted, since it is not part of basic PostScript.

Note that type 1 fonts are essentially a collection of symbols identified by name, whereas application programs like \LaTeX{} address characters by positional number. It is the encoding vector that associates those numbers with symbol names for the benefit of the application. However, they are not intrinsic to the font itself.

For fonts other than the 35 standard ones included in every PostScript output device, additional files must also be input. For example

\verbatim{hlhr8r LucidaBright "TeXBase1Encoding ReEncodeFont" <8r.enc <hlhr8a.pfb}

says that the file hlhr8a.pfb must also be inserted into the PostScript output, containing the drawing instructions for the LucidaBright font.

The .map files are listed in the config.ps file that configures dvips. Often there is a single mapping file named psfonts.map, included with the line

\verbatim{p psfonts.map}

in config.ps, or individual mapping files are listed, as

\verbatim{p psnfss.map}
\verbatim{p +pazo.map}
\verbatim{p + ...}

prefixed with a + sign after the first one.

In some cases, the map file listings are also included in additional config files, such as config.cms for the PostScript versions of the standard Computer Modern fonts. If these are not already contained in config.ps or in psfonts.map, they may be added on the dvips command line with the option -P cms. Other families of PostScript fonts can be included in the same manner.

### 10.1.4 Long road from input code to output character

Let us illustrate with an example how \LaTeX{} translates an input character with given font attributes into a particular symbol in a specific font. Suppose we have the following attributes (Section A.1):

\verbatim{Encoding: OT1  Family: ptm  Series: b  Shape: s1}
10.2 Portable Document Format (PDF)

In 1993, Adobe Systems Inc. published the *Portable Document Format*, or PDF, as a page-oriented format for electronic documents. As a successor to PostScript, which is also an Adobe creation, it combines fixed textual contents with hypertext elements, compression methods, and security measures. It corresponds to true electronic publishing with a fixed layout but still with hardcopy possibilities. Compared to PostScript files, the PDF output is considerably smaller, especially when graphics are included. Today it is considered the ideal medium for storage and transmitting large documents electronically in final format.

10.2.1 Producing PDF output

Adobe has made the PDF specification public and freely available so that anyone is entitled to write software to produce and to read files in this format. They have also made available a free reader program, *Acrobat Reader*, so it is hardly worthwhile writing another one, at least not commercially. (Ghostscript can also interpret PDF as well as PostScript.)
Where Adobe makes its money is in the sales of programs to produce and manipulate PDF files. Of these, the main ones are:

- Acrobat Distiller to convert PostScript into PDF, and
- Acrobat program to modify existing PDF files.

In the latter case, ‘modification’ means adding links, changing colors, including navigational aids, and inserting notes; it does not mean altering the actual textual content in any way.

Today most graphical and text processing programs are capable of generating PDF output. For the \LaTeX user, there are three routes that may be taken to produce PDF output. The obvious one is \texttt{.tex \rightarrow .dvi \rightarrow .ps \rightarrow .pdf} using \texttt{dvips} to generate the PostScript and then Distiller or Ghostscript to create the PDF file.

The other two methods are to employ a DVI-to-PDF driver, or the \texttt{pdfTeX} program to go directly to PDF, by-passing the intermediate DVI output. We describe each of these in the next sections.

Used on their own, all of these methods produce a simple PDF file containing nothing more than the text itself, a sort of electronic paper. The hypertext features that make an electronic document really live can be added automatically with the help of the \texttt{hyperref} package described in Section 10.2.4.

### 10.2.2 The \texttt{dvipdfm} driver

The driver program \texttt{dvipdfm} by Mark A. Wicks is the equivalent of \texttt{dvips} for PDF output. As its name implies, it converts DVI to PDF output. (The \texttt{m} suffix is to distinguish it from a previously announced \texttt{dvipdf} program that never came to fruition.)

Graphics input can be included with the \texttt{graphics} or \texttt{graphicx} packages (Section 6.1) by specifying the option \texttt{dvipdfm}. The allowed graphics formats are JPEG, PNG, and PDF.

The program is called from a command line with

```bash
dvipdfm myfile
```

to convert \texttt{myfile.dvi} to \texttt{myfile.pdf}. As for \texttt{dvips}, there are many options that may be included. It is best to read the provided manual in \texttt{dvipdfm.pdf} or \texttt{dvipdfm.dvi}, or produce it by processing \texttt{dvipdfm.tex} with \TeX.

The configuration is set up by means of a file named \texttt{config} in the \texttt{dvipdfm} directory. This is set up similarly to the \texttt{config.ps} file for \texttt{dvips}, including the list of mapping files, although these have a different syntax to those of \texttt{dvips}.

Hypertext features and other PDF additions can be added with the \texttt{hyperref} package (Section 10.2.4).
10.2.3 The pdfTEX program

The most direct method of getting PDF output is to process the \LaTeX file with pdfTEX, a variation on the \TeX program developed by Hàn Thế Thành. The following points should be noted:

- pdfTEX behaves just like the regular \TeX program unless the declaration \texttt{\pdfoutput=1} is issued before the first page is output; in this case, it generates PDF instead of DVI output and activates many new \TeX-like commands for including PDF features.

- Since pdfTEX combines the functionality of both the \TeX program and a DVI driver, it must be able to deal with fonts in the way that a driver program would. Originally this meant that it could only handle PostScript type 1 fonts, but now it can also cope with MetaFont bitmap fonts (type 3) as well as TrueType fonts. The bitmap fonts however look very bad when viewed, but this is a viewer problem.

- When started, the program reads a configuration file \texttt{pdftex.cfg} which can set many parameters, as such page size, offsets, PDF compression level, and specifies the names of font mapping files. It may also set the default output to PDF, avoiding having to set \texttt{\pdfoutput} in the source text.

- The font map files are identical to those used by \texttt{dvips} (page 236); they relate the \TeX name for the font to various characteristics, encoding scheme, true name, and specify the file containing the character drawing instructions.

- New \TeX-like commands exist to include PDF features, like compression level, page attributes, document information, opening setup, forms, annotations, links, bookmarks, and article threads.

- Graphics can be included in PNG, TIFF, or JPEG formats; PDF files may be inserted provided they contain only a single page without any fonts or bitmaps. The graphics packages (Section 6.1) support pdfTEX by means of a \texttt{pdftex} graphics option.

To run pdfTEX with \LaTeX (often referred to as pdfLaTeX), it is necessary to generate a \LaTeX format (Section B.1.3) using pdfTEX instead of \TeX, with the \texttt{-ini} option, renaming the resulting \texttt{latex.fmt} to \texttt{pdflatex.fmt}; it is then called with the command

\texttt{pdftex &pdflatex}

Most installations contain a command \texttt{pdflatex} that translates to this combination. The editor shell should also have an icon that invokes this command.
We do not describe the new \TeX-like commands to add PDF features to the output, preferring to use the \LaTeX-like commands of the hyperref package. For information on the intrinsic new pdf\TeX commands, see *The \LaTeX Web Companion* (Goossens and Rahtz, 1999), which also explains the hyperref package in detail. The pdf\TeX manual should also be available on your system under `texmf/doc/pdftex` in various formats.

So which of the above methods is to be recommended? The PostScript + Distiller is the most inelegant, involving two intermediate steps (DVI and PS) and requires purchasing the Distiller program. Alternatively one can use the free GhostView program for the conversion to PDF, but one still has the intermediate steps. And if one selects this route, one must be certain that `dvips` uses the PostScript version of the Computer Modern fonts, by setting up the mapping files accordingly.

The `dvipdfm` program is in the original spirit of \TeX, that uses DVI as a universal intermediate format for all outputs. Purists might tend to respect this ideal. After all, no one ever considered rewriting \TeX to produce PostScript output directly. That said, one must consider that \TeX was invented in the days when no one printer specification dominated the field. Today, PDF is much more than a printer format; it is the means of representing documents electronically. That alone would not justify preferring pdf\TeX over a DVI-to-PDF converter, nor would the fact that it saves a processing step; the deciding argument is that pdf\TeX has established itself as reliable, robust, and flexible.

In the end, it is likely a question of which program one is more comfortable with, and which one has given the better results for the particular user.

### 10.2.4 The hyperref package

Sebastian Rahtz has written an ambitious package `hyperref` to add automatic hypertext links to \LaTeX documents that are intended to become HTML (using \LaTeX2HTML, Section E.1.1 or \TeX4ht, Section E.1.2) or PDF files (using any of the methods described above). Not only do all internal cross-references link to their reference points, citations are also linked to the list of references, table of contents to the section headings, and index listings to the original text. Additional links to external documents are possible by means of a single syntax for all types of output. And it does this with a uniform syntax even though all the conversion methods have their own syntaxes.

Because this package redefines many aspects of the \LaTeX kernel and of many other packages, it should always be loaded as the very last one to be sure that it has the final word.

It is especially for PDF output, whether by means of PostScript plus Distiller, with `dvipdfm`, or directly with pdf\TeX, that this package is extremely useful. Figure 10.1 on the next page gives an example of such a
PDF file, where it should be noted that the table of content entries are all links to their sections, as are the entries in the left-hand outline, which is also generated automatically. The package provides \LaTeX-like commands for the PDF features.

Before we go any further, we should explain certain terms that are used below.

**Links** are indicated as a piece of text in a different color, or surrounded by a frame, or may be an image, which when clicked cause the viewer to jump to another document or some other place in the current document.

**Anchors** (also called *targets*) are the destinations for links. They are provided with an identifying name so that the link can find them. Some anchors are meant only for internal linking within the document, while others are visible from outside.

**Thumbnails** are small images of the pages that are displayed in a thumbnail window by the viewer. They may be used to locate a certain page more quickly, especially if it has an obviously different appearance because of a figure it contains. Double clicking the page image causes the viewer to jump to that page.

**Bookmarks** are special links to places in the document, associated with a *view*, usually meaning a zoom factor, or a specification like ‘fit...
page to width of window’. The bookmark window displays all these, and by double clicking them, one jumps to that location and view. Bookmarks are ordered within a tree, with sub-bookmarks and so on. The bookmark display may have the tree opened or closed.

**Outline** is the usual use for bookmarks, building a table of contents, or outline, for the whole document. The word ‘outline’ is used here, but the PDF viewers refer to ‘bookmarks’. The latter if really far more general.

**Options for hyperref**

There are a large number of parameters that may be configured for the package, either as options in the \usepackage command, or with the \hypersetup command, or with a configuration file. An example of setting options with the first method is:

```latex
\usepackage[pdftex,
    pdftitle={Graphics and Color with LaTeX},
    pdfauthor={Patrick W Daly},
    pdfsubject={Importing images and use of color in LaTeX},
    pdfkeywords={LaTeX, graphics, color},
    pdfpagemode=UseOutlines,
    bookmarks,bookmarksopen,
    pdfstartview=FitH,
    colorlinks, linkcolor=blue, citecolor=blue,
    urlcolor=red,
  ]{hyperref}
```

Options that take a value are given with an equals sign and the value; for multiple values, or text, the values must be placed within curly braces, as shown in the example. Options that are either true or false are simply given (to make them true) or equated to false. Options may also be given with the \hypersetup command, as for example:

```latex
\hypersetup{colorlinks, linkcolor=blue}
```

Values that are colors must be a name of a defined color, as explained in Section 6.2; the name must be defined before it is first used. However, some color values must be the three numbers of the rgb model. The colors of link borders may only be given as rgb. The hyperref package automatically loads the color package if colorlinks is given.

One option must specify the driver name for which the output is intended; in the above example this is pdftex, but other possibilities are:

```latex
dvipdf   dvipdfm dvips dvipsone dviwindow hypertex latex2html pdftex tex4ht textures vtex
```
The option \texttt{hypertex} is not for any particular driver, but rather for any that conforms to the Hyper\TeX\ guidelines, such as \texttt{dvips}. There is also a \texttt{draft} option that turns off all hypertext features.

The next four options in the example (\texttt{pdftitle–pdfkeywords}) are entries for the PDF document information stored in the file and displayed only when the viewer shows the document summary. The complete set of such options is:

\begin{verbatim}
   pdfauthor    pdfcreator    pdfkeywords
   pdfproducer  pdfsubject   pdftitle
\end{verbatim}

The standard paper size options of Section 3.1.1 are all recognized and one of them must be given, either as an option to \texttt{hyperref} or as a global option in the \texttt{\documentclass} command. This is necessary to set up the size of the PDF page. Reminder, these are:

\begin{verbatim}
   a4paper    a5paper    b5paper
   executivepaper    legalpaper    letterpaper
\end{verbatim}

Here is a list of all of the remaining options, with default value in brackets.

4 \hspace{1em} (false) use Acrobat 4 features;

\texttt{anchorcolor} \hspace{1em} (black) set color of anchors (targets of internal links);

\texttt{backref} \hspace{1em} (false) adds links in the bibliography back to the citation; with \texttt{pagebackref}, a list of pages is added to the bibliography, each being a link back to the citation;

\texttt{baseurl} \hspace{1em} determines the prefix for URLs given with \texttt{\href};

\texttt{bookmarks} \hspace{1em} (true) make bookmarks out of sectioning commands;

\texttt{bookmarksnumbered} \hspace{1em} (false) include section numbers in bookmark text;

\texttt{bookmarksopen} \hspace{1em} (false) start document with bookmarks tree opened;

\texttt{bookmarksopenlevel} \hspace{1em} (max) the level to which bookmarks are opened;

\texttt{bookmarkstype} \hspace{1em} (toc) specifies which 'contents' file used for bookmarks;

\texttt{breaklinks} \hspace{1em} (false) allows link text to break over lines;
10.2. Portable Document Format

citebordercolor
(0 1 0) the color of the border around citations;

citecolor
(green) color of citation links when colorlinks given;

colorlinks
(false, but true for tex4ht and dviwindo) link and anchor
texts are to be colored, according to color specifications for the
various types of links; without this, link text is surrounded by a
frame;

debug (false) output diagnostic messages;

draft (false) turn off all hyperlinking;

extension
(dvi) default extension of linked files using the xr package (Sec-
tion 9.2.3); the package tries to guess whether the file specified
by \externaldocument should be .dvi or .pdf; with PDF output,
the viewer will open that external file and switch to it;

filebordercolor
(0 .5 .5) color of the border around file links;

filepath
(cyan) color of file links when colorlinks given;

frenchlinks
(false) use small caps instead of color for links;

hyperfigures
(false) make figures into hyperlinks;

hyperindex
(true) make index entries into links back to referenced text;

hypertextnames
(true) use guessable (obvious) names for links;

implicit
(true) redefine \LaTeX internal commands;

linkbordercolor
(1 0 0) color of the border around links;

linkcolor
(red) color of link text when colorlinks given;

linktocpage
(false) the page numbers in the table of contents (or list of
figures/tables) are the links, rather than the text itself;
menubordercolor
(1 0 0) color of the border around Acrobat menu links;

menucolor
(red) color for Acrobat menu links when colorlinks given;

naturalnames
(false) use \LaTeX-computed names for links;

nesting
(false) links are allowed to be nested, although none of the existing drivers support this;

pageanchor
(true) an anchor (target) is placed automatically on every page, something that is necessary of the table of contents is to act as links;

pagebackref
(false) adds to each entry in the bibliography a list of page numbers linked to their citations;

pagebordercolor
(1 1 0) color of the border around page links;

pagecolor
(red) color of page links when colorlinks given;

pdfborder
(0 0 1, but 0 0 0 with colorlinks) width of the border around links in PDF files; the first two numbers are always zero, the third is the line thickness in points;

pdfcenterwindow
(false) positions the document window in the center of the computer monitor when the PDF file is opened;

pdffitwindow
(false) resizes the document window to fit the first page when the PDF file is opened;

pdfhighlight
(/I) determines how the Acrobat menu buttons behave when clicked; /I for inverse, /N for none, /O for outline, /P inset highlighting;

pdfmenubar
(true) makes the menu bar in the PDF viewer visible;

pdfnewwindow
(false) a new window is opened when the PDF viewer switches to another file;
pdfpagelayout
(SinglePage) determines the page layout for the PDF viewer; other values are OneColumn (all pages in a continuous column), TwoColumnLeft (two columns, with odd numbers on the left), TwoColumnRight (two columns, with odd numbers on the right);

pdfpagemode
(None) determines initial display in the PDF viewer; other values are UseThumbs (show thumbnails), UseOutline (show bookmarks), FullScreen (no tool or menu bars, window is full screen);

pdfpagescrop
a set of four numbers determining the crop box when the PDF file is opened, similar to a bounding box;

pdfpagetransition
(R) determines how the pages are changed; possible values are Split (like opening a curtain), Blinds (like Venetian blinds), Box (a rectangle opens from the center), Wipe (vertical line moves across page), Dissolve (old page dissolves into new), Glitter (variation on dissolve), R (replace); additional parameters can be added where appropriate: /Dm (with Split and Blinds) followed by /H (horizontal) or /V (vertical), /Di (with Wipe and Glitter) followed by 90, 180, 270 for the direction of motion, /M (with Split and Box) followed by /I (inwards), /O (outwards); see page 344 for a simplified interface to this feature;

pdfstartpage
(1) the first page number to be displayed when the PDF file is opened;

pdfstartview
(Fit) determines the initial view when the PDF file is opened; takes values of Fit (fit page to window), FitH (fit to width), FitV (fit to height), FitB (fit to bounding box), FitBH (fit to bounding box width), FitBV (fit to bounding box height), XYZ (no change);

pdftoolbar
(true) makes the tool bar in the PDF viewer visible;

pdfview
(XYZ) determines the page view when a link moves to a new page; takes same values as pdfstartview;

pdfwindowui
(true) makes the user interface elements (bookmarks, thumbnails) in the PDF viewer available; if false, they are permanently switched off;
plainpages
(true) makes the anchors for the pages to be their Arabic form; if false, the page anchors are as the page numbers are formatted, highly recommended if the document contains both Roman and Arabic page numbers;

raiselinks
(false) with the hypertex driver, makes the border around links the actual height of their contents, rather than the current baseline separation;

runbordercolor
(0 .7 .7) color of the border around ‘run’ links;

unicode
(false) enables Unicode encoded PDF strings;

urlbordercolor
(0 1 1) color of the border around URL links;

urcolor
(magenta) color of URL links when colorlinks given;

verbose
(false) issues many extra messages.

Using a configuration file

Very often a user will find that he or she needs the same set of options in almost every document. Rather than issuing that set again and again, it is possible to prepare a configuration file named hyperref.cfg containing a \hypersetup command with all the option settings that one normally wants. One effectively resets the default values. Any settings in the \LaTeX source text itself will override those defaults.

The hyperref.cfg file must be located somewhere where it will always be found by the \LaTeX processing. It might very well be that a file of that name already exists, and will be read in in preference to yours. To check just which location is being used, examine the transcript .log file for the full path name, and then edit that version of hyperref.cfg.

Linking commands

The main purpose of the hyperref package is to add the automatic links between cross-referencing commands and tables of contents. However, it also enables access to PDF features with \LaTeX-style commands. One of these is the ability to add manual links within the file and to other documents, locally or on the Web. The commands to achieve this are:
10.2. Portable Document Format

\hyperbaseurl{url.pre}
sets a prefix for all external URLs specified by other commands, so that the common part of a set of URLs need only be given once; this is the same as the baseurl option, but can be changed within the document;

\href{url}{text}
makes \text into a link to the full URL given by url prefixed with the current base URL; special characters like # and & can be type in url directly;

\hyperimage{url}{image.url}
inserts the image referenced by image.url;

\hyperdef{category}{name}{text}
establishes \text as a target (anchor) with name category.name;

\hyperref{url}{category}{name}{text}
makes \text a link to url#category.name;

an alternate syntax is \hyperref[key]{text} to make \text an internal link to the location marked with \label{key};

\hypertarget{name}{text}
establishes \text as an internal target (anchor) with the name #name;

\hyperlink{name}{text}
makes \text a link to the internal target #name.

The cross-referencing commands \ref{key} and \pageref{key} (Section 9.2.1) automatically create links to the corresponding \label{key} location. As pointed out above, one can also create a manual link with \hyperref[key]{text}.

The hyperref package also provides starred versions \ref* and \pageref* which function as the regular ones but without creating a link.

Adding Acrobat buttons

An electronic document is active, as indicated by the links within it for jumping to other sections. But it is also possible to insert other types of 'actions', in the form of buttons which, when clicked, do something, like flipping to the next page, opening another document, or even quitting.

The hyperref package can add such functionality for the Acrobat viewers, by means of the command

\Acrobatmenu{action}{text}
Table 10.2: Actions available from the Acrobat menus (after Goossens and Rahtz, 1999, page 48)

<table>
<thead>
<tr>
<th>Acrobat Menu</th>
<th>Available options for \Acrobat menu</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Open, Close, Scan, Save, SaveAs, Optimizer:SaveAsOpt, Print, PageSetup, Quit</td>
</tr>
<tr>
<td>File→Import</td>
<td>ImportImage, ImportNotes, AcroForm:ImportFDF</td>
</tr>
<tr>
<td>File→Export</td>
<td>ExportNotes, AcroForm:ExportFDF</td>
</tr>
<tr>
<td>Edit→Prefs</td>
<td>GeneralPrefs, NotePrefs, FullScreenPrefs, Weblink:Prefs, AcroSearch:Preferences(Windows)or, AcroSearch:Prefs(Mac), Cpt:Capture</td>
</tr>
<tr>
<td>Edit→Fields</td>
<td>AcroForm:Duplicate, AcroForm:TabOrder</td>
</tr>
<tr>
<td>Document</td>
<td>Cpt:CapturePages, AcroForm:Actions, CropPages, RotatePages, InsertPages, ExtractPages, ReplacePages, DeletePages, NewBookmark, SetBookmarkDest, CreateAllThumbs, DeleteAllThumbs</td>
</tr>
<tr>
<td>View</td>
<td>ActualSize, FitVisible, FitWidth, FitPage, ZoomTo, FullScreen, FirstPage, PrevPage, NextPage, LastPage, GoToPage, GoBack, GoForward, SinglePage, OneColumn, TwoColumns, ArticleThreads, PageOnly, ShowBookmarks, ShowThumbs</td>
</tr>
<tr>
<td>Tools</td>
<td>Hand, ZoomIn, ZoomOut, SelectText, SelectGraphics, Note, Link, Thread, AcroForm:Tool, Acro_Movie:MoviePlayer, TouchUp:TextTool, Find, FindAgain, FindNextNote, CreateNotesFile</td>
</tr>
<tr>
<td>Window</td>
<td>ShowHideToolBar, ShowHideMenuBar, ShowHideClipboard, Cascade, TileHorizontal, TileVertical, CloseAll</td>
</tr>
<tr>
<td>Help(Windows)</td>
<td>About</td>
</tr>
</tbody>
</table>
where *text* becomes a button to execute *action* when clicked. Of course, *text* can be more than mere text, it is often a graphics that really does look like a button.

The actions available are those provided by the Acrobat and Acrobat Reader programs. These are not documented by Adobe, but are listed in Table 10.2 on the facing page, which is taken from Goossens and Rahtz (1999), who credit Merz (1998) for having figured them out. These actions are therefore specific to these viewers only, and may change with later versions. They may all be executed by the viewers themselves, by means of pull-down menus in the menu bar, indicated in the left column of Table 10.2.

The main purpose of this command is to set up navigational controls on each page of a document that is meant essentially for screen viewing. The regular tool bar may be turned off, which is why the document itself must provide it. This is best put into a footline, and that is most conveniently done with the fancyhdr package (Section 3.2.2).

\documentclass\[a4paper\]{article}
\usepackage{fancyhdr}
\usepackage[ pdftex, colorlinks, pdftoolbar=false, pdfmenubar=false, pdfwindowui=false, pdfpagemode=FullScreen, ]{hyperref}
\pagestyle{fancy}
\cfoot{\navbar}
\newcommand{\navbar}{\Acrobatmenu{FirstPage}{$\Leftarrow$} \Acrobatmenu{PrevPage}{$\leftarrow$} \Acrobatmenu{NextPage}{$\rightarrow$} \Acrobatmenu{LastPage}{$\Rightarrow$} \Acrobatmenu{Close}{$\bullet$}}

The above produces a footline containing

$$\Leftarrow\leftarrow\rightarrow\Rightarrow\bullet$$

(by default in red) as clickable buttons to jump to the first page, to the previous page, to the next page, to the last page, respectively. And clicking the bullet $\bullet$ closes the document. Of course, it looks even better if icons are made up and inserted with \includegraphics.
The original versions of \TeX and \LaTeX were set up for the English language, American variant. The built-in hyphenation patterns were for English only, and several English words such as ‘Figure’ and ‘Bibliography’ were included explicitly in certain commands. This in fact violates the rules of good programming which forbid doing anything explicitly. In Europe, \LaTeX users quickly made customizations for their particular languages. The adaptation for German, \texttt{german.sty}, is a set of macros from many contributors collected together by H. Partl of the Technical University of Vienna, and has become the standard for the German-speaking \TeX Users Group (DANTE). It contains some facilities for other languages, including French and English.

The key to the \texttt{german} package and other language adaptations is the fact that the standard \LaTeX document styles (and later classes for \LaTeXe) had been altered so that explicit English words in the output were replaced by reprogrammable command names. These names had become standardized among European users for application to all languages. The modified \LaTeX version by J. Schrod from Darmstadt was known as \texttt{ILaTeX}, for International \LaTeX. These names and their standard English values are to be found on page 459.

As of December 1, 1991, these naming features became part of the \LaTeX standard. They represent good programming practice, since they allow even English-speaking users to change certain titles easily, for example ‘Abstract’ into ‘Summary’ or ‘Contents’ into ‘Table of Contents’. Naturally, they were taken over, and extended upon, by \LaTeXe.

Another important feature for multilingual usage is the \TeX \texttt{\language} counter which allows more than one set of hyphenation patterns to be stored in the format with \texttt{initex} (Section B.1.3). Different patterns are activated by setting \texttt{\language} to the appropriate number. This is now standard for all \TeX version over 3.0, which should now be universal.

Individual language packages such as \texttt{german} maintained by Bernd Raichle and \texttt{french} by Bernard Gaulle have now been superseded to a large extent today by the universal multilanguage system: \texttt{babel}.
11.1 The babel system

The babel system has been developed by Johannes Braams at the Dutch PTT Research Laboratories. Its main purpose is to provide a standard means of switching languages with a flexible method for loading the hyphenation patterns. It has been written to work with both \TeX and \LaTeX{}.

The three main requirements for a single language adaptation are: translation of the explicit English words, special commands to simplify certain features of that language, and selection of the appropriate hyphenation patterns. To these babel adds: removal of the special commands and testing for the current language. The language-specific files must conform to the babel selection command structure. The concept of 'dialects' is also included, that is, two 'languages' that share a common set of hyphenation patterns. Examples are German and Austrian as well as British and US English, which only differ in the form of the dates.

At present, the babel package contains definition files for the following languages (more can be added at any time):

- Afrikaans
- Bahasa
- Basque
- Breton
- Bulgarian
- Catalan
- Croatian
- Czech
- Danish
- Dutch
- English
- Esperanto
- Estonian
- Finnish
- French
- Galician
- German
- Greek
- Hebrew
- Hungarian
- Icelandic
- Irish Gaelic
- Italian
- Latin
- Lower Sorbian
- North Sami
- Norwegian
- Polish
- Portuguese
- Romanian
- Russian
- Scottish Gaelic
- Spanish
- Slovakian
- Slovenian
- Swedish
- Serbian
- Turkish
- Ukrainian
- Upper Sorbian
- Welsh

The necessary definition files are delivered with the installation. The hyphenation patterns for the various languages must be obtained from another source, however. For most of these files, there are no special commands for accents or punctuation: they only redefine the naming commands for those words that normally print English text.

The german.sty package has been a great inspiration for babel. However, because of the name conflict, it was necessary to create a new germanb.sty (and ngermanb.sty for the new German grammar rules) for babel applications, which contains the same set of special commands, but with a few babel additions. Similarly, the babel French file had to be named frenchb.sty to distinguish it from the existing french.sty.

To exploit the babel features most fully, one should produce a \LaTeX{} format (Section B.1.3) that incorporates it directly. To this end, the hyphen.cfg supplied with babel must be used with the language.dat file configured for the local languages and hyphenation patterns that are to be pre-loaded. Only in this way can one conveniently get the right word division for each language.

Alternatively, one can use a normal \LaTeX{} format and load babel as a regular package, as illustrated below. The switching of hyphenation
patterns will then be questionable.

The \texttt{babel} files

The \texttt{babel} installation provides the following files for its implementation:

\texttt{babel.def} contains some of the basic macros for running \texttt{babel}, and must be loaded by the first language file read in; the remaining macros are either already in the format itself (loaded from \texttt{hyphen.cfg}) or read in from \texttt{switch.def}; \texttt{babel.def} determines which is the case and reacts accordingly;

\texttt{switch.def} contains the additional \texttt{babel} macros, to be read in if these macros are not already stored in the \LaTeX\ format;

\texttt{hyphen.cfg} contains the same macro definitions as in \texttt{switch.def}, plus some more that are to be run by \texttt{initex}; if this file is available during the creation of the \LaTeX\ format (Section B.1.3), these macros are built into that format and file \texttt{switch.def} is not necessary at runtime;

\texttt{babel.sty} is a master package that loads the language files specified as options;

\texttt{language.dat} contains a list of languages and the file names of their hyphenation patterns; this file is read in by \texttt{hyphen.cfg} during the \texttt{initex} run; it is the only file that may (must!) be edited for the particular installation (Section 11.2);

\texttt{esperant.ldf}... the language definition files.

\texttt{esperant.sty}... compatibility files that read in the \texttt{.ldf} files.

Invoking \texttt{babel}

The \texttt{babel} package is loaded with the desired languages as options:

\begin{verbatim}
\usepackage[english,esperanto]{babel}
\end{verbatim}

Alternatively, the language names may be used as global options, something that is recommended if there are other packages that take the languages as options, as

\begin{verbatim}
\documentclass[english,esperanto]{article}
\usepackage{babel,varioref}
\end{verbatim}

In both cases, the last named language is the one that is immediately active.

The recognized language names that may be used as options are:
afrikaans =canadian =polution =portuguese
bahasa esperanto greek =brazilian
basque estonian hebrew =brazil
breton finnish magyar =romanian
bulgarian french =hungarian russian
catalan =francais icelandic =scottish
croatian =canadien irish =spanish
czech =acadian italian slovak
danish galician latin slovene
dutch german lowersorbian swedish
english =germanb samin =serbian
=USenglish =ngerman norsk =turkish
=american =austrian =nynorsk =ukrainian
=UKenglish =naustrian polish uppersorbian
=british greek portuges welsh

Those names preceded by = are synonyms for the language option above them. (The equals sign is not part of the option name.) For example, both hungarian and magyar load the file magyar.ldf.

If the babel package is loaded, all it does is read in the specified language packages; they in turn read in babel.def and possibly switch.def (if its macros are not already in the format).

Language switching commands

The normal user needs to know very little about the babel internal operations. The new high-level user commands are:

\selectlanguage{language}
\begin{otherlanguage}{language} text \end{otherlanguage}
\begin{otherlanguage*}{language} text \end{otherlanguage*}
\foreignlanguage{language}{text}
\iflanguage{language}{yes_text}{no_text}

The \selectlanguage command and the otherlanguage environment are two ways to switch to another language, with all its features and translations. The \foreignlanguage command and the otherlanguage* environment also switch to language but without the translations or date changes; they are meant for setting short sections of text in a language different from the regular one.

The \iflanguage executes yes_text if language is current otherwise no_text.

In addition, \languagename contains the name of the currently selected language.

Of course, each language definition file can also have its own special commands to assist typing accented and special characters. For example,
11.1. The babel system

in German, one may type "a as a shorthand for \"a. To find out what
special commands are available for each language, and any other spe-
cial considerations, one must read the documentation for that language.
Documentation should be found in the texmf/doc/tex/generic/babel
directory, or you can process the desired language.dtx file in the texmf/
source/tex/generic/babel directory. (These paths might be different
on your system.)

Contents of a language definition file

Although one does not normally need to know anything about how the
switching mechanism functions, we will outline it here for the interested
user.

Two additional internal babel commands are:

\addlanguage{lang_num} \ and
\adddialect{lang_num.1}{lang_num.2}

where lang.num is an internal language number for specifying the set of
hyphenation patterns. The command \addlanguage sets its argument
equal to the next available language number. The form of lang.num used
in babel is \l@language; for example, \l@english. This command is
executed on those language names listed in language.dat during the
initex run. The command \adddialect sets the first argument equal to
the second so that the two languages make use of the same set of hy-
phenation patterns. For example, in english.ldf there is the command
\adddialect{\l@american}{\l@english}.

A language definition file must provide four commands:

\captionslang to redefine the naming commands like \tablename
(see Section D.4.1);
\datelang to define \today (see Section D.4.2);
\extraslang to define any language-specific commands;
\noextraslang to remove the language-specific commands.

If the language lang is not one of those with prestored hyphenation
patterns in the current format file (that is, if \l@lang is undefined), it is
set to be a ‘dialect’ of language number 0.

Finally, the language definition file calls \selectlanguage{lang} to
activate that language. This is accomplished by

• calling \language\l@lang to select the set of hyphenation patterns,

• invoking \originalTeX in order to remove any existing language-
specific commands,

• activating \captionslang, \datelang, and \extraslang to invoke
the language-specific names, date, and commands,
• redefining \originalTeX{} to be \noextraslang{} so that the next language switch will remove those features specific to \textit{lang}.

## 11.2 Contents of the \texttt{language.dat} file

As has been mentioned before, with \TeX{} version 3.0 and later it is possible to store more than one set of hyphenation patterns in the format file. The counter \texttt{language} is used to switch between them by setting it to a different number.

However, there is no standard to dictate which languages belong to which numbers. If a package were to assume a certain ordering, it would most certainly function incorrectly at installations other than the one for which it was designed. The \texttt{babel} system invokes a more reliable procedure.

During the production of the format by \texttt{initex} (Section B.1.3), the file \texttt{hyphen.cfg} is input, which in turn reads in \texttt{language.dat}, the only file to be tailored to the local installation. This file contains a list of languages to add as well as the name of the file with the hyphenation patterns and the name of any additional file to be included. It also indicates names of dialects that use the same hyphenations by prefixing the name with an equals sign. For example, if \texttt{language.dat} contains

```
=USenglish
american ushyphen.tex
english ukhyphen.tex
=UKenglish
=british
french frhyphen.tex
german dehyphn.tex
ngerman dehyphn.tex
```

the hyphenation patterns stored in files \texttt{ushyphen.tex}, \texttt{ukhyphen.tex}, \texttt{frhyphen.tex}, \texttt{dehyphn.tex}, and \texttt{dehyphn.tex} are loaded under the language numbers 0 to 4 respectively, and \texttt{\@american}, \texttt{\@english}, \texttt{\@french}, \texttt{\@german}, and \texttt{\@ngerman} are defined as numbers 0 through 4 for use with the \texttt{\selectlanguage} command. Languages \texttt{\@USenglish} and \texttt{\@UKenglish} are synonyms for the current language, in this case 0 and 1, respectively. Similarly \texttt{\@british} is identical to \texttt{\@UKenglish}. 
Math Extensions with \texttt{\textcopyright AMS-L\LaTeX}

The American Mathematical Society, \texttt{\textcopyright AMS}, has supported the development and usage of \LaTeX since its first release. It in fact owns the \LaTeX logo as a registered trademark.

Shortly after \LaTeX 82 became available, the \texttt{\textcopyright AMS} produced a macro package for generating a special format \texttt{amstex}, described in \textit{The Joy of \LaTeX} by Spivak (1986). The macros in this \texttt{\textcopyright AMS-\LaTeX} complement the mathematical typesetting features of Plain \LaTeX by adding additional ones and simplifying others.

However, \texttt{\textcopyright AMS-\LaTeX} is not a documentation preparation system like \texttt{\LaTeX}, describing the logical layout of a document by means of markup commands, but rather is simply an extension to Plain \LaTeX.

The great popularity of \LaTeX as a text formatting program is due primarily to the availability of \texttt{\LaTeX} as a user-friendly interface to the underlying \LaTeX machinery. Many authors therefore asked the \texttt{\textcopyright AMS} to provide \texttt{\LaTeX} with the same mathematical features as in \texttt{\textcopyright AMS-\LaTeX}. The \texttt{\textcopyright AMS-\LaTeX} project was thus launched in 1987, with version 1.0 completed three years later by Frank Mittelbach and Rainer Schöpf, together with Michael Downes of the \texttt{\textcopyright AMS}. \texttt{\textcopyright AMS-\LaTeX} has been fully converted to \texttt{\LaTeX2\epsilon} with version 1.2 in 1996.

The \texttt{\textcopyright AMS-\LaTeX} collection consists of three parts: packages for extending mathematical typesetting, extra classes for articles and books published by the \texttt{\textcopyright AMS}, and supplemental fonts for additional symbols, math alphabets, and Cyrillic fonts.

In the next sections, we give an overview of the math extensions available in the package \texttt{amsmath}; a more detailed user’s manual is delivered with the collection, in the file \texttt{amsldoc.tex}. How to invoke the additional fonts is described below in Section 12.4, or in the \texttt{\textcopyright AMS-supplied manual found in the file amsfndoc.tex}. The extra classes will not be described in this book; \texttt{\textcopyright AMS} authors should refer to the instructions in the document \texttt{instr-1.tex} provided with \texttt{\textcopyright AMS-\LaTeX}. 
Chapter 12. Math Extensions with $\text{AMS-LATeX}$

12.1 Invoking $\text{AMS-LATeX}$

If the $\texttt{documentclass}$ statement at the beginning of the $\text{LATeX}$ document selects one of the $\text{AMS}$ classes $\text{amsbook}$, $\text{amsart}$, or $\text{amsproc}$, then most of the $\text{AMS-LATeX}$ features are loaded automatically. These features may still be employed with other classes by including the main extension package $\text{amsmath}$ by means of

\begin{verbatim}
\usepackage[options]{amsmath}
\end{verbatim}

in the document's preamble. The list of allowable options is described below in Section 12.2.8.

The $\text{amsmath}$ package defines many of the new math typesetting features itself, but it also loads a number of other packages from the $\text{AMS-LATeX}$ collection that contain further extensions, such as $\text{amsopn}$, $\text{amstext}$, and $\text{amsbsy}$. These packages could be loaded separately without $\text{amsmath}$ if only their limited features are wanted. On the other hand, the packages $\text{amscd}$ and $\text{amsthm}$ are not included in $\text{amsmath}$ and must be loaded explicitly if their features are desired. Simply add their names to the list of packages in $\texttt{\usepackage}$. The following Section 12.2 describes the new commands and environments made available with $\text{amsmath}$ and its associated packages. We call these the standard features of $\text{AMS-LATeX}$. Further extensions added by other packages are explained afterwards.

The examples in this chapter employ the user-defined commands $\texttt{\mi}$, $\texttt{\me}$, and $\texttt{\dif}$ defined in Section 5.4.10 for printing upright i, e, and d in math mode.

12.2 Standard features of $\text{AMS-LATeX}$

This relatively long section presents those standard features of $\text{AMS-LATeX}$ that are activated by loading the $\text{amsmath}$ package, or by selecting one of the $\text{AMS}$ classes $\text{amsart}$, $\text{amsbook}$, or $\text{amsproc}$.

12.2.1 Additional font switching commands

Standard $\text{LATeX}$ provides the math alphabet commands $\texttt{\mathcal}$, $\texttt{\mathbf}$, $\texttt{\mathsf}$, $\texttt{\mathfrak}$, $\texttt{\mathit}$, $\texttt{\mathtt}$ for changing fonts within math mode (Section 5.4.2). With $\text{AMS-LATeX}$, one may also use the commands

\begin{verbatim}
\boldsymbol{symbol} \quad \text{and} \quad \pmb{symbol}
\end{verbatim}

to print $\texttt{symbol}$ in a bold face, provided there is an appropriate bold font for it. Whereas the command $\texttt{\mathbf}$ sets only Latin letters, numbers, and Greek upper case letters in bold, these commands also affect math symbols and Greek lower case letters. Compare the result of \$\texttt{\mathbf{\nabla\times V\,d\sigma}}\$ ($\nabla \times \mathbf{V} d\sigma$) with that
of $\nabla \times V \, d\sigma$. With the standard \LaTeX command \texttt{mathbf}, only the letters $V$ and $d$ appear bold, while the other characters remain in normal weight. Not only that, these letters are upright, and not italic as required by international standards. With \texttt{boldsymbol}, all symbols are bold and italic.

Those symbols for which no bold face font exists will remain in normal weight with the \texttt{boldsymbol} command. Such symbols are, for example, those that come in two sizes (Section 5.3.7) like $\sum$, $\int$, $\cup$, and so on. The command \texttt{pmb} (poor man’s bold) simulates a bold face even for these symbols by printing them several times slightly displaced. The result of $\texttt{pmb}{\sum\;\int\;\bigcup}$ is $\sum \int \bigcup$.

These commands are defined in the package \texttt{amsbsy}, which may be loaded separately without \texttt{amsmath}.

A short piece of normal text can be given within a formula with \texttt{amstext}

\begin{verbatim}
\text{short text}
\end{verbatim}

In contrast to \texttt{mbox{short text}} for standard \LaTeX, the \texttt{text} command switches font size correctly when used as superscripts and subscripts. Thus \texttt{\text{Word}} sets \texttt{Word} lower and changes to \texttt{\scriptstyle} font size. To achieve the same effect in standard \LaTeX, the font size must be given explicitly, as \texttt{\text{\scriptstyle Word}}. Furthermore, the name \texttt{text} is more precise than \texttt{mbox}.

This command is defined in the package \texttt{amstext}. Like \texttt{amsbsy}, it may be loaded on its own without \texttt{amsmath} if none of the other \texttt{AMS-LaTeX} features are wanted.

Another command to insert normal text inside a displayed equation is \texttt{intertext}

\begin{verbatim}
\text{insert text}
\end{verbatim}

The \texttt{insert text} is inserted as a left-justified line of text between those of the formula. The alignment of the formula lines remains unaffected, something that is not guaranteed if one closes the displayed equation, inserts text and reopens the equation. For example:

$$(x + iy)(x - iy) = x^2 - ixy + ixy - i^2y^2$$

$$= y^2 + y^2 \quad \text{since} \quad i^2 = -1 \quad \text{is true.}$$

On the other hand

$$(x + iy)^2 = x^2 + 2ixy + i^2y^2 = x^2 + 2ixy - y^2$$

$$(x - iy)^2 = x^2 - 2ixy + i^2y^2 = x^2 - 2ixy - y^2$$

\begin{verbatim}
\begin{align*}
(x + \mi y)(x - \mi y) & = x^2 - \mi xy + \mi xy - \mi ^2y^2 \\
& = y^2 + y^2 \quad \text{\texttt{\quad text{since}}} \quad \text{\quad \texttt{\quad \quad is true.}}
\end{align*}
\end{verbatim}
\begin{align*}
\intertext{On the other hand}
(x + \im y)^2 &= x^2 + 2\im xy + \im^2 y^2 = x^2 + 2\im xy - y^2 \\
(x - \im y)^2 &= x^2 - 2\im xy + \im^2 y^2 = x^2 - 2\im xy - y^2
\end{align*}

The alignment of the lines on the first equals sign is maintained through the interruption with the line of text ‘On the other hand’. The environment align (Section 12.2.6) is one of several new ones provided by \texttt{AMS-\LaTeX} to replace the standard \texttt{\LaTeX} environment eqnarray. Note that \texttt{\intertext} may only be issued immediately after the \texttt{\\} for starting a new line.

\subsection*{12.2.2 Multiple mathematical symbols}

Mathematical formulas often require the same symbol to appear several times, such as multiple integral signs, or symbols and arrows to be stacked above or below a mathematical expression. Usually the distances between these symbols depends on the mathematical meaning, something that \texttt{\LaTeX} cannot automatically recognize. \texttt{AMS-\LaTeX} provides a number of structures to help the user find the right spacing without tedious trial and error.

\subsubsection*{Multiple integrals}

With \texttt{AMS-\LaTeX}, the commands \texttt{\iint}, \texttt{\iiint}, \texttt{\iiiint}, and \texttt{\idotsint} output multiple integrals, with upper and lower limits being added in the usual manner. In text formulas, they are printed as $\iint$, $\iiint$, $\iiiint$, and $\idotsint$, while in displayed formulas, they appear as

\begin{align*}
\iint_S f(x,y) \, dS & \quad \text{[ \texttt{\iint}limits\_S f(x,y), \texttt{\diff} S \]} \\
\iiint_V f(x,y,z) \, dV & \quad \text{[ \texttt{\iiint}limits\_V f(x,y,z), \texttt{\diff} V \]} \\
\iiiint_G f(x,y,z,t) \, dG & \quad \text{[ \texttt{\iiiint}limits\_G f(x,y,z,t) \texttt{\\}, \texttt{\diff} G \]} \\
\idotsint_U f(x_1, \ldots, x_k) \, dU & \quad \text{[ \texttt{\idotsint}limits\_U f(x_1, \texttt{\ldots}, x_k), \texttt{\diff} U \]}
\end{align*}

whereby the command \texttt{\limits} may be left off if the option intlimits has been specified when \texttt{amsmath} was loaded (see Section 12.2.8).
12.2. Standard features of $\texttt{AMS-\LaTeX}$

**Multiline limits**

One often needs multiline limits or indices for summations, as in the examples below.

$$\Delta_{\substack{p_1 p_2 \cdots p_{n-k} \\ q_1 q_2 \cdots q_{n-k}}} \quad \sum_{\substack{k_0, k_1, \ldots \ge 0 \\ k_0 + k_1 + \cdots = 0}} a_{0k_0} a_{1k_1} \cdots$$

$\texttt{AMS-\LaTeX}$ provides a command $\backslash \text{substack}$ for this purpose, with syntax

$\backslash \text{substack}\{1st\ line\|2nd\ line\|\ldots\| last\ line\}$

where the command must immediately follow the $^\wedge$ or $_\_\_$ shifting commands, and be entirely enclosed in curly braces $\{\}$. The index for the left-hand example above was generated with

$\Delta_{\substack{p_1 p_2 \cdots p_{n-k} \\ q_1 q_2 \cdots q_{n-k}}}$

The lines of text printed by $\backslash \text{substack}$ are centered horizontally, as is apparent in the right-hand example above. This was produced with

$\[ \sum_{\substack{k_0, k_1, \ldots \ge 0 \\ k_0 + k_1 + \cdots = 0}} a_{0k_0} a_{1k_1} \cdots \]$  

On the other hand, the environment $\text{subarray}$ offers more control over the horizontal alignment:

$\begin{subarray}\{pos\} 1st\ line\|2nd\ line\|\ldots\| last\ line\end{subarray}$

The argument $pos$ may be $c$ for centered, or $l$ for left-justified lines.

$$\sum_{i \in \Lambda \atop i < j < n} P(i, j) \quad  \[ \sum\{\begin{subarray}{l} i \in \Lambda \\ i < j < n \end{subarray}\} P(i, j) \]$  

**Special limits**

To add a differential prime sign to a summation symbol without a lower limit, like $\sum' E_n$, in a displayed formula, one can easily give

$\[ \sum\backslash \text{nolimits}' E_n \]$  

However, if there to be an upper limit as well, the prime can only be added with much fiddling. The $\backslash \text{sideset}$ command simplifies this task considerably.

$$\sum_{n=0}^{\infty} (2n + 1) E_{2n+1} \quad  \[ \backslash \text{sideset}\{\}'\backslash \text{sum}\{n=0\}'\backslash \text{nfty} (2n+1) E_{\{2n+1\}} \]$  

The complete syntax for this command is:
\sideset{pre}{post}\symbol

where \(pre\) and \(post\) are the superscripts and subscript commands to be added before and after the \(\symbol\), respectively. They must contain raising and lowering operators \(^\ast\) and \(\_\).

The product symbol \(\prod\) is given daggers above and asterisks below with \(\sideset{\_\dag\_\dag}{\_\dag\_\dag}\prod\), as shown at the right.

Additional shifting commands are \(\overset{\text{char}}{\symbol}\) and \(\underset{\text{char}}{\symbol}\) which place the arbitrary \(\text{char}\) above or below the \(\symbol\) in the size appropriate for superscripts and subscripts. Thus \(\overset{\ast}{X}\) produces \(\ast X\) and \(\underset{\ast}{X}\) yields \(X\ast\).

**Extended arrows**

The \texttt{amsmath} package provides a number of commands to produce extra long arrows for combining with mathematical expressions. The commands

\[
\overleftarrow{expr} \quad \underleftarrow{expr} \\
\overrightarrow{expr} \quad \underrightarrow{expr} \\
\overleftrightarrow{expr} \quad \underleftrightarrow{expr}
\]

produce lengthened arrows pointing left and right, as well as double arrows above and below the mathematical expression \(expr\).

\[
\begin{eqnarray*}
\overrightarrow{ABCD} & = & \underrightarrow{AB} + \underrightarrow{BC} + \underrightarrow{CD} \\
\overleftarrow{ABCD} & = & \underleftarrow{DC} + \underleftarrow{CB} + \underleftarrow{BA} \\
\overleftrightarrow{ABCD} & = & \underleftrightarrow{DCAB}
\end{eqnarray*}
\]

The lower arrows are actually too close to the expression, as is seen on the right-hand side of the last example. This should also be the case for all the other examples with a lower arrow, except that we have added a strut to push them down somewhat:

\[
\underleftarrow{\rule[-2pt]{0pt}{2pt}AB} \quad \text{... and} \quad \underleftarrow{\rule[-2pt]{0pt}{2pt}DC} \quad \text{...}
\]

A smaller font size will be used for the arrows when they appear in exponents, indices, superscripts, and subscripts:
\[ \int_{0.2\pi} r \, d\phi = 2\pi r \]
\[ r', d\varphi = 2\pi r \]

There are two more commands for horizontal arrows of variable length:
\[
\xleftarrow{below}{above} \quad \xrightarrow{below}{above}
\]
which place the mandatory \textit{above} in superscript size over the arrow, and the optional \textit{below} in subscript size beneath it.

\[
A \xleftarrow{n+\mu-1} B \xrightarrow[T]{n\pm i-1} C
\]

The package \texttt{amscd} (Section 12.3.2) offers further possibilities for combining arrows and text.

\textbf{Stacked accents}

Attempting to place multiple accents over a character with standard \LaTeX, say with $\hat{\hat{A}}$, results in a misplacement: \(\hat{\hat{A}}\). The same applies to the other mathematical accents commands $\check$, $\breve$, $\acute$, $\grave$, $\tilde$, $\bar$, $\vec$, $\dot$, and $\ddot$ (Section 5.3.9). The \texttt{amsmath} package provides a set of math accent commands with the same names but capitalized,

\[
\Hat \Breve \Grave \Bar \Dot \Check \Acute \Tilde \Vec \Ddot
\]
which can be safely combined with one another for the expected results: $\Hat{\Hat{A}}$, $\Breve{\Bar{B}}$, and $\Tilde{\Tilde{C}}$ produce \(\hat{\hat{A}}, \breve{\bar{B}}, \text{and } \tilde{\tilde{C}}\).

Three or four dots in a row over a symbol are often used for time derivatives of third and fourth order. They can be placed with the commands

\[
\dddot{\text{sym}} \quad \text{and} \quad \ddddot{\text{sym}}
\]
In this way \(\dddot{u}\) and \(\ddddot{u}\) are produced with $\dddot{u}$ and $\ddddot{u}$.

\textbf{Continuation dots}

In standard \LaTeX, one has the commands \texttt{\ldots} and \texttt{\cdots} for printing three continuation dots, either on the baseline or raised to the center of the line. \texttt{AMS-L\TeX} offers a number of additional possibilities. The most general of these is the \texttt{\dots} command which adjusts the vertical height according to the symbol that follows it. If this is an equals sign or binary operator, such as $+$ or $-$, the dots are raised, as with \texttt{\cdots}, otherwise they are on the baseline, as with \texttt{\ldots}.
\$a_0+a_1+\ldots+a_n \implies a_0 + a_1 + \cdots + a_n$
\$a_0,a_1,\ldots,a_n \implies a_0, a_1, \ldots, a_n$

If the continuation dots come at the end of a formula, there is no following symbol to determine the height of the dots. In this case, one must manually indicate the height by means of one of the commands \texttt{\ldots c} (comma), \texttt{\ldots b} (binary), \texttt{\ldots m} (multiplication), or \texttt{\ldots i} (integral).

The \texttt{\ldots c} is to be used with commas, so \$A_1,A_2,\ldots\$ produces \(A_1,A_2,\ldots\); the \texttt{\ldots b} sets them for a binary operator, thus \$A_1+A_2,\ldots\$ yields \(A_1 + A_2 + \cdots\); the command \texttt{\ldots m} is actually identical to \texttt{\ldots b}, but it is logically meant to be applied to multiplication, \$A_1A_2\ldots\$ makes \(A_1A_2\cdots\); finally \texttt{\ldots i} places the dots at the mean height of an adjacent integral sign.

\[\int_{A_1} \int_{A_2} \ldots\]

### 12.2.3 Fractions

The \TeX{} fraction commands \texttt{\atop}, \texttt{\choose}, and others may be allowed in standard \LaTeX{}, but not in \AMSLaTeX{}. With the \texttt{amsmath} package, only those fraction commands described here may be used.

#### Basic fraction commands

In addition to the regular \LaTeX{} command \texttt{\frac{over}{under}}, \AMSLaTeX{} provides the commands \texttt{\tfrac} and \texttt{\dfrac} with the same syntax. These are effectively the same as \texttt{\frac} but with the font set to \texttt{\textstyle} or \texttt{\displaystyle}, respectively (Section 5.5.2). We demonstrate their effects with some examples from the \AMSLaTeX{} user's manual.

\[
\left[ \frac{1}{k} \log_2 c(f) \right]^{qquad} \left[ \tfrac{1}{k} \log_2 c(f) \right]^{qquad} \left[ \sqrt{\dfrac{1}{k} \log_2 c(f)} \right]^{qquad} \left[ \sqrt{\tfrac{1}{k} \log_2 c(f)} \right]^{qquad}
\]

#### Binomial expressions

A binomial expression looks something like a fraction, but is enclosed in round parentheses and is missing the horizontal rule. The basic command in the \texttt{amsmath} package is

\[\binom{over}{under}\]
which functions in the same way as \texttt{\frac} and the other fraction commands.

\[
\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}
\]

Similarly there are the commands \texttt{\tbinom} and \texttt{\dbinom} analogous to \texttt{\frac} and \texttt{\binom}.

**User-defined fractions**

The \texttt{amsmath} package provides a powerful tool for defining fraction-like structures:

\[
\genfrac{\left(}{\right)}{0pt}{}{\text{mathsize}}{\over}{\under}
\]

where \texttt{left} and \texttt{right} are the parenthesis characters on the left and right, \texttt{thickness} is the thickness of the horizontal line, and \texttt{mathsize} is a number 0–3 representing the math sizes \texttt{\displaystyle}, \texttt{\textstyle}, \texttt{\scriptstyle}, and \texttt{\scriptscriptstyle}, respectively. The last two arguments, \texttt{over} and \texttt{under}, are the texts in the two parts of the fraction, the same arguments as in \texttt{\frac} and \texttt{\binom}.

If the \texttt{thickness} is left blank, the standard thickness for \LaTeX fractions is used. If \texttt{mathsize} is empty, the font size is determined automatically by the normal rules in Section 5.5.2.

Rather than repeating \texttt{\genfrac} with the same first four arguments time and again, one should define new fraction commands with it. For example, the following definitions are given in \texttt{amsmath.sty}:

\begin{verbatim}
\newcommand{\frac}{\genfrac{}{}{0pt}{}{#1}{#2}}
\newcommand{\dfrac}{\genfrac{}{}{0pt}{}{#1}{#2}}
\newcommand{\tfrac}{\genfrac{}{}{1pt}{}{#1}{#2}}
\newcommand{\binom}{\genfrac{(}{)}{0pt}{}{#1}{#2}}
\end{verbatim}

As a further example, consider the redefinition of the command \texttt{\frac}

\[
\renewcommand{\frac}[3]{\genfrac{}{}{#1}{}{#2}{#3}}
\]

in which the line thickness is now an optional first argument; without this optional argument, the command behaves as normal. Thus

\[
\binom{n}{m} = \frac{\frac{n!}{M!(n-m)!}}{\text{yields } \frac{n}{m} = \frac{n!}{M!(n-m)!}}
\]

**Continued fractions**

Continued fractions can be made in \texttt{\LaTeX} with the command

\[
\cfrac{\text{pos}}{\over}\{\text{under}\}
\]

whereby the denominator \texttt{under} may contain further \texttt{\cfrac} commands.
\[ a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{a_4 + \ldots}}}} \]

produces

\[
a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4 + \cdots}}}}
\]

If the optional argument \textit{pos} is missing, the numerator \textit{over} is centered on the horizontal rule; otherwise it may take values of 1 or r to left or right justify the numerator.

### 12.2.4 Matrices

Standard \LaTeX{} possesses the environment \texttt{array} for producing arrays and matrices. The \texttt{amsmath} package provides the additional environments \texttt{pmatrix}, \texttt{bmatrix}, \texttt{Bmatrix}, \texttt{vmatrix}, and \texttt{Vmatrix}, which automatically add the enclosing braces (, [, {, | | and || | | around the array, and in the right size. For completeness, there is also a matrix environment with no braces.

In contrast to the standard \texttt{array} environment (Section 4.8.1), these matrix environments do not require an explicit column specification as argument. By default, up to 10 centered columns may be used without any argument. (This maximum number may be changed by giving the special counter \texttt{MaxMatrixCols} a new value with either \texttt{setcounter} or \texttt{addtocounter}.) Otherwise the matrix environments are used in the same way as the \texttt{array} environment.

The following example is taken from Section 5.4.3 on page 134 and is recast here using \LaTeX{} constructs.

\[
\sum_{(1,2,\ldots,n)} \Delta_{p_1p_2\ldots p_{n-k}} \sum_{q_1<q_2<\ldots q_k} \begin{vmatrix}
a_{q_1q_1} & a_{q_1q_2} & \cdots & a_{q_1q_k} \\
a_{q_2q_1} & a_{q_2q_2} & \cdots & a_{q_2q_k} \\
\vdots & \vdots & \ddots & \vdots \\
a_{q_kq_1} & a_{q_kq_2} & \cdots & a_{q_kq_k}
\end{vmatrix}
\]

\[
\sum_{p_1<p_2<\ldots<p_{n-k}} \{1,2,\ldots,n\} \sum_{p_1p_2\ldots p_{n-k}} \sum_{q_1<q_2<\ldots q_k} a_{q_1q_1} a_{q_1q_2} \cdots a_{q_1q_k}
\]

\[
\Delta_{\substack{p_1p_2\ldots p_{n-k} \quad q_1<q_2<\ldots q_k}}
\]

\[
\sum_{p_1<p_2<\ldots<p_{n-k}} \{1,2,\ldots,n\} \sum_{p_1p_2\ldots p_{n-k}} \sum_{q_1<q_2<\ldots q_k}
\]
12.2. Standard features of $\text{AMS-}\LaTeX$

$$\hdotsfor[2.0]{4}$$

$$a_{q_kq_1} & a_{q_kq_2} & \cdots & a_{q_kq_k}$$

Comparing this input text with that on page 134, one sees that it is simpler and easier to follow. The only new command used here is $\hdotsfor$ which has the syntax

$$\hdotsfor[\text{stretch}]{n}$$

and which prints a continuous line of dots through $n$ columns. The optional argument $\text{stretch}$ is a multiplicative number to increase the dot density, being 1.0 by default.

$$\begin{array}{cccc} a & b & c & d & e \\ x & \hdotsfor{3} & z \end{array}$$

Compare the standard dot spacing above with that from $\hdotsfor[2.0]$ in the previous example.

The initial letter of each of the $\text{xmatrix}$ environments indicates the type of braces that enclose it: $\text{pmatrix}$ for (round) parentheses, $\text{bmatrix}$ for (square) brackets, $\text{Bmatrix}$ for (curly) braces, $\text{vmatrix}$ for vertical lines, and $\text{Vmatrix}$ for double vertical lines. They appear as

$$\begin{pmatrix} r & s & t \\ u & v & w \\ x & y & z \end{pmatrix}$$

$$\begin{bmatrix} r & s & t \\ u & v & w \\ x & y & z \end{bmatrix}$$

$$\begin{Bmatrix} r & s & t \\ u & v & w \\ x & y & z \end{Bmatrix}$$

$$\begin{vmatrix} r & s & t \\ u & v & w \\ x & y & z \end{vmatrix}$$

$$\begin{Vmatrix} r & s & t \\ u & v & w \\ x & y & z \end{Vmatrix}$$

where each matrix has been produced with

$$\begin{array}{c} \begin{pmatrix} r & s & t \\ u & v & w \\ x & y & z \end{pmatrix} \\ \begin{bmatrix} r & s & t \\ u & v & w \\ x & y & z \end{bmatrix} \\ \begin{Bmatrix} r & s & t \\ u & v & w \\ x & y & z \end{Bmatrix} \\ \begin{vmatrix} r & s & t \\ u & v & w \\ x & y & z \end{vmatrix} \\ \begin{Vmatrix} r & s & t \\ u & v & w \\ x & y & z \end{Vmatrix} \end{array}$$

where $\text{xmatrix}$ is set to $\text{matrix}$, $\text{pmatrix}$, $\text{bmatrix}$, $\text{Bmatrix}$, $\text{vmatrix}$, and $\text{Vmatrix}$ one after the other.

To generate a small array within a text formula, one can apply the $\text{smallmatrix}$ environment. In this way $\left( \begin{smallmatrix} a & b & c \\ e & m & r \end{smallmatrix} \right)$ can be made with

$$\bigl( \begin{smallmatrix} a & b & c \\ e & m & r \end{smallmatrix} \bigr)$$

12.2.5 User extensions and fine adjustments

Function names

Package: $\text{amsopn}$ Standard $\LaTeX$ recognizes a number of preprogrammed function names (Section 5.3.8) that are printed in math made by placing a backslash in
front of that name: arccos, arccsin, arctan, arg, cos, cosh, cot, coth, csc, deg, det, dim, exp, gcd, hom, inf, ker, lg, lim, liminf, limsup, ln, log, max, min, Pr, sec, sin, sinh, sup, tan, tanh. Not only do these names appear in an upright font, as is required for function names, but the spacing with adjacent parts of the mathematical expression is adjusted automatically.

\( \text{AMS-\LaTeX} \) provides some more function names, as variations on the standard \( \lim \) name:

\begin{align*}
&\varlimsup \lim \quad \varinjlim \quad \varprojlim \\
&\varliminf \lim \quad \varprojlim 
\end{align*}

These functions may take on limits with the raising and lowering operators ^ and _; for example \( \varliminf_{n\to\infty} \) for \( \lim_{n\to\infty} \).

It is also possible to define new function names with the same font and spacing properties as the predefined ones. The command

\begin{verbatim}
\DeclareMathOperator{\cmd}{name}
\end{verbatim}

which may only be issued in the preamble, before \( \begin{document} \)\end{document} defines a command \( \cmd \) that prints the function name \( \text{name} \). For example, to define a function name \( \text{doit} \), give

\begin{verbatim}
\DeclareMathOperator{\doit}{doit}
\end{verbatim}

and then \( \$A=3\doit^2(B)\$ \) yields \( A = 3 \text{doit}^2(B) \). Note that superscripts and subscripts are printed beside the operator; if they are to be printed as limits, that is, above and below the operator in displayed math mode, use the *-form to define them. For example:

\begin{verbatim}
\DeclareMathOperator*{\Lim}{lim}
\end{verbatim}

\begin{verbatim}
[ \Lim_{n\to-\infty}\quad \Lim_{n\to+\infty}
\end{verbatim}

The \( \text{name} \) text need not be identical to the command name. In particular, it may contain special characters not allowed in command names.

Modulo expressions are printed in standard \LaTeX with \texttt{\bmod} and \texttt{\pmod} commands, and are complemented in \( \text{AMS-\LaTeX} \) by \texttt{\mod} and \texttt{\pod}. The possibilities are:

\begin{align*}
z \equiv x + y \mod n^2 & \quad z \equiv x + y \ \bmod\{n^2\} \\
z \equiv x + y \pmod{n^2} & \quad z \equiv x + y \ \pmod\{n^2\} \\
z \equiv x + y \mod n^2 & \quad z \equiv x + y \ \mod\{n^2\} \\
z \equiv x + y (n^2) & \quad z \equiv x + y \ \pod\{n^2\}
\end{align*}

The automatic parentheses are missing with \texttt{\mod}, while with \texttt{\pod} the name 'mod' is omitted. Furthermore, \texttt{\pmod} is redefined for text formulas to reduce the preceding space: \( \$y\pmod{a+b}\$ \): \( y \pmod{a+b} \).

The \texttt{\DeclareMathOperator} command and additional function names are defined in the \texttt{amsopn} package, which may be loaded on its own without \texttt{amsmath}. 
12.2. Standard features of $\mathcal{A}_M\LaTeX$

**Fine-tuning roots**

The positioning of an index to a root sign is not always ideal under standard $\LaTeX$. In $\sqrt[\beta]{k}$, for example, the $\beta$ could be somewhat higher and shifted slightly to the right. The $\mathcal{A}_M\LaTeX$ commands

\[
\leftroot{shift} \uproot{shift}
\]

cause such manual displacements, where $shift$ is a number specifying the size in small, internal units. Negative numbers represent a shift in the opposite direction. Compare the above standard result with that of $\sqrt[\leftroot{-1}\uproot{3}\beta]{k}$:

$\beta\sqrt{k}$.

The size of the root sign depends on its contents. If they hang below the baseline, the root sign extends lower down than for contents that have no depth. Note the differences between $\sqrt{x}, \sqrt{y},$ and $\sqrt{z}$. Some publishers want all root signs be at the same height, as $\sqrt{x} + \sqrt{y} + \sqrt{z}$. This is accomplished with the $\LaTeX$ command $\text{\texttt{\smash}}$ which places its argument in a box of zero height and depth. The $\mathcal{A}_M\LaTeX$ version of this command allows an optional argument $b$ or $t$ to zero only the depth or height, respectively. The above example is produced with $\sqrt{\smash[b]{y}}$. The option $b$ is taken because we want only the depth to be ignored, not the height of the letter $y$.

**Spacing adjustment**

With standard $\LaTeX$, there are a number of commands to fine-tune the spacing in a math formula (Section 5.5.1). These are $\$, $\,: \;$, $\quad$ and $\qquad$ for increasing amounts of positive spacing, and $\!$ for negative spacing. With the $\texttt{amsmath}$ package, the first three still exist, but may also be called with the more obvious names $\texttt{\text{\texttt{\thinspace}}}$, $\text{\texttt{\medspace}}$, and $\text{\texttt{\thickspace}}$. There is also $\text{\texttt{\negthinspace}}$ as an alias for $\!$.

The complete set of spacing commands are summarized in the table below taken from the $\mathcal{A}_M\LaTeX$ manual.

<table>
<thead>
<tr>
<th>Short form</th>
<th>Command name</th>
<th>Demo</th>
<th>Short form</th>
<th>Command name</th>
<th>Demo</th>
</tr>
</thead>
<tbody>
<tr>
<td>$,$</td>
<td>$\text{\texttt{\thinspace}}$</td>
<td>$!$</td>
<td>$!$</td>
<td>$\text{\texttt{\negthinspace}}$</td>
<td>$!$</td>
</tr>
<tr>
<td>$,,$</td>
<td>$\text{\texttt{\medspace}}$</td>
<td>$!$</td>
<td>$\text{\texttt{\negmedspace}}$</td>
<td>$!$</td>
<td></td>
</tr>
<tr>
<td>$,$</td>
<td>$\text{\texttt{\thickspace}}$</td>
<td>$!$</td>
<td>$\text{\texttt{\negthickspace}}$</td>
<td>$!$</td>
<td></td>
</tr>
<tr>
<td>$\quad$</td>
<td>$!$</td>
<td>$!$</td>
<td>$!$</td>
<td>$\text{\texttt{\negthickspace}}$</td>
<td>$!$</td>
</tr>
<tr>
<td>$\qquad$</td>
<td>$!$</td>
<td>$!$</td>
<td>$!$</td>
<td>$\text{\texttt{\negthickspace}}$</td>
<td>$!$</td>
</tr>
</tbody>
</table>

The general math spacing command is

\[
\text{\texttt{\mspace{mu}}}
\]

which inserts space in mathematical spacing units ‘mu’ (1/18 em). For example, $\text{\texttt{\mspace{-9mu}}}$ puts in negative spacing of 1/2 em.
Chapter 12. Math Extensions with $\textit{AMS-\LaTeX}$

Vertical bars

In standard $\LaTeX$, the commands $|$ and $\backslash|\$ are used for single and double vertical bars, $|$ and $\parallel$. However, these symbols are often used as delimiters (that is, like braces) in which case different spacing requirements are needed. In particular, a distinction must be made between the left and right delimiter in expressions like $|a|$ and $\parallel v\parallel$. The $\LaTeX$ commands are only appropriate for single appearances, like $p|q$ or $f(t,x)|_{t=0}$.

The $\texttt{amsmath}$ package defines the delimiter commands $\backslash lvert$, $\backslash rvert$ for a single bar, and $\backslash lVert$, $\backslash rVert$ for a double bar. They are useful for defining commands that set their arguments in such delimiters, as

\begin{verbatim}
\newcommand{\abs}[1]{\lvert#1\rvert}
\newcommand{\norm}[1]{\lVert#1\rVert}
\end{verbatim}

Now $\abs{a}$ produces $|a|$, and $\norm{v}$ produces $\parallel v\parallel$.

A similar recommendation can be made for the standard commands $\langle$ and $\rangle$. By defining

\begin{verbatim}
\newcommand{\mean}[1]{\langle#1\rangle}
\end{verbatim}

one gives $\mean{x}$ to generate $\langle x \rangle$, rather than $\langle x \rangle$ which produces $< x >$.

Boxed formulas

A formula may be placed in a box with the command

\begin{verbatim}
boxed{formula}
\end{verbatim}

For example,

\[
\boxed{\int_0^\infty f(x) \, \dif x \approx \sum_{i=1}^n w_i e^{x_i} f(x_i)}
\]

produces

\[
\int_0^\infty f(x) \, \dif x \approx \sum_{i=1}^n w_i e^{x_i} f(x_i)
\]

12.2.6 Multiline equations

Equations consisting of several lines which are horizontally aligned at set points, such as the equals sign, can be generated in standard $\LaTeX$ with the $\texttt{eqnarray}$ and $\texttt{eqnarray*}$ environments (Section 5.4.7). Many authors consider these to be far too limited for publications with complicated multiline equations. $\textit{AMS-\LaTeX}$ therefore provides a range of further alignment environments for formulas extending over a single line:

\begin{verbatim}
align gather falign multline alignat split
\end{verbatim}
With the exception of split, all exist in a standard and a *-form. As for eqnarray, the standard form adds an automatic equation number to each line, while the *-form does not. The standard \LaTeX equation environment for single line formulas is also available in \AMS-L\LaTeX in a *-form. It may be used in combination with multiline environments.

**Common features of alignment environments**

All the alignment, or multiline, environments switch to math mode at the start and back to text mode at the end, except for split which must be called in math mode. A new line is forced in the formula with the \verb|\| command, as usual; an optional argument \verb|\[len]| can be added to increase the line spacing by \textit{len}, again as usual.

The automatic numbering with the standard forms can be suppressed for single lines by adding \verb|\notag| before the \verb|\| line break. Alternatively, the line can be given a desired marker with \verb|\tag{mark}|. For example, with \verb|\tag{$\dag$}|, the marker is ($\dag$). Using the *-form instead, the marker text is printed without the parentheses.

The vertical position of the equation number or marker can be adjusted with the command

\verb|\raisetag{len}|

which moves the marker upwards by \textit{len} for that line only. A negative value moves it downwards.

**The multline environment**

The multline environment is a variant of the equation environment for single formulas that are too long for one line. The line breaks occur where the user forces them with the \verb|\| command. The first line is left justified, the last right justified, and lines in between are centered. However, if the option fleqn has been given, all the lines appear left justified.

The equation number, if present, appears at the right of the last line by default or if the option reqno has been selected; if the option leqno has been chosen, the number is placed at the left of the first line. (See Section 12.2.8 for the amsmath options.)

It is possible to shift individual lines fully to the left or right with the commands \verb|\shoveleft{formula}| and \verb|\shoveright{formula}|. The entire formula text for that line, except the terminating \verb|\|, is placed in their arguments.

The left and right margins for the formula are set by the length parameter \verb|\multlinegap| which is initially 10 pt. This may be altered by the user with the \verb|\setlength| or \verb|\addtolength| commands.
An equation with five lines could be broken to look as follows:

First line — left justified
Second line — horizontally centered
Third line — pushed to the left
Fourth line — pushed to the right
Last line — right justified

\begin{multline}
\framebox[.75\textwidth]{First line --- left justified} \\
\framebox[.6\textwidth]{Second line --- horizontally centered} \\
\shoveleft{\framebox[.6\textwidth]{Third line --- pushed to the left}} \\
\shoveright{\framebox[.6\textwidth]{Fourth line --- pushed to the right}} \\
\framebox[.75\textwidth]{Last line --- right justified}
\end{multline}

A real equation would contain mathematical expressions and not the \framebox commands in the above demonstration.

The \texttt{split} environment

Like \texttt{multiline}, the \texttt{split} environment is meant for a single equation that does not fit on one line. Line breaks are again forced with the \\ command; the difference is that in each line there is an alignment marker & such that the lines are horizontally positioned to line up the markers.

The \texttt{split} environment does not switch into math mode, nor does it produce an equation number. It is intended to be applied within another math environment, such as \texttt{equation} or \texttt{gather}. This is why there is an \texttt{equation*} environment in \texttt{AMS-\LaTeX}.

The equation number, if present, is provided by the outer environment. It is applied to the entire multiline formula, which by default, or with the option \texttt{centertags}, is centered on the group of lines. With the option \texttt{tbtags}, it is placed either at the left of the last line, or at the right of the first line, depending on the further options \texttt{leqno} and \texttt{reqno}, respectively. (See Section 12.2.8.)

\begin{equation}
H_c = \frac{1}{2n} \sum_{l=0}^{n} (-1)^l (k-l)^{p-2} \sum_{l_1, \ldots, l_p = l}^{p} \prod_{i=1}^{p} \left( \frac{n_i}{l_i} \right) \\
\times [(k-l) - (k_i - l_i)]^{k_i - l_i} \times [(k-l)^2 - \sum_{j=1}^{p} (k_i - l_i)^2]
\end{equation}

(12.2)
\begin{equation}\begin{split}
H_c&=\frac{1}{2n}\sum_{l=0}^n (-1)^l (k-l)^{p-2}
\sum_{l_1+\ldots+l_p=l} \prod_{i=1}^p \binom{n_i}{l_i}
&\quad\times[(k-l) - (k_i-l_i)]^{k_i-l_i}
&\quad\times\left[(k-l)^2 - \sum_{j=1}^p (k_i-l_i)^2\right]
\end{split}\end{equation}

The alignment has been chosen to be just before the equals sign. The second line begins with the alignment marker & so that the equation continues below the = in the first line. A \quad has been inserted so that the × is not immediately below the equals sign. Alternatively, one could place the & after the = and dispense with \quad in the second line. However, the above is more suitable when there are continuation lines beginning with =. Note the centered equation number at the right.

**The gather environment**

The gather environment switches to math mode, centering each of its formula lines without any alignment. The formula lines are separated by \\ commands. Each line receives an equation number, unless the *-form has been used, or \notag has been issued in that line.

\begin{gather}
\frac{1}{2} + \left(\frac{2}{3}\right)^4 + \left(\frac{3}{4}\right)^9 + \ldots + \left(\frac{n}{n+1}\right)^{n^2} + \ldots = \sum_{n=1}^\infty \left(\frac{n}{n+1}\right)^{n^2} \\
\text{converges since} \quad \lim_{n\to\infty} \sqrt[n]{\left(\frac{n}{n+1}\right)^{n^2}} = \lim_{n\to\infty} \left(\frac{1}{1 + \frac{1}{n}}\right)^n = \frac{1}{e} < 1 \\
\tag*{root condition}
\end{gather}

\begin{gather}
2 + \frac{3}{4} + \frac{4}{9} + \ldots + \frac{n+1}{n^2} + \ldots = \sum_{n=1}^\infty \frac{n+1}{n^2} \\
\text{diverges since} \quad \int_c^\infty \frac{x+1}{x^2} \, dx = \left[\ln x - \frac{1}{x}\right]_c^\infty = \infty \quad \text{(integral condition)}
\end{gather}
\[ \frac{d x}{dx} = \left[ \ln x - \frac{1}{x} \right] \bigg|_{c}^{\infty} = \infty \tag{integral condition} \]

Note the use of \texttt{tag} and \texttt{notag} (page 271) to add text as markers to the second and fourth lines.

**The \texttt{align} environment**

The \texttt{align} environment is intended for multiple equations with horizontal alignment, usually on an equals sign or equivalent. New lines are indicated with `\\` as usual. Each line is split into aligned columns such that the first column is right justified against the `&` character, the second left justified; the third column is right justified against the third `&`, the fourth column left justified again, and so on. This is the same as an \texttt{array} environment with column specification `{rl rl rl ...}`.

\[
\begin{align*}
(x^n)' &= nx^{n-1} & (e^x)' &= e^x & (\sin x)' &= \cos x \\
\left(\frac{1}{x^n}\right)' &= -\frac{n}{x^{n+1}} & (a^x)' &= a^x \ln a & (\cos x)' &= -\sin x \\
\left(\sqrt[n]{x}\right)' &= \frac{1}{n \sqrt[n]{x^{n-1}}} & (\ln x)' &= \frac{1}{x} & (\tan x)' &= \frac{1}{\cos^2 x} \\
(\log_a x)' &= \frac{1}{x \ln a} & (\cot x)' &= -\frac{1}{\sin^2 x} \\
\end{align*}
\]

The \texttt{align*} environment is used here to prevent the lines from being numbered, something that is not appropriate for such a collection of formulas. The alignment within each of the three column pairs is on the equals sign. The input for the last line begins with a double `&&` to produce an empty column pair.

Occasionally a set of formulas is to be aligned on several equals signs in one line, as in the equations below for the volume \( V \), inertial moment \( I_z \), and mass \( M \) of an arbitrary body, in Cartesian, cylindrical, and spherical coordinates. In this case, the second and third parts are separated by a
double \&\& so that the left-hand sides of these column pairs are empty: the equals signs are always on an odd-numbered alignment marker.

\[ V = \int_V \dif v = \iiint \dif x \dif y \dif z = \iiint \rho \dif x \dif \rho \dif \phi \]
\[ I_z = \int_V \rho^2 \dif v = \iiint (x^2 + y^2) \dif x \dif y \dif z = \iiint \rho^3 \dif z \dif \rho \dif \phi \]
\[ M = \int_V \delta \dif v = \iiint \delta \dif x \dif y \dif z = \iiint \delta \rho \dif z \dif \rho \dif \phi \]

\begin{align}
V &= \int\limits_V \dif v \quad \&= \iiint \dif x \\dif y \\dif z \\
&\quad \&= \iiint \rho \dif x \\dif \rho \dif \phi \\
&\quad \&= \iiint r^2 \sin \theta \dif r \dif \theta \dif \phi \\
I_z &= \int\limits_V \rho^2 \dif v \quad \&= \iiint (x^2 + y^2) \dif x \dif y \dif z \\
&\quad \&= \iiint \rho^3 \dif z \dif \rho \dif \phi \\
&\quad \&= \iiint r^4 \sin^3 \theta \dif r \dif \theta \dif \phi \\
M &= \int\limits_V \delta \dif v \quad \&= \iiint \delta \dif x \\dif y \\dif z \\
&\quad \&= \iiint \delta \rho \dif z \dif \rho \dif \phi \\
&\quad \&= \iiint \delta r^2 \sin \theta \dif r \dif \theta \dif \phi
\end{align}

There are two variations on the align environment, \texttt{falign} and \texttt{alignat}. The first has exactly the same syntax as align but it inserts so much spacing between the column pairs that the entire line is filled out. The alignat environment is just the opposite: no spacing is inserted automatically between the column pairs. It must take the number of column pairs as a mandatory argument, but otherwise the syntax of the contents is the same as that for align. The example above with the volume, inertial moment, and mass of a body could just as well have been given with

\begin{alignat}{3}
V &= \int\limits_V \dif v \quad \&= \iiint \dif x \\dif y \\dif z \\
&\quad \&= \iiint \rho \dif x \\dif \rho \dif \phi \\
&\quad \&= \iiint r^2 \sin \theta \dif r \dif \theta \dif \phi \\
I_z &= \int\limits_V \rho^2 \dif v \quad \&= \iiint (x^2 + y^2) \dif x \dif y \dif z \\
&\quad \&= \iiint \rho^3 \dif z \dif \rho \dif \phi \\
&\quad \&= \iiint r^4 \sin^3 \theta \dif r \dif \theta \dif \phi \\
M &= \int\limits_V \delta \dif v \quad \&= \iiint \delta \dif x \\dif y \\dif z \\
&\quad \&= \iiint \delta \rho \dif z \dif \rho \dif \phi \\
&\quad \&= \iiint \delta r^2 \sin \theta \dif r \dif \theta \dif \phi
\end{alignat}

(In fact, this is precisely what was done in order to fit it within the line width of this book.)

If one or more columns are empty, as in this example, it is possible to control their widths precisely in the alignat environment by adding
explicit spacing between the two & characters in one of the lines. See Section 5.5.1 for spacing in math mode.

In summary, for the align environment and its variants, the first, third, fifth, etc. & characters are alignment markers, while the second, fourth, sixth, etc. are column pair separators.

**Nested alignment environments**

We have already pointed out on page 272 how the split environment is to be placed inside an equation environment. The same is true for the environments aligned and gathered, which may be used as building blocks within formulas. Their contents and behavior are otherwise the same as their related environments.

Both of these environments take an optional argument `pos`

\begin{aligned}
\alpha &= aa \\
\beta &= bbbbbb \\
\gamma &= g
\end{aligned}

\begin{aligned}
\delta &= dd \\
\eta &= eeeeee \\
\varphi &= f
\end{aligned}

which takes values of t or b to determine the vertical alignment (top or bottom) when they appear beside other elements. When no `pos` is given, they are centered. In this way

\begin{equation*}
\begin{aligned}
\alpha &= aa \\
\beta &= bbbbbb \\
\gamma &= g
\end{aligned} \qquad \text{versus} \qquad \\
\begin{aligned}
\delta &= dd \\
\eta &= eeeeee \\
\varphi &= f
\end{aligned}
\end{equation*}

is produced with

\begin{verbatim}
\begin{equation*}
\begin{align*}
\alpha &= aa \\
\beta &= bbbbbb \\
\gamma &= g
\end{align*} \qquad \text{versus} \qquad \\
\begin{align*}
\delta &= dd \\
\eta &= eeeeee \\
\varphi &= f
\end{align*}
\end{equation*}
\end{verbatim}

**The cases environment**

Although it is possible with standard \LaTeX\ to produce structures of the form

\[ P_{r-j} = \begin{cases} 
0 & \text{if } r-j \text{ is odd}, \\
\binom{r-j}{j}(-1)^{r-j}/2 & \text{if } r-j \text{ is even.}
\end{cases} \quad (12.8) \]
as demonstrated by a similar example in Section 5.4.1 on page 132, the \texttt{AMS-LATEX} cases environment allows a simpler input:

\begin{equation}
P_{r-j} = \begin{cases} 0 & \text{if } r-j \text{ is odd,} \\
   r! \cdot (-1)^{(r-j)/2} & \text{if } r-j \text{ is even.}
\end{cases}
\end{equation}

There may be more than two cases in the environment, as in the example reproduced here from page 132:

\[
y = \begin{cases} -1 : & x < 0 \\
   0 : & x = 0 \\
   +1 : & x > 0
\end{cases}
\]

12.2.7 Equation numbering

**Numbering hierarchy**

With the standard \texttt{LATEX} classes \texttt{book} and \texttt{report}, equations are given a double number with the chapter designation and then a sequential number starting at 1 for each new chapter. For the \texttt{article} class, the equations are numbered sequentially throughout the work.

With the \texttt{amsmath} package, it is possible to alter this hierarchy. For example, if an article is to have the equations numbered within each section, with the section number, give

\texttt{\numberwithin{equation}{section}}

to redefine the equation numbers to include the section number and to make the equation counter reset every time the section counter is incremented. This is as though the equation counter had been created with \texttt{\newcounter{equation}[section]} (Section 8.1.2), something which the user cannot normally bring about. Furthermore, \texttt{\theequation} is redefined to be \texttt{\thesection.\arabic{equation}}, something that is in the user’s power but which is not much use if the equation counter is never reset.

**Subnumbering equations**

On page 190 we give an example of how equations may be subnumbered, that is, the main equation number stays the same and a letter is appended to it, as 1.8a, 1.8b, 1.8c .... The \texttt{amsmath} package provides this feature with the \texttt{subequations} environment. Numbered equations appearing within

\begin{subequations}
\end{subequations}
will all have the same main number which is one more than that of the previous one, with sequential, lower case letters attached.

Within the environment, the equation counter refers to the subnumber, that is, to the letters, while the main number is to be found in the parentequation counter. To change the format of the subnumber, say to 1.8-A, 1.8-B,..., give

\begin{subequations}
\renewcommand{\theequation}{\theparentequation-\Alph{equation}}
\end{subequations}

Referencing equation numbers

The \LaTeX{} cross-reference system is described in Section 9.2.1 and works exactly the same way with $\text{AMS-LATeX}$: when \texttt{\label{marker}} is issued in a mathematical formula that receives an automatic equation number, that number can be printed anywhere in the text with \texttt{\ref{marker}}, where \texttt{marker} is arbitrary text to identify that equation.

The \texttt{amsmath} package adds a command \texttt{\eqref{marker}} to print the equation number in parentheses, as it appears beside the math formula. For example, the cases equation on page 276 is referred to as equation 12.8 with \texttt{\ref}, or as equation (12.8) with \texttt{\eqref}.

If \texttt{\label} is given immediately after the start of a \texttt{subequations} environment, the corresponding \texttt{\ref} commands will print the main equation number without the extra letter. In this way one can refer to the entire group of equations. Later \texttt{\label} commands are associated with individual equation lines and reference them with the letters.

Page breaks within multiline formulas

Unlike the standard \LaTeX{} \texttt{eqnarray} environment, the $\text{AMS-LATeX}$ multiline math environments do not normally allow any page breaks to occur within them. The idea is that the author should have more control over where such breaks may occur. To allow or force a page break within a multiline equation, one gives

\texttt{\displaybreak[\textit{num}]}

just before the line breaking command \texttt{\}. Here the optional \textit{num} has the same meaning as for the standard \texttt{\pagebreak} command (page 33): without it, a new page is forced, but it may take values of 0–4 to allow a break with increasing degree of encouragement, whereby 4 also forces the page break.

Alternatively, one can issue \texttt{\allowdisplaybreaks} in the document preamble to allow \LaTeX{} to break pages automatically within multiline
formulas as necessary. This command too takes an optional argument `num` with possible values between 0 and 4 which make it progressively easier for automatic page breaks to occur.

Once `\allowdisplaybreaks` has been given in the preamble, it is still possible to suppress page breaks within a formula by ending an equation line with `\*` instead of with `\`.

### 12.2.8 Package options for `amsmath`

The main `amsmath` package for `AMS-\LaTeX` recognizes a number of options that may be given when it is loaded with

```
\usepackage[options]{amsmath}
```

They are listed here as pairs with opposing effects. The member of each pair that is assumed if neither is given, the default, is indicated by underlining.

**centertags | tbtags** The equation number for a `split` environment (page 272) is centered vertically by default. With `tbtags`, it is placed either to the left of the first line or to the right of the last line, depending on the side on which numbers are to appear.

**sumlimits | nosumlimits** In displayed formulas, initial and final limits appear below and above the `\sum` sign with the `sumlimits` option. With `nosumlimits`, they are placed beside the sign, raised and lowered with the usual `^` and `_` characters.

Other symbols that are affected by these options are `\prod \coprod \bigcap \bigsqcup \bigvee \bigwedge \bigodot \bigotimes` and `\bigoplus`. On the other hand, integral signs are not influenced by them.

**sumlimits**

\[
\sum_{n=0}^\infty \frac{1}{2^n} = 2; \prod_{i=0}^{m-1} n - i = \frac{n!}{(n-m)!}
\]

**nosumlimits**

\[
\sum_{n=0}^\infty \frac{1}{2^n} = 2; \prod_{i=0}^{m-1} n - i = \frac{n!}{(n-m)!}
\]

The input text is the same for both of the above cases.

**intlimits | nointlimits** Integral signs normally have their limits at the side; these options allow them to be placed above and below as for summations.

**intlimits**

\[
\int_0^a \sqrt{a^2 - x^2} \, dx = \int_0^1 a^2 \sqrt{1 - \sin^2 t} \, d \sin t = a^2 \int_0^{\pi/2} \cos^2 t \, dt = \frac{\pi a^2}{4}
\]
\[
\int_0^a \sqrt{a^2 - x^2} \, dx = \int_0^1 a^2 \sqrt{1 - \sin^2 t} \, d\sin t = a^2 \int_0^{\pi/2} \cos^2 t \, dt = \frac{\pi a^2}{4}
\]

where again the input text is the same for both cases. As a reminder, the standard \LaTeX treatment of limits on integral signs is the same as for \texttt{nointlimits}.

\texttt{namelimits} | \texttt{nonamelimits} The function names \texttt{\det}, \texttt{\gcd}, \texttt{\inf}, \texttt{\lim}, \texttt{\liminf}, \texttt{\limsup}, \texttt{\max}, \texttt{\min}, \texttt{\Pr}, and \texttt{\sup} frequently take lower limits which normally appear below the name in displayed formulas. With \texttt{nonamelimits}, they are placed at the lower right.

\begin{align*}
\texttt{namelimits} & \quad \lim_{x \to \infty} \left(1 + \frac{1}{x}\right)^x = e = 2.7182\ldots \\
\texttt{nonamelimits} & \quad \lim_{x \to \infty} \left(1 + \frac{1}{x}\right)^x = e = 2.7182\ldots
\end{align*}

The above options determine the standard placement of limits for the entire document. It is still possible to change the behavior in any particular case with the \texttt{\limits} and \texttt{\nolimits} commands, as in normal \LaTeX.

The remaining package options select the side for equation numbers and the horizontal positioning of equations.

\texttt{1eqno} | \texttt{reqno} The standard location for equation numbers is on the right side, at the margin; with \texttt{1eqno}, they are placed on the left side of the equation.

\texttt{fleqn} With this option, all displayed equations are printed flush left, set off from the left margin by an amount \texttt{\mathindent}. Without this option, equations are centered. (This is like the \texttt{fleqn} class option for the standard classes, page 39.)

### 12.3 Further \AMSLaTEX packages

The \AMSLaTEX packages described in this section must be loaded explicitly if their features are to be exploited. Unlike the \texttt{amsbsy} and \texttt{amstext} packages, they are not loaded automatically with \texttt{amsmath}. They may, however, be used on their own, independently of the main package.

#### 12.3.1 Extended theorem declarations

\texttt{Package: \texttt{amsthm}} The \texttt{amsthm} package offers many additional possibilities for generating theorem-like declarations described for standard \LaTeX in Section 4.5.

As for standard \LaTeX, a new theorem declaration is created with a statement like
\newtheorem{com}{Comment}

where the first argument is the name of the theorem type (here \texttt{com}) and
the second is the title that is printed when the theorem declaration is
invoked. For example,

\begin{com}
  Theorem declarations can have any name.
\end{com}

produces the declaratory text

\textbf{Comment 1.} \textit{Theorem declarations can have any name.}

In addition to the two mandatory arguments, the \texttt{\newtheorem} com-
mand may have one of two optional ones. The complete syntax is

\newtheorem{\texttt{type}}\[\texttt{num}\_\texttt{like}]\{\texttt{title}\}
or
\newtheorem{\texttt{type}}\{\texttt{title}\}[\texttt{in}\_\texttt{counter}]

where \texttt{num}\_\texttt{like} is the name of an existing theorem-like declaration which
is to be numbered in the same sequence as \texttt{type}, and \texttt{in}\_\texttt{counter} is a
counter name like \texttt{chapter} or \texttt{section} to reset the numbers of the \texttt{type}
declarations.

All this is standard \LaTeX so far. The \texttt{amsthm} package adds the following
features.

\begin{itemize}
  \item A \texttt{\newtheorem*} is provided that defines an unnumbered theorem
        structure.
  \item Three predefined theorem styles are available:
        \begin{itemize}
          \item \texttt{plain} in which the title and number are in bold face and the text
                italic;
          \item \texttt{definition} with title and number in bold face and the text in
                normal font;
          \item \texttt{remark} for title and number in italic and the text normal.
        \end{itemize}
  \item The desired style is activated by first issuing \texttt{\theoremstyle{style}};
        all subsequent \texttt{\newtheorem} statements will have this style until a
        new one is activated.
  \item A \texttt{\swapnumbers} can be issued to cause all following new theorem
types to have the numbers appear before the title, as 1 \textbf{Comment}.
  \item New theorem styles may be defined by the user by means of a
        \texttt{\newtheoremstyle} command, or additional predefined styles may
        be loaded with package options. Since this is fairly specialized and
        complex, it is best to examine the example file \texttt{thmtest.tex} or read
        the documentation in \texttt{amsthm.dtx}.
\end{itemize}
• A proof environment is available for presenting short proofs. It is an unnumbered structure with the title Proof. The text is terminated automatically with the Q.E.D. symbol □. This symbol may be altered by redefining the command \qedsymbol; it may be printed at any time by issuing \qed.

The amsthm package has much in common with Frank Mittelbach's theorem package in the tools collection of Section B.5.4.

12.3.2 Commutative diagrams

Package: amscd

The extra \AMS-L\TeX package amscd makes it easier to generate commutative diagrams like the one here at the right.

\[
\begin{CD}
S^\mathcal{W}_\Lambda \otimes T @> j >> T \\
( S \otimes T )/I @= ( Z \otimes T )/J \\
\end{CD}
\]

These diagrams are created within the CD environment using some additional arrow commands. These bear the rather unusual names @>>> @<<< @AAA and @VVV for arrows pointing right, left, upwards, and downwards, respectively. The command @= draws a horizontal double rule, a lengthened equals sign.

Any text or symbols between the first and second > or < characters will appear above the horizontal arrow in \scriptstyle font. Similarly, any text or symbols between the second and third characters will be printed below the arrow.

For vertical arrows, text or symbols between the first and second A or V are placed to the left; those between the second and third to the right, again in \scriptstyle.

The above example diagram, which is taken from the \AMS-L\TeX manual amsldoc.tex, was produced with

\[
\begin{CD}
S^\mathcal{W}_\Lambda \otimes T @> j >> T \\
( S \otimes T )/I @= ( Z \otimes T )/J \\
\end{CD}
\]

The command \End to print the function name ‘End’ is not standard. It must be previously defined with \DeclareMathOperator{\End}{End} (see page 268).

12.3.3 References with upref package

Package: upref

Normally the numbers printed with the \ref and \pageref commands are in the current font, whether that be bold, italic, or upright. In order to ensure that the numbers are always upright, load the extra \AMS-L\TeX package upref.
12.4 The $\mathcal{AM}$S fonts

The $\mathcal{AM}$S makes a number of fonts available to complement the regular Computer Modern fonts provided with the standard \TeX/\LaTeX installation. They include extra math alphabets, supplemental CM bold math italic and symbol fonts in smaller sizes than 10 pt, Cyrillic fonts, and additional symbol fonts.

In the next sections we describe these various fonts and how to take advantage of them.

12.4.1 Extra CM math fonts

Standard \TeX installations of Computer Modern fonts provide bold math italic \texttt{cmmib10}, the bold symbols \texttt{cmbsy10}, and math extensions \texttt{cmex10} fonts only in 10 pt size, as indicated by the suffix 10 to their names. The $\mathcal{AM}$S has supplemented these with versions in sizes 5–9 pt.

\begin{verbatim}
cmmib5  cmmib6  cmmib7  cmmib8  cmmib9  
cmbsy5  cmbsy6  cmbsy7  cmbsy8  cmbsy9  
cmex7   cmex8   cmex9
\end{verbatim}

The small caps font \texttt{cmcsc10} is also given companions \texttt{cmcsc8} and \texttt{cmcsc9}.

The normal \TeX installation automatically assumes that these fonts are on the system and incorporates them into the necessary NFSS font definition files. Substitutions will be made if they are missing.

12.4.2 Cyrillic fonts

The $\mathcal{AM}$S Cyrillic fonts were originally used in reviews of books published in Russian and other Slavic languages in which the titles were to be rendered in the original language. In 1988, the Humanities and Arts Computing Center of the University of Washington redesigned them for general purpose Slavic studies, adding the pre-Revolutionary and accented letters. The overall appearance was also greatly enhanced (see Layout 8 on page 497).

These fonts all bear the prefix \texttt{wncy}, followed by the style designation \texttt{r} (upright), \texttt{b} (bold), \texttt{i} (italic), \texttt{sc} (small caps), or \texttt{ss} (upright sans serif), and the design size in points.

The best way to enable the Cyrillic fonts is simply to select font encoding \texttt{OT2} under NFSS, as illustrated in Section A.2 on page 371. The \texttt{.fd} (font definition) files for the Cyrillic fonts have been so set up that the other font attributes fully parallel those of the Latin fonts. This means \texttt{wncy10} has all the same attributes as \texttt{cmr10}, except for the encoding: family \texttt{cyr}, shape \texttt{n}, series \texttt{m}. (Of course, if the CM fonts are not the
current standard ones, it will require more than just selecting the OT2 encoding to activate these Cyrillic fonts.)

The layout of the Cyrillic fonts has been chosen in such a way that the input text may be entered following the regular English transliteration scheme. Thus Cyrillic C is in position 83 where Latin S is normally situated. Typing S when a Cyrillic font is active outputs the correct equivalent C. Numerals and punctuation are to be found in the standard locations and so may be typed in as usual. ‘Санкт-Петербург 10?’ is thus generated by \{\cyr Sankt-Peterburg 10?\}.

Since the Cyrillic alphabet possesses more letters than the Latin, many of them must be transliterated with multi-letter combinations. These are automatically programmed into the fonts using \TeX's ligature system. For example, Ch is treated as a ligature for symbol Ch just as fi is for \textit{fi} in a Latin font. This means that multi-letter transliterations are simply typed in. The input for ‘Хрущёв’ is {\cyr Khрущчев}, where Kh \rightarrow X and shch \rightarrow Ш. The transliteration scheme is that for English; other languages have their own systems to reproduce the original pronunciation. For example, ‘Горбачёв’ is \textit{Gorbatschow} in German, \textit{Gorbaciov} in Italian, and \textit{Gorbachev} in English. These other schemes do not work with these fonts.

Not all letters can be produced so automatically (for example, the \textit{ё} in Горбачёв) and for this reason the \textit{AMS} provides a file \texttt{cyracc.def} containing macro definitions for accented letters and other special features, such as the hard and soft signs. When these macros are given in a Latin font, additional transliteration symbols appear.

### 12.4.3 Euler fonts

The fonts known as the \textit{Euler} collection, named after the eighteenth-century mathematician Leonhard Euler, have been collected from various sources by the \textit{AMS}. They include a 'blackboard' font, representing a professor’s handwriting on a blackboard, a Fraktur or Gothic font, and a script font. Their main purpose in mathematics is to be a substitute for the CM calligraphic math alphabets.

To use the blackboard characters, one must load the package \texttt{amsfonts} (which is included automatically with the \texttt{amssymb} package) and then employ the math alphabet command \texttt{\mathbb}. Thus $\mathbb{A B C ..}$ produces

$$\text{abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ}$$

The package \texttt{eucal} redefines the \texttt{\mathcal} command to use the Euler script characters in place of the CM calligraphic letters. If this package is loaded with the option \texttt{mathscr}, the command \texttt{\mathcal} is left unchanged and instead \texttt{\mathscr} is defined to invoke these letters. In this case, $\mathscr{A B C ..}$ produces

$$\text{abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ}$$
Finally, the package \texttt{eufak} defines the math alphabet command \texttt{\mathfrak}, with which $\mathfrak{A B C ..}$ yields

A B C D E F G H I J K L N M O P Q R S T U V W X Y Z

This math alphabet is also enabled with the \texttt{amsfonts} package.

### 12.4.4 Extra math symbols

The set of symbols in the CM math symbol fonts by no means exhausts the fantasies of active mathematicians. To overcome this deficiency, the \texttt{AMS} has produced two fonts, \texttt{msam10} and \texttt{msbm10}, containing only symbols, arrows, and the blackboard characters. They are also available in sizes 5–9 pt. Since these fonts originated in the days when \TeX{} could only handle 128 characters in any font, that is exactly how many they contain. Today, they could be combined into one font of 256 characters.

Two packages permit access to this treasure trove of hieroglyphs:

\texttt{amsfonts} enables the \texttt{\mathbb} math alphabet command for the blackboard characters, and defines those symbol names which are otherwise only provided in the \texttt{latexsym} package (Section 5.3.3).

\texttt{amssymb} is the more convenient package, which loads \texttt{amsfonts} and then defines names for all the symbols in the two fonts.

For example,

\begin{verbatim}
\circlearrowright \textcircled{C} \cup \textcircled{\ll lessapprox} \lll \varpropto \because \circled{\vartriangle} \vDash \blacktriangle \sphericalangle \bigcirc \owns \leq \ll \propto \because \models \implies \blacklozenge \Diamond \blacklozenge
\end{verbatim}

All the symbols and their associated names from the \texttt{amssymb} package are to be found in Tables H.20–H.26 on pages 599–601.
The inclusion of graphical material from other programs is treated in Chapter 6. Such ‘foreign’ files do add some complications (which are greatly reduced today with \LaTeX\textsuperscript{2ε}) and can cause portability problems. What would be desirable is a means to do graphics with \LaTeX\ itself so that the source text file is fully self-contained.

Standard \LaTeX\ does contain the means to make somewhat primitive drawings on its own. The word ‘primitive’ should not be considered derogatory, for simple building blocks are the basic units for constructing very complicated, sophisticated structures. They are also useful for superimposing imported graphics or for adding embellishments to them, as demonstrated in Section 6.1.5.

This chapter describes the intrinsic \LaTeX\ drawing capabilities, and then explains some extensions that are available to enhance them.

Many more possibilities exist for adding specialized diagrams, from chemistry, to music, to chess positions. These are described in detail in The \LaTeX\ Graphics Companion by Goossens et al. (1997).

13.1 The picture environment

13.1.1 Picture coordinates

The picture building blocks can only be put in place once a coordinate system has been established for that picture. This consists of a reference point or origin, and two mutually perpendicular coordinate axes, as well as a length unit for the coordinates. The origin is the lower left corner of the picture and the axes are its lower and left edges. These edges are referred to as the x-axis (lower) and y-axis (left).

Once the unit of length (UL) has been specified, every point within the picture area can be uniquely referred to with two decimal numbers: the first is the number of length units along the x-axis, the second the number along the y-axis.
The coordinate numbers are generally positive, meaning that the point lies to the right and above the reference point. Since this point is the lower left corner of the picture, all other points should be more to the right and higher than it. However, negative values are also possible. A negative x value (a negative number for the first member of a coordinate pair) defines a point left of the origin, while a negative y value (the second member of the pair is negative) specifies a point below the reference point.

The unit length is selected with the command

\setlength{\unitlength}{length}

In the left-hand example above, the unit length was set to a value of 1.5 cm with the command \setlength{\unitlength}{1.5cm}. The point (1.8,1.4) then lies 1.8 times the unit length (= 2.7 cm) to the right, and 1.4 times (= 2.1 cm) above the origin. In the right-hand example, the unit length is set to 1 cm.

The unit of length is normally set to a convenient size such as 1 cm, 1 mm, or 1 in and the picture is built up accordingly. Once it has been completed, it is possible to rescale the whole thing simply by changing the value of the length unit. A picture that was originally designed with \unitlength set to 1 cm can be enlarged by a factor 1.2 by redefining its length unit to be 1.2 cm.

13.1.2 The picture environment

Pictures are constructed within the picture environment which is started with

\begin{picture}(x_dimen,y_dimen)
    picture_commands
\end{picture}
13.1. The picture environment

where \( (x_{\text{dimen}}, y_{\text{dimen}}) \) is a pair of numbers that specifies the size (dimensions) of the picture in the \( x\text{-direction} \) (horizontal) and \( y\text{-direction} \) (vertical). This pair of numbers is enclosed in \textit{round} parentheses! The unit of length is that previously selected by \texttt{\unitlength}.

\[
\setlength{\unitlength}{1.5cm}
\begin{picture}(4,5) \ldots \ldots \ldots \end{picture}
\]

produces a picture that is 4 length units wide and 5 units high. Since the length unit has been set to 1.5 cm, the actual size is 6 cm wide and 7.5 cm high.

\textit{Picture commands} are those commands described below that are used to produce and position the individual picture elements. These are the only commands that are allowed within the \texttt{picture} environment, other than font style and size declarations (Section 4.1) and the line thickness commands \texttt{\thicklines} and \texttt{\thinlines}. These last determine which of the two available line thicknesses will become current for drawing lines: one may switch back and forth as one pleases. Initially \textit{thin} lines are active.

The value of the parameter \texttt{\unitlength} must not be altered within the picture environment, for it must remain the same for the entire picture. It may, however, be changed between pictures.

If the \texttt{\unitlength} specification together with the \texttt{picture} environment are enclosed within another environment, such as \texttt{\begin{center} \ldots \end{center}}, then that value of \texttt{\unitlength} is valid only until the end of the environment. A picture environment without a preceding \texttt{\unitlength} command uses the standard value of 1 pt.

13.1.3 The positioning commands

Picture elements are generated and positioned by means of the two commands \texttt{\put} and \texttt{\multiput}, which have the syntaxes:

\[
\put(x,y)\{\textit{pic\_elem}\}
\multiput(x,y)(\Delta x,\Delta y)\{num\}\{\textit{pic\_elem}\}
\]

The \textit{pic\_elem} is one of the picture element commands described in the next section. The arguments \((x, y)\) are the placement coordinates, designating the location of the picture element within the picture coordinate system, in units of \texttt{\unitlength}. If this is 1 cm, then \((2.5, 3.6)\) means that the element is to be positioned 2.5 cm to the right and 3.6 cm above the lower left corner of the picture.

The \texttt{\multiput} command generates the same picture element \textit{num} times, moving it \((\Delta x, \Delta y)\) each time. Thus the element is drawn at

\[
(x, y), (x + \Delta x, y + \Delta y),
(x + 2\Delta x, y + 2\Delta y), \ldots \text{ up to }
(x + [num - 1]\Delta x, y + [num - 1]\Delta y)
\]
The coordinate pair \((x, y)\) is incremented by \((\Delta x, \Delta y)\) for each successive placement. The values of the incrementing pair may be positive or negative.

Thus \texttt{\multiput}(2.5,3.6)(0.5,\textendash0.6)\{pic\_elem\} produces the \texttt{pic\_elem} a total of five times, first at the location (2.5,3.6) and then at (3.0,3.0), (3.5,2.4), (4.0,1.8), and finally (4.5,1.2).

Note that the numbers for the coordinate and increment pairs are given in round parentheses \((,\)) and that the two numbers within each pair are separated by a comma. The \texttt{num} and \texttt{pic\_elem} entries, on the other hand, are enclosed in curly brackets \{\} as usual.

**Warning:** Since the comma separates the two numbers in a coordinate pair, it may not be used in place of a decimal point. For coordinate entries, decimal numbers must be written with a period, not a comma.

### 13.1.4 Picture element commands

**Text within pictures**

The simplest picture element of all is a piece of text, positioned at the desired location within the picture. This is accomplished by putting text in place of \texttt{pic\_elem} in the \texttt{\put} or \texttt{\multiput} command.

```
(1.8,1.2)
```

An arrow

The arrow points to the location (1.8,1.2). The command \texttt{\put(1.8,1.2){An arrow}} inserts the text 'An arrow' so that its lower left corner is at the specified position.

The text as picture element may also be packed into a \texttt{\parbox} or a \texttt{\minipage} environment, and the reference point for the coordinate entry in the \texttt{\put} command depends on the positioning arguments of that box:

- \texttt{\parbox[b]{..}{..}}
  - Reference point is the lower left corner of the last line in the parbox.

- \texttt{\parbox[t]{..}{..}}
  - Reference point is the lower left corner of the top line in the parbox.

- \texttt{\parbox[32mm]{..}{..}}
  - For a standard parbox, the reference point is the vertical center of the left edge.

**Exercise 13.1:** Produce a picture 100 mm wide and 50 mm high with \texttt{\unitlength} equal to 1 mm. Place the given texts at the following locations: (0,0) 'The First Picture', (90,47) 'upper left', (70,40) 'somewhere upper right', and put a \texttt{parbox} of width 60 mm at (25,25) containing 'A separate exercise file with the name picture.tex should be created for the exercises in this Chapter.'

**Exercise 13.2:** Repeat the picture processing with a value of 1.5 mm for \texttt{\unitlength}, and with positioning arguments \texttt{t} and \texttt{b} for the \texttt{parbox}.
13.1. The picture environment

**Picture boxes—rectangles**

The box commands \framebox, \makebox, and \savebox (Section 4.7.1) are available in the picture environment but with an extended syntax. In addition, there is another box command \dashbox:

\makebox\( (x_{\text{dimen}}, y_{\text{dimen}}) \)[pos]\{text\}
\framebox\( (x_{\text{dimen}}, y_{\text{dimen}}) \)[pos]\{text\}
\dashbox\( \text{dash}_{\text{len}} \)(x_{\text{dimen}}, y_{\text{dimen}})[pos]\{text\}

The *dimensional* pair \((x_{\text{dimen}}, y_{\text{dimen}})\) defines the width and height of the rectangular box in units of \unitlength. The positioning argument \texttt{pos} determines how the \texttt{text} is located within the box. It may take on values:

[t] top \quad The input text appears—centered horizontally—*below* the upper edge of the box.

[b] bottom \quad The input text appears—centered horizontally—*above* the lower edge of the box.

[l] left \quad The input text begins—centered vertically—at the left edge of the box.

[r] right \quad The input text ends—centered vertically—at the right edge of the box.

[s] stretch \quad The input text is stretched horizontally to fill up the box, and centered vertically.

Without the optional argument \texttt{pos}, the input text is centered vertically and horizontally within the box.

These positional values may be combined two at a time:

[tl] top left \quad The text appears at the upper left.

[tr] top right \quad The text appears at the upper right.

[bl] bottom left \quad The text appears at the lower left.

[br] bottom right \quad The text appears at the lower right.

The order of the values is unimportant: tl has the same effect as lt.

These box commands are to be used as \texttt{pic}._\texttt{elem} in the placement commands \texttt{\put} and \texttt{\multiput}. The box is so placed that its lower left corner is at the position given by the coordinate pair in the placement command.

\put(1.5,1.2){\framebox(2.5,1.2){\texttt{center}}}
The arrow indicates the point (1.5,1.2) which is the position of the lower left corner of the rectangle with width 2.5 units and height 1.2 units. The text 'center' is centered both horizontally and vertically. UL = 0.8 cm.

The effect of the text positioning argument is made clear with the following examples (UL = 1 cm):

- \put(0.0,1.95){\framebox(2,1.0)
  \[t\]{top center}}
- \put(3.0,1.95){\framebox(2,0.8)
  \[lb\]{bot. left}}
- \put(3.0,3.2){\framebox(2,0.6)
  \[r\]{center right}}
- \put(2.0,0.3){\framebox(2,0.6)
  \[s\]{stretch\hfill center}}

The picture element \makebox is exactly the same as the \framebox command but without the rectangular frame. It is most often employed with the dimensional pair (0,0) in order to place text at a desired location. (See Section 4.7.1 for the effect of zero width boxes on the enclosed text.)

- \put(3,1.6){\makebox(0,0){center center}}
- \put(2,0.5){\makebox(0,0)[tr]{top right}}
- \put(4,1.0){\makebox(0,0)[b]{bot center}}
- \put(2,2.8){\makebox(0,0)[l]{flush left}}

The combination [lb] positions the text in exactly the same way as simply typing the text in as pic.elem without a box, as shown on page 290.

The picture element \dashbox also produces a framed box, but with a dashed line around it. The argument dash_len specifies the dash length.  
\put(1.0,0.75){\dashbox{0.2}(4,1){dashed frame}}

A dashed frame looks best when both the height and width are a multiple of the dash length.

Even in the above picture box commands, the entered text may be put into a vertical box (\parbox or \minipage). Since vertical boxes themselves possess an optional positioning argument b or t, which must not conflict with that of the picture box, the following rule must be observed:

If a picture box command contains the positioning argument b or t, then the same value must be applied to the enclosed vertical box. If the picture box command has no positioning argument or only r or l, then the vertical box must be used in the standard (no argument) form.
The positioning argument of a picture box has the same effect on the enclosed vertical box as it does on a line of text, in that the whole box is treated as a single unit.

**Exercise 13.3:** Reproduce the organization table below with the boxes and included text but without the horizontal and vertical lines and arrows. These will be part of the next exercise.

*Hint:* first draw the boxes on a piece of squared paper so that the edges of the boxes lie on the printed rules. Select the unit of length to be the rule spacing of the paper. Take as the origin the lower left corner of an imaginary frame surrounding the entire diagram.

### Straight lines

In the picture environment, \LaTeX can draw straight lines of any length, horizontally and vertically as well as at a limited number of angles. The syntax for this picture element reads

\[
\texttt{\textbackslash line}(\Delta x, \Delta y)\{\texttt{length}\}
\]

For horizontal and vertical lines, *length* specifies how long the line is to be in length units. For lines at an angle, it has a somewhat more complicated meaning, as is explained below. The line begins at that spot given by the placement coordinates in the \texttt{put} or \texttt{multiput} command.

\[
\begin{array}{c}
\texttt{\textbackslash thicklines} \\
\texttt{\textbackslash put}(0,0)\{\texttt{\textbackslash line}(1,0)\{6\}\}
\end{array}
\]

The angle at which the line is drawn is given by the *slope pair* \((\Delta x, \Delta y)\). The slope pair \((1,0)\), in which \(\Delta x = 1\) and \(\Delta y = 0\), produces a horizontal line, while the pair \((0,1)\) leads to a vertical line. This is illustrated in the above example.

In general \((\Delta x, \Delta y)\) has the following meaning:
Beginning at a point A on the line and moving a distance $\Delta x$ in the x direction (horizontally), $\Delta y$ is the distance one must move in the y direction (vertically) in order to rejoin the line.

By specifying a slope pair $(\Delta x, \Delta y)$, a line is drawn at just that angle to fulfill the above conditions.

As mentioned already, the number of different slopes available is limited. This is because $\Delta x$ and $\Delta y$ may only take on values according to certain rules:

1. The number must be a whole integer (negative or positive).
2. Only the values 0, 1, \ldots, 6 are allowed.
3. The two numbers in the pair may not contain a common divisor.

Pairs such as $(3.5, 1.2)$ (rule 1) and $(7, 0)$ (rule 2) are thus forbidden. Similarly $(2, 2)$ and $(3, 6)$ are invalid pairs by rule 3, since both numbers in the first pair are divisible by 2, and those in the second by 3. The same angles are achieved with the pairs $(1, 1)$ and $(1, 2)$ respectively. In all there are 25 allowed slope pairs, including $(1, 0)$ for horizontal and $(0, 1)$ for vertical lines, as one can verify by writing down all the possibilities.

In addition, the numbers in the slope pair may be positive or negative, for example, $(0, -1), (-2, -5)$. A negative $\Delta x$ in the above diagram means motion to the left, and a negative $\Delta y$ is for motion downwards. Thus \texttt{\textbackslash put}(2,3)\{\texttt{\textbackslash line}(0, -1)\{2.5\}\} draws a line starting at point (2,3) and going 2.5 length units vertically downwards.

For lines at an angle, the argument \textit{length} determines the projected distance along the x-axis. This is made clearer with the help of the diagram on the left.

\texttt{\textbackslash put}(1.0,2.75)\{\texttt{\textbackslash line}(2,-1)\{3.5\}\}

If one draws dashed lines straight down from the two end points, then that part of the x-axis between them is the projection of the line on to the x-axis.

Sloping lines must have a minimum length of about 10 pt or 3.5 mm, otherwise a warning is issued and no line is drawn.

\textbf{Arrows}

The \textit{arrow} picture element is made with the command
\vector(\Delta x, \Delta y)\{\text{length}\}

which functions exactly the same as the \texttt{line} command as far as the arguments and their limitations are concerned. The command draws a line from the placement location given in the \texttt{put} or \texttt{multiput} command and places an arrow head at the end.

Just as for lines, arrows too must have a length of at least 10 pt or 3.5 mm. Rules 1–3 apply to \(\Delta x\) and \(\Delta y\) as well, with the further restriction that the allowed numbers are limited to 0, 1, \ldots, 4. This makes a total of 13 possible angles for arrows, not counting changes in sign.

\begin{picture}(5,2)\thicklines
  \put(5,0){\vector(-1,0){5}}
  \put(0,0){\vector(1,1){2}}
  \put(2,2){\vector(3,-2){3}}
\end{picture}

**Exercise 13.4:** Complete the diagram in Exercise 13.3 by including the missing horizontal and vertical lines and arrows.

**Exercise 13.5:** Generate the figure at the right. The corner points are (0,5), (0,10), (5,15), (10,15), (15,10), (15,5), (10,0), and (5,0) and the length unit is 0.1 inch.

**Circles**

The \texttt{circle} picture element is produced with the commands

\begin{verbatim}
\circle{\text{diameter}}
\circle*[\text{diameter}]
\end{verbatim}

With the *-form of the command, a solid filled-in circle is printed rather than just an outline as for the standard form. Only certain sizes are available so \LaTeX{} selects the one closest to the specified \texttt{diameter} entry. If the size is too small, a warning is output to the monitor and no circle is printed.

\begin{verbatim}
\begin{picture}(3,1.6)
  \put(1,1){\circle*[0.2]}
  \put(1,1){\circle{1.2}}
  \put(1,1){\vector(0,1){0.6}}
  \put(2.5,1){\circle*[0.5]}
\end{picture}
\end{verbatim}

The placement location in the corresponding \texttt{put} command refers to the center of the circle.
Ovals and rounded corners

The term *oval* is used here to mean a rectangle whose corners have been replaced by quarter circles; the largest possible radius is chosen for the circles such that the sides join together smoothly. The command to produce them is

\[ \text{\texttt{\textbackslash oval}}(x_{\text{dimen}}, y_{\text{dimen}})[\text{part}] \]

The placement location in the corresponding \texttt{\textbackslash put} command refers to the center of the oval.

\[ \text{\texttt{\textbackslash put}}(3.0, 0.75)\{\text{\texttt{\textbackslash oval}}(4.0, 1.5)\} \]

Here we have set the values $x_{\text{dimen}}=4.0$ \texttt{UL} and $y_{\text{dimen}}=1.5$ \texttt{UL}, whereby the unit length \texttt{UL} has been selected to be 0.8 cm. The center of the oval is at the placement coordinates in the \texttt{\textbackslash put} command, at $(3.0, 0.75)$.

The optional argument \texttt{part} takes on values $t$, $b$, $l$, or $r$, for making half ovals.

The width and height specifications for half ovals are always those of the entire figure even though only part of it is being drawn. Similarly, the placement coordinates in the corresponding \texttt{\textbackslash put} command refer to the center of the complete oval. (Unit length here is \texttt{UL} = 1 cm.)

The argument \texttt{part} may also be one of the combinations $tl$, $tr$, $bl$, or $br$ to generate a quarter oval. The order of the two letters is unimportant, so that $lt$, $rt$, $lb$, and $rb$ are equally valid.

Quarter and half circles may also be drawn as partial ovals with equal width and height, but only up to a certain size. The following examples
13.1. The picture environment

demonstrate that sections of circles are possible up to a size of about 1.5 cm.

Sections of ovals may be combined with other picture elements. The placement coordinates in the \put command, however, require some serious consideration to get them positioned properly.

In all the above examples, the centers of the ovals have been marked with a black dot to indicate where they are. They are not normally a part of the \oval picture element.

**Exercise 13.6:** Although the object pictured here is very offensive, it does make an excellent exercise for the picture environment. Hint: sizing and positioning can be worked out best by overlaying the illustration with transparent graph paper.

**Vertically stacked text**

It is sometimes necessary to write text vertically in a diagram, as in the example here at the right. This is carried out with the command

\shortstack[pos]{col}

The positioning argument can take on values l, r, or c. The standard is c for centered. The command is similar to a tabular environment with only one column. The text is entered as col, each row being separated from the next by \\.
vertically. The individual rows are separated from each other with the smallest possible vertical spacing. This means that rows with letters sticking up or down (like \textit{h} and \textit{y}) will have larger apparent gaps between them than rows without such letters. Adding \texttt{\strut} to each line, as in the third example, equalizes the vertical spacing.

\begin{tabular}{|c|c|c|c|c|}
\hline
This & Not & A strut & Letters & \\
spacing & really & in each & best & \\
leaves & the & line & \\
some & best & \\
room & \\
for & \\
help & at all & is better & \\
& \\
\hline
\end{tabular}

The placement coordinates of the corresponding \texttt{\put} command refer to the lower left corner of an imaginary box that contains the vertically stacked text. The first of the above texts is left justified, the second centered, and the third right justified. The last two on the right are centered. They were entered with

\begin{verbatim}
\put(1.0,0.5){\shortstack{l\{This\ spacing\ leaves\ some\ ...\}}}
\put(3.0,0.5){\shortstack{Not\ really\ the\ best\ \ ...\}}
\put(5.0,0.5){\shortstack{A strut\ \strut in each\ ...\}}
\put(8.0,0.8){\shortstack{L\ \e\ t\ t\ e\ r\ s}}
\put(9.0,1.2){\shortstack{b\ e\ s\ t}}
\end{verbatim}

The \texttt{\shortstack} command may also be used outside of the \texttt{picture} environment within normal text. One possible application is for marginal notes, as in Section 4.10.5.

\textbf{Framed text}

The \texttt{\framebox} command generates a frame of a predetermined size in which text may be inserted at various positions (page 291). In text mode, there is the command \texttt{\fbox} for drawing a frame around text that fits it exactly (Section 4.7.1). This command is also available in the \texttt{picture} environment.

The amount of spacing between the box frame and the enclosed text is given by the parameter \texttt{\fboxsep}. The placement of an \texttt{\fbox} by means of the \texttt{\put} command occurs in an unexpected manner, as shown below:

\begin{verbatim}
\begin{picture}(5,2)
\setlength{\fboxsep}{0.25cm}
\put(0,0){\framebox(5,2){}}
\put(1,1){\fbox{fitted frame}}
\end{picture}
\end{verbatim}

The additional frame spacing is often unwanted in a diagram, especially if the frame surrounds a picture object rather than text. In this case, the command
13.1. The picture environment

\frame{pic.elem}

is used instead. The placement coordinate of the \put command then refers to the lower left corner as usual.

\put(0.0,0.5){\frame{TEXT}}
\put(1.5,0.0){\frame{\shortstack{W\O\R\D}}}

The contents of the \frame command can be any of the previous picture elements, and need not be merely text. However, in many cases the output comes out wrong.

\put(0,0){\frame{\vector(1,1){1.0}}}
\put(2,0){\frame{\circle{1.0}}}

The first example produces the correct result, while the second has failed. In such cases, one can try putting the picture object inside a \makebox of suitable size and positioning as argument for the \frame command. However, it would then make more sense to use the \framebox command itself in place of \frame{\makebox...}.

Curved lines

Curved lines may be drawn in the picture environment with the commands

\qbezier[num](x_1,y_1)(x_2,y_2)(x_3,y_3)

which draw a quadratic Bézier curve from point \((x_1,y_1)\) to \((x_3,y_3)\) with \((x_2,y_2)\) as the extra Bézier point. The curve is actually drawn as \(num + 1\) dots. The number of points \(num\) is an optional argument; if it is omitted, its value will be calculated to produce a solid-looking line.

The meaning of the extra point can be illustrated with the example at the right. The input is

\begin{picture}(40,20)
\qbezier(0,0)(20,20)(40,10)
\end{picture}

The curve is drawn from \((0,0)\) to \((40,10)\) such that the tangents at the endpoints (the dotted lines) intersect at the extra Bézier point \((20,20)\). Another way of stating this is that, as one moves from the first to the third point, one begins by heading directly towards the second point, and on arrival at the destination, one is moving directly away from that second point again. The dotted lines in the above example, which are not drawn by the \bezier functions, illustrate this.

A dotted curve may be drawn by specifying the number of points \(num\); some experimentation may be required to get just the right effect.
Line thickness

There is a choice of two line thicknesses for the picture elements \texttt{circle}, \texttt{oval}, and \texttt{vector}, as well as for sloping lines. These may be selected with

\texttt{thicklines or thinlines}

Each of these declarations remains in effect until countermanded by a call to the other one, or until the end of the environment in which it was invoked. Initially, \texttt{thinlines} is in effect.

The thickness of the horizontal and vertical lines may be set with the declaration

\texttt{linethickness\{thickness\}}

to any desired size. The argument \texttt{thickness} is a positive length specification. With \texttt{linethickness\{1.5mm\}}, all subsequent horizontal and vertical lines will have a thickness of 1.5 mm.

Since frame boxes are constructed out of horizontal and vertical lines, the command \texttt{linethickness} also affects the \texttt{framebox} and \texttt{dashbox} commands.

Saving parts of pictures

It is possible to store a combination of picture elements as a sub-picture under a certain name, and to recall the whole set as often as one wants without having to reissue the individual commands every time.

First each sub-picture must have a name reserved for it with

\texttt{newsavebox\{\texttt{sub\_pic\_name}\}}

which creates a box with the name \texttt{sub\_pic\_name} for storing the picture. Afterwards, the sub-picture is saved with

\texttt{savebox\{\texttt{sub\_pic\_name\} (x\_dimen, y\_dimen) [pos] \{sub\_pic\}}

where the arguments \texttt{(x\_dimen, y\_dimen)} and \texttt{pos} have the same meaning as for \texttt{makebox} on page 291.

If the picture commands \texttt{sub\_pic} are simply a piece of text, this command is exactly the same as the \texttt{makebox} command except that the text is not printed but stored under the name \texttt{sub\_pic\_name}. The \texttt{sub\_pic} may be set down anywhere within the main picture as a separate picture element.

\texttt{usebox\{\texttt{sub\_pic\_name}\}}

\texttt{newsavebox\{\texttt{sub}\}}
\texttt{savebox\{\texttt{sub\}\{2,1\}[br]\{small Sub-Pic\}}

\texttt{put(0.7,0.0)\{\texttt{frame\{usebox\{\texttt{sub}\}\}}\}}
\texttt{put(3.0,1.0)\{\texttt{frame\{usebox\{\texttt{sub}\}\}}\}}
13.1. The picture environment

This example may not seem very practical, since the \savebox and \usebox commands could have been replaced by a \framebox together with \multiput to achieve the same result with even less effort. However, the main advantage of these two commands is not for multiple setting of text, but rather for more complex sub.pic compositions.

It should be pointed out that a \savebox command may be given outside of the picture environment, and even within the preamble. Such a sub-picture is then available in all picture environments throughout the document. However, if a \savebox is defined within an environment, it keeps its contents only until that environment comes to an end.

The picture elements inside a \savebox will be sized according to the value of \unitlength in effect at the time that the box is constructed. It will not be rescaled by a later change in \unitlength.

13.1.5 Making graph paper

The package graphpap adds a new command for drawing gridded paper:

\graphpaper[num](x,y)(lx,ly)

which places a grid with its lower left corner at (x,y), which is lx units wide and ly units high. Grid lines are drawn for every num units, with every fifth one thicker and labeled. If num is not specified, it is assumed to be 10. All arguments must be integers, not decimal fractions. For example, \graphpaper(50,50)(200,100) produces

Exercise 13.7: Produce a sheet of graph paper 10 cm by 15 cm with 2 mm separation between the lines. This sheet may be used to determine the positioning of picture elements when planning a new diagram.

13.1.6 Shifting a picture environment

The generalized syntax of the picture environment contains a further coordinate pair as an optional argument
\begin{picture}((x\ dimen,y\ dimen)\ (x\ offset,y\ offset))\ picture\ commands\ \end{picture}

In this form, \( (x\ offset,y\ offset) \) specifies the coordinates of the lower left corner. This means that for all \texttt{put} commands in the environment, the amounts \( x\ offset \) and \( y\ offset \) are effectively subtracted from the placement coordinates, so that the entire picture is shifted by \( x\ offset \) to the \textit{left} and by \( y\ offset \) \textit{downwards}.

13.2 \textbf{Extended pictures}

13.2.1 \textbf{The epic package}

With the intention of adding some higher level commands and creating a more user-friendly interface to the \texttt{picture} environment, Sunil Podar released his \texttt{epic} package, adding new features and enhancements. One idea is to be able to draw multiple objects relative to each other, with only a limited number of absolute coordinates, making reuse and shifting much easier. Another goal is to simplify the drawing of lines. This code was originally written for \LaTeX{} 2.09, but functions just as well as a \LaTeX{} 2\epsilon package. Thus it is implemented with

\texttt{\usepackage{epic}}

It makes the following new commands available to the user:

\begin{itemize}
  \item \texttt{multiputlist}
  \item \texttt{matrixput}
  \item \texttt{grid}
  \item \texttt{dottedline}
  \item \texttt{dashline}
  \item \texttt{drawline}
  \item \texttt{jput}
  \item \texttt{picsquare}
  \item \texttt{putfile}
\end{itemize}

as well as the environments:

\begin{itemize}
  \item \texttt{dottedjoin}
  \item \texttt{dashjoin}
  \item \texttt{drawjoin}
\end{itemize}

The \texttt{multiputlist} command is a variation on the regular \texttt{multiput} command; rather than placing the same element in several locations, it puts \textit{different} elements at regularly spaced intervals.

\texttt{multiputlist}(x,y)(\Delta x,\Delta y)[pos]{Obj1, Obj2, ..., ObjN}

places the \( N \) picture elements \( \textit{Obj1, ..., ObjN} \) at \( (x,y), (x+\Delta x,y+\Delta y) \), and so on. They are actually put into a series of \texttt{makebox}((0,0)[\textit{pos}]{}), so that the optional \textit{pos} argument specifies the location of the element relative to the plotted point (Section 13.1.4). The elements in the list are separated by commas, so they must be enclosed in {} if they themselves contain commas.

The \texttt{matrixput} command is the 2-D equivalent of \texttt{multiput}, creating an array of a single picture element. Its syntax is:
Thus, \matrixput(0,0)(3,5){3}(10,0){5}\{\circle{1.5}\} produces

\[
\begin{array}{ccccccc}
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\end{array}
\]

The \grid command is similar to the \graphpaper command, but it will also optionally label the axes.

\grid(width, height)(\Delta width, \Delta height)[X_0,Y_0]
draws a grid of size width×height, with grid lines at intervals \Delta width and \Delta height. If the optional argument is given, the grid lines are labeled starting at \(X_0, Y_0\), which must be integers, incremented by \(\Delta width\) and \(\Delta height\), which must also be integers in this case. The \grid command must be placed inside a \put command to specify its lower left corner: \put(0,0){\grid...}.

A set of points can be joined by a dotted line with the command

\dottedline[dot_char]{dot_gap}((x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n))

where \(dot_gap\) is the spacing (in \unitlength units) between the dots and \(dot_char\) is the character used for the dot; if it not specified, the \picsquare (see below) is used by default, a small square. A near solid line can be achieved by making \(dot_gap\) very small, but this could cause \TeX to run out of memory.

To join a set of points with a dashed line, one uses

\dashline[stretch]{dash_len}{dot_gap}((x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n))

where \(dash_len\) is the length of the dashes (in \unitlength units), which are constructed as a series of dots to produce a solid looking dash. If the optional \(dot_gap\) is given, then the dashes are made as a series of dots with that separation. The optional \(stretch\) is a number between −100 and infinity. If it is 0 or missing, the number of dashes is chosen so that the gaps between them are the same length as the dashes; if \(stretch\) is positive, the number of dashes is increased by \(stretch\) percent; if negative, it is reduced by that amount. With various combinations of \(stretch\) and \(dot_gap\), different dash patterns can be produced.

To draw a solid-looking line through a set of points, one uses

\drawline[stretch]((x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n))

which draws the lines as a series of line segments using the \LaTeX line fonts. Since these have a limited number of angles, the result can appear fairly jagged. By increasing the optional \(stretch\) factor (again a percent), more segments are used and the appearance is improved; by decreasing it, the line can begin to look dashed.

As an example of these three line drawing commands,
For each of the three line drawing commands, there is a corresponding environment, named \texttt{dottedjoin}, \texttt{dashjoin}, and \texttt{drawjoin}, each taking the same set of optional and mandatory arguments as the commands, except for the set of coordinates. Within these environments, one gives a set of \texttt{jput} commands, which behaves exactly like the regular \texttt{put}, except that all its elements are joined by the selected line type. For example,

\begin{verbatim}
\begin{picture}(100,13)
\begin{dottedjoin}{2}
 \jput(0,0){\circle{3}}
 \jput(20,8){\makebox(0,0){\Large$\ast$}}
 \jput(40,3){\makebox(0,0){\small Finish}}
\end{dottedjoin}
\begin{dashjoin}{3}
 \jput(55,3){}
 \jput(70,10){\oval(10,5)}
 \jput(80,0){\circle*{2}}
\end{dashjoin}
\end{picture}
\end{verbatim}

produces:

\begin{center}
\begin{picture}(100,13)
\begin{dottedjoin}{2}
 \jput(0,0){\circle{3}}
 \jput(20,8){\makebox(0,0){\Large$\ast$}}
 \jput(40,3){\makebox(0,0){\small Finish}}
\end{dottedjoin}
\begin{dashjoin}{3}
 \jput(55,3){}
 \jput(70,10){\oval(10,5)}
 \jput(80,0){\circle*{2}}
\end{dashjoin}
\end{picture}
\end{center}

Real life data curves are generated by other software programs. These results can be imported into the picture environment by storing them in a data file, named say \texttt{mycurve.put} and then issuing

\begin{verbatim}
\putfile{mycurve.put}{\picsquare}
\end{verbatim}

The data file contains a list of $x y$ coordinates, one pair per line, with possible comments beginning with \texttt{%} as usual. The \texttt{putfile} command places its second argument at all the points listed in the data file.

The default dot character for plotting all lines is the \texttt{picsquare}, a black square that scales with the current line thickness. A different dot character may be used with \texttt{putfile} if one wishes. It is also possible to specify the dot character for \texttt{dottedline}, but the default is \texttt{picsquare}. 
The default values for the optional \textit{stretch} argument in \texttt{\textbackslash dashline} and \texttt{\textbackslash drawline} are initially 0, but may be changed at any time by redefining (with \texttt{\textbackslash renewcommand}) \texttt{\textbackslash dashlinestretch} and \texttt{\textbackslash drawlinestretch}, respectively. This makes it possible to revise a whole set of curves with one command, rather than manually changing each one.

### 13.2.2 The eepic package

The epic commands are still subject to many of the limitations of the \texttt{picture} environment: limited number of slopes for lines, limited line thicknesses, restrictions on circle sizes. This is because the \texttt{picture} environment uses \LaTeX\ fonts to `draw' inclined lines and circles, making the results portable to all output devices.

The eepic package, by Conrad Kwok repairs these problems by employing graphic commands executed by the DVI driver itself. They are transmitted to the .\texttt{dvi} file by \texttt{\special} commands, which are driverspecific. Thus greater flexibility is achieved, but at the price of loss of portability. The package is therefore most suitable for use with the PostScript \texttt{dvips} driver, as well as with the previewers \texttt{xdvi} and \texttt{windvi}. It does not work with \texttt{pdfLaTeX}, unfortunately; on the other hand, it has no problems with \texttt{dvipdfm}.

However, eepic not only recodes the features of epic, it also adds some additional functionality. If one wishes to invoke these extra features with a driver that does not recognize the graphics \texttt{\specials}, one can use the emulation package eepicemu instead. A comparison of the results with these two packages is shown in Figure 13.1 on the next page.

In either case, one must load \texttt{epic} with one of \texttt{eepic} or \texttt{eepicemu}:

\begin{verbatim}
\usepackage{epic,eepic} or \usepackage{epic,eepicemu}
\end{verbatim}

The following standard \texttt{picture} elements are modified by the eepic package:

\begin{verbatim}
\line(\Delta x,\Delta y)\{length\} (page 293) where \Delta x and \Delta y may take on any positive or negative integer values, and not just those between ±6.
\end{verbatim}

\begin{verbatim}
\circle\{diameter\} (page 295) may have any value for \texttt{diameter}; the same for \texttt{\circle*}.
\end{verbatim}

\begin{verbatim}
\oval (page 296) may have the maximum diameter of the corners set to any value; this is stored in the length \texttt{\maxovaldiam} which the user may change; the default value is 40 pt.
\end{verbatim}

Similarly, all the extra commands from epic package are also revised internally. The restrictions on slopes never apply directly to them, but lines of arbitrary slope are drawn with little segments at the nearest available slope, or as series of dots, requiring much computer time and
overloading the output file. With \eepic, these lines are drawn more efficiently.

Additional features added by \eepic (and \eepicemu, even if they do not work very well):

\texttt{\allinethickness\{dimen\}}

sets the line thickness to \textit{dimen} for \textit{all} picture elements, including circles, ovals, splines, not just for straight lines;

\texttt{\Thicklines}

sets thickness of straight lines to 1.5 times that of \texttt{\thicklines};

\texttt{\path(x_1,y_1)(x_2,y_2)...(x_n,y_n)}

is a fast version of \texttt{\drawline}, without the \textit{stretch} option; a solid line is always drawn;

\texttt{\spline(x_1,y_1)(x_2,y_2)...(x_n,y_n)}

draws a curve from the first to last point, with intermediate points as control points;

\texttt{\ellipse(x\_diam,y\_diam)}

draws an ellipse with horizontal and vertical sizes \textit{x\_diam}, \textit{y\_diam} respectively; there is also \*-form to produce a solid ellipse;

\texttt{\arc\{diameter\}\{start\}\{end\}}

draws the arc of a circle with \textit{diameter} (in \texttt{\unitlength} units) from angle \textit{start} to \textit{end}, in radians, clockwise from 0 pointing to the right; \textit{start} must be from 0 to $2\pi$, and \textit{end} from \textit{start} to $\textit{start} + 2\pi$. 

\begin{figure}[h]
\centering
\includegraphics{figure13.1}
\caption{Comparison of results using the true \eepic drawing functions (top) and those with the emulation in \eepicemu (bottom). (This example is taken from the \eepic user's manual.)}
\end{figure}
13.3 Other drawing packages

13.3.1 The \texttt{XY-pic} package

As indicated at the beginning of this chapter, an exhaustive description of many other specialized packages, for drawing or using \textsc{Metafont} fonts, can be found in \textit{The \LaTeX Graphics Companion} by Goossens et al. (1997). Most of these are designed for \LaTeXe in general, but will also work with \LaTeX. We mention two of these here just to show the possibilities.

The \texttt{XY-pic} package can carry out general drawing entirely with extra \textsc{Metafont} fonts, making it fully portable among all output drivers. The fonts are also available in type 1 coding (Section G.6) so that the package works just as well with PDF output. It was created by Kristoffer H. Rose and then extended by Ross Moore and others.

Even an overview of \texttt{XY-pic} is beyond the scope of this book. We recommend the supplied \textit{User's Guide} and more extensive \textit{Reference Manual} both to be found in \texttt{texmf/doc/generic/xy-pic}. The descriptions and illustrations in Chapter 5 of Goossens et al. (1997) are especially helpful.

To give a flavor of the syntax in \texttt{XY-pic}, we give the following examples:

\begin{verbatim}
\xymatrix{
  U \ar@/_/[ddr]_y \ar@/^/[drr]^x \ar@{.>}[dr]|-{(x,y)} \\
  & X \times_Z Y \ar[d]^q \ar[r]_p & X \ar[d]_r \\
  & Y \ar[r]^g & Z}
\end{verbatim}

that produce the diagrams:

\begin{center}
\begin{tikzpicture}
\node (U) at (0,0) {$U$};
\node (x) at (1.5,0) {$X \times_Z Y$};
\node (X) at (3,0) {$X$};
\node (Y) at (0,-1.5) {$Y$};
\node (Z) at (3,-1.5) {$Z$};
\node (P) at (4.5,-0.75) {$P$};
\node (C) at (4.5,-3.25) {$C$};
\draw[->] (U) to node[above]{$x$} (x);
\draw[->] (U) to node[left]{$y$} (Y);
\draw[->] (x) to node[below]{$q$} (Y);
\draw[->] (x) to node[right]{$p$} (X);
\draw[->] (Y) to node[below]{$r$} (Z);
\draw[->] (X) to node[below]{$\theta$} (Z);
\end{tikzpicture}
\end{center}

The first example, with \texttt{xymatrix}, organizes its contents in rows and columns, not dissimilar to the \texttt{tabular} environment. In this case, the first row contains \texttt{U} in the first column, and the rest are empty; the second row has an empty first column (starts with \&), the column separator) while the next two columns contain \texttt{X \times_Z Y} and \texttt{X}; the third row also has an empty first column, with entries \texttt{Y} and \texttt{Z} in the remaining columns. Everything else describes how these nodes are connected. For example,
\ar@/_/[ddr]_y \text{ means ‘draw an arrow, curving down to the node located two down and one right, and subscript it with } y. \text{ And so on for the other connections.}

The second example, with \xy \ldots \endxy gives more complicated drawing instructions, placing objects at given coordinates, joining them with arrows, lines, curves. As can be seen from both of these examples, it takes some practice to learn the syntax and to exploit \text{Xy-pic} fully.

13.3.2 \text{PSTricks}

In Chapter 10 we describe how the PostScript language plays a very important role in \TeX\ and \LaTeX\ by making a wide spectrum of fonts available to complement the original Computer Modern fonts. But PostScript is also a powerful plotting language. It was in order to exploit the PostScript possibilities directly that Timothy van Zandt wrote the \text{PSTricks} package for direct use within \TeX\, and thus within \LaTeX.\n
Actually \text{PSTricks} consists of over a dozen packages. The complete functionality may be loaded with the package \text{pstricks}, or the individual packages may be loaded singly. (Since \text{PSTricks} is really a \TeX\ utility, the \LaTeX\ package \text{pstricks.sty} does nothing more than load the \text{pstricks.tex} file containing the actual code.) Some of these packages are:

\begin{itemize}
  \item \text{pstricks-3d} 3-D drawing
  \item \text{pstricks-blur} adding blurred shadows
  \item \text{pstricks-char} stroking and filling character paths
  \item \text{pstricks-coil} coil and zigzag objects
  \item \text{pstricks-eps} export objects to EPS files
  \item \text{pstricks-fill} adds \text{\psboxfill} command
  \item \text{pstricks-gr3d} drawing 3-D grids
  \item \text{pstricks-grad} gradient color fills
  \item \text{pstricks-lens} simulating effect of a lens
  \item \text{pstricks-node} placing and joining nodes
  \item \text{pstricks-osci} emulating oscilloscopes
  \item \text{pstricks-plot} plotting data
  \item \text{pstricks-poly} drawing polygrams
  \item \text{pstricks-s1pe} improvement on \text{pstricks-grad}
  \item \text{pstricks-text} setting text along a path
  \item \text{pstricks-tree} tree commands
\end{itemize}

As with \text{Xy-pic}, it is beyond the scope of this book on document production with \LaTeX\ to describe the extensive plotting features of \text{PSTricks}. We refer to Goossens \textit{et al.} (1997), Chapter 4 for a description of most of these packages (some have been added since then) and otherwise to the supplied manuals and documentation to be found in \text{texmf\generic\source\pstricks}.\n
Bibliographic Databases and \texttt{BibTeX}

In academic publications, a bibliography or list of references is standard procedure. In Section 9.3 we describe how a bibliography can be formatted with the \texttt{thebibliography} environment and how its entries are referred to within the text. Often authors find that they are constantly referring to the same publications in most of their own papers. Similarly, researchers working in the same field will frequently be referring to the same set of papers. This means that many of the entries in the \texttt{thebibliography} environment will be repeated from one work to the next, or even among co-workers at one institute.

It would be very useful if the bibliographic entries could be stored in one database file once and for all to be available for all documents with a list of references in that field. Such a database system is possible with the \texttt{BibTeX} program supplied with the \LaTeX{} installation. The information about the various publications is stored in one or more files with the extension \texttt{.bib}. For each publication there is a key that identifies it, and which may be used in the text document to refer to it. Such a file is called a bibliographic database.

14.1 The \texttt{BibTeX} program

\texttt{BibTeX} is an auxiliary program to \LaTeX{} that automatically constructs a bibliography for a \LaTeX{} document by searching one or more databases. To this end, the \LaTeX{} file must contain the command

\begin{verbatim}
\bibliography{database1, database2, \ldots}
\end{verbatim}

at that point in the text where the bibliography is to appear. Here the argument \texttt{database1, database2, \ldots} is a list of root names, separated by commas, of the bibliographic database files that are to be searched. The extension \texttt{.bib} is not explicitly written.

Reference can be made to a publication in one of the databases at any place in the text with the commands
as is explained in Sections 9.3.3 and 9.3.4. The key is the database identifier for that publication, which of course the user must know beforehand. After the first \LaTeX processing, the \BibTeX program must be run either by clicking the appropriate icon in the editor shell, or from a command line with \texttt{bibtex} plus the root name of the \LaTeX file. Supposing this source file name is \texttt{comets.tex}, the call

\begin{verbatim}
bibtex comets
\end{verbatim}

produces a new file with the name \texttt{comets.bbl}, containing the extracted information for those publications for which there was a \texttt{\cite} reference, packaged in a \texttt{thebibliography} environment to be input into the document on the next \LaTeX run.

Occasionally the bibliography is to include publications that were \textit{not} referenced in the text. These may be added with the command

\begin{verbatim}
\nocite{key}
\end{verbatim}

given anywhere within the main document. It produces no text at all but simply informs \BibTeX that this reference is also to be put into the bibliography. With \texttt{\nocite{*}}, \textit{every} entry in all the databases will be included, something that is useful when producing a list of all entries and their keys.

After running \BibTeX to make up the \texttt{.bbl} file, it is necessary to process \LaTeX \textit{at least twice} to establish both the bibliography and the in-text reference labels. The bibliography will be printed where the \texttt{\bibliography} command is issued; it in fact inputs the \texttt{.bbl} file.

The formatting of the bibliography may be selected with the declaration

\begin{verbatim}
\bibliographystyle{style}
\end{verbatim}

which may be issued anywhere in the document. Its only function is to inform \BibTeX what bibliography style file (\texttt{.bst}) is to be loaded to determine the resulting format. Many contributed \texttt{.bst} files exist, designed for specific journals and publishing houses, and it is possible to create one’s own (Section 14.3). However, the supplied standard ones for numerical citation only are:

\begin{description}
\item[plain] The entries in the bibliography are ordered alphabetically; each is assigned a running number in square brackets as the in-text reference marker, printed where the \texttt{\cite} commands are issued.
\item[unsrt] The entries are ordered according to their first references by the \texttt{\cite} and \texttt{\nocite} commands. The entry for the first \texttt{\cite}
receives the number 1, that of the next \cite with a different key the number 2, and so on. The markings and listings are otherwise the same as for plain.

**alpha** The ordering in the bibliography is the same as for plain but the markers are an abbreviation of the author's name plus year of publication. A reference to Smith (1987) would appear as [Smi87] instead.

**abbrv** The ordering and marking are the same as for plain, but the bibliographic listing is shortened by abbreviating first names, months, and journal names.

For author-year citations and the natbib package (Section 9.3.4), there exist the bibliographic styles plainnat, unsrtnat, and abbrvnat, producing the same bibliography format as the corresponding standard styles, but with natbib compatibility.

### 14.2 Creating a bibliographic database

Creating a bibliographic database might seem like more work than typing up a list of references with the \thebibliography environment; the great advantage is that the entries need to be included in the database once and only once and are then available for all future publications. Even if a different bibliography style is demanded in later works, all the information is already on hand in the database for \LaTeX to write a new \thebibliography environment in another format. In fact, we feel that even for a single document, it is simpler to make an entry into the database than to adhere to the very precise and fiddly requirements of a literature list, especially regarding punctuation and positioning of the authors' initials. The database entry proceeds very quickly and easily if one has a generalized template, as illustrated in Section 14.2.6.

The entries in a bibliographic database are of the form

```latex
@BOOK{knuth:86a,
    AUTHOR = "Donald E. Knuth",
    TITLE = \{The \TeX\}{}book",
    EDITION = "third",
    PUBLISHER = "Addison--Wesley",
    ADDRESS = \{Reading, Massachusetts\},
    YEAR = 1986
}
```

The first word, prefixed with @, determines the entry type, as explained in the next section. The entry type is followed by the reference information for that entry enclosed in curly braces \{ \}. The very first entry is the key
for the whole reference by which it is referred to in the \cite command. In the above example this is knuth:86a. The key may be any combination of letters, numerals, and symbols, except commas. The actual reference information is then entered in various fields, separated from one another by commas. Each field consists of a field name, an = sign, with optional spaces on either side, and the field text. The field names shown above are AUTHOR, TITLE, PUBLISHER, ADDRESS, and YEAR. The field text must be enclosed either in curly braces or in double quotation marks. However, if the text consists solely of a number, as for YEAR above, the braces or quotation marks may be left off.

For each entry type, certain fields are required, others are optional, and the rest are ignored. These are listed with the descriptions of the various entry types below. If a required field is omitted, an error message will occur during the Bib\TeX run. Optional fields will have their information included in the bibliography if they are present, but they need not be there. Ignored fields are useful for including extra information in the database that will not be output, such as a comment or an abstract of the paper. Ignored fields might also be ones that are used by other database programs.

The general syntax for entries in the bibliographic database reads

```latex
@entry_type{key, 
  field_name = {field text}, 
  ... 
  field_name = {field text} }
```

The names of the entry types as well as the field names may be written in capitals or lower case letters, or in a combination of both. Thus @BOOK, @book, and @bOOk are all acceptable variations.

The outermost pair of braces for the entire entry may be either curly braces \{ \}, as illustrated, or parentheses ( ). In the latter case, the general syntax reads

```latex
@entry_type(key, ... ... )
```

However, the field text may only be enclosed within curly braces \{ ... \} or double quotation marks " ... " as shown in the example above.

### 14.2.1 The entry types

The following is a list of the standard entry types in alphabetical order, with a brief description of the types of works for which they are applicable, together with the required and optional fields that they take. The meanings of the fields are explained in the next section.

@article Entry for an article from a journal or magazine.

- **required fields** author, title, journal, year.
- **optional fields** volume, number, pages, month, note.
14.2. Creating a bibliographic database

<table>
<thead>
<tr>
<th>Entry Type</th>
<th>Description</th>
<th>Required Fields</th>
<th>Optional Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>@book</td>
<td>Entry for a book with a definite publisher.</td>
<td>author or editor, title, publisher, year.</td>
<td>volume or number, series, address, edition, month, note.</td>
</tr>
<tr>
<td>@booklet</td>
<td>Entry for a printed and bound work without the name of a publisher or sponsoring organization.</td>
<td>title.</td>
<td>author, howpublished, address, month, year, note.</td>
</tr>
<tr>
<td>@conference</td>
<td>Is the same as @inproceedings below.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@inbook</td>
<td>Entry for a part (chapter, section, certain pages) of a book.</td>
<td>author or editor, title, chapter and/or pages, publisher, year.</td>
<td>volume or number, series, type, address, edition, month, note.</td>
</tr>
<tr>
<td>@incollection</td>
<td>Entry for part of a book that has its own title.</td>
<td>author, title, booktitle, publisher, year.</td>
<td>editor, volume or number, series, type, chapter, pages, address, edition, month, note.</td>
</tr>
<tr>
<td>@inproceedings</td>
<td>Entry for an article in conference proceedings.</td>
<td>author, title, booktitle, year.</td>
<td>editor, volume or number, series, pages, address, month, organization, publisher, note.</td>
</tr>
<tr>
<td>@manual</td>
<td>Entry for technical documentation.</td>
<td>title.</td>
<td>author, organization, address, edition, month, year, note.</td>
</tr>
<tr>
<td>@mastersthesis</td>
<td>Entry for a Master's thesis.</td>
<td>author, title, school, year.</td>
<td>type, address, month, note.</td>
</tr>
<tr>
<td>@misc</td>
<td>Entry for a work that does not fit under any of the others.</td>
<td>none.</td>
<td>author, title, howpublished, month, year, note.</td>
</tr>
<tr>
<td>@phdthesis</td>
<td>Entry for a PhD thesis.</td>
<td>author, title, school, year.</td>
<td>type, address, month, note.</td>
</tr>
</tbody>
</table>
@proceedings Entry for conference proceedings.
required fields title, year.
optional fields editor, volume or number, series, address, month, organization, publisher, note.

@techreport Entry for a report published by a school or other institution, usually as part of a series.
required fields author, title, institution, year.
optional fields type, number, address, month, note.

@unpublished Entry for an unpublished work with an author and title.
required fields author, title, note.
optional fields month, year.

Each entry type may also have the optional fields key and crossref. The former provides additional information for alphabetizing the entries, especially when the author information is missing. The author information is normally found in the author field but may also be in the editor or even organization fields. This key field has nothing to do with the key for identifying the entry in the \cite command. The crossref field gives the key for another entry in the database that shares many of the information fields, as illustrated in Section 14.2.3.

14.2.2 The fields

The fields that may be used within a bibliographic entry are listed below together with their meanings. They are always given in the form:

\[field\_name = \{field\_text\}\ or\ field\_name = "field\_text"

address
The address of the publisher or other institution. For major publishing houses, it is sufficient to give just the city. For smaller publishers, giving the full address is recommended.

annotate An annotation that may be used by non-standard bibliography styles to produce an annotated bibliography. The standard Bib\TeX\ styles ignore it.

author The name(s) of the author(s) as described in Section 14.2.4.

booktitle The title of a book when only part of it is being cited. See Section 14.2.4 for special considerations on capitalization.

chapter A chapter or section number.
crossref
The key of another entry in the database that shares many of the
same field entries. See Section 14.2.3.

edition
The edition of a book, usually written in full and capitalized,
as 'Second'. The standard styles will change it to lower case as
necessary.

editor
The name(s) of the editor(s) as described in Section 14.2.4. If
there is also an author field, then this gives the editor of the
book or collection in which the citation appears.

howpublished
States anything unusual about the method of publishing. Should
be capitalized. Example: 'Privately published'.

institution
The name of the sponsoring institution for a technical report.

journal
The name of a journal or magazine. Abbreviations are provided
for the most common ones (see Section 14.2.5).

key
An addition for alphabetizing purposes when the author inform-
ation is missing. This is not the same as the key for identifying
the entry to a \cite command.

month
The month in which the work was published or, if unpublished,
when it was written. Abbreviations exist in the form of the first
three letters of the (English) names.

note
Any additional information that should be added. Capitalize the
first letter.

number
The number of a journal, magazine, technical report, or work in
a series. Journals are usually identified by volume and number;
technical reports are issued a number by the institution; books
in series sometimes have a number given to them.

organization
The sponsoring organization for a conference or a manual.

pages
A page number or a range of pages, in the form 32, 41, 58 or
87–101 or 68+. The last form indicates page 68 and following
pages. A single hyphen given for a range will be converted by
the standard styles to the double hyphen to produce a dash, as
‘87–101’.

publisher
The publisher’s name.
school The name of the academic institution where a thesis was written.

series The name of a series or set of books. When citing a book from a series, the title field gives the name of the book itself while the optional series specifies the title of whole set.

title The title of the work, obeying the capitalization rules listed in Section 14.2.4.

type The type of technical report, for example, ‘Research Note’.

url The universal resource locator, or Internet address, for online documents; this is not standard but supplied by more modern bibliography styles.

volume The volume number of a journal or multi-volume book.

year The year in which the work was published or, if unpublished, in which it was written. It should normally consist of four numerals, such as 1993.

Additional field names may be included, which \texttt{BibTeX} will simply ignore. For example, to add the abstract of an article in the database,

\begin{verbatim}
abstract = \{text of an abstract\}
\end{verbatim}

This can be of use in other applications as well as for database management.

### 14.2.3 Cross-referencing fields

If many entries in the bibliographic database share a common set of field information, such as for a number of works all appearing in the same conference proceedings, it is possible to refer to another entry containing that common set with the crossref field. For example,

\begin{verbatim}
@INPROCEEDINGS{xyz-1,
crossref = {xyz-proceedings},
author = {J. S. Jones},
title = {The First Results from the \{Appleville Experiment\}},
pages = {34--38} }
\end{verbatim}

\begin{verbatim}
@PROCEEDINGS{xyz-proceedings,
editor = {C. H. Kelvin},
title = {Proceedings of the First Conference on the \{Appleville Experiment\}},
booktitle = {Proceedings of the First Conference on the \{Appleville Experiment\}}
year = 1991 }
\end{verbatim}
The first entry, with key \texttt{xyz-1}, is to obtain all its missing fields from the second entry, by means of the \texttt{crossref} field referring to the entry with key \texttt{xyz-proceedings}. The missing fields are \texttt{editor}, \texttt{booktitle}, and \texttt{year}, those that are common to all articles in the conference proceedings. Note that \texttt{booktitle} is an ignored field for \texttt{@PROCEEDINGS} but needs to be included here since it is required for \texttt{@INPROCEEDINGS}.

If an entry is referred to by two or more other entries, then it too will be included in the bibliography, even though its key never appeared as the argument of a \texttt{\cite} or \texttt{\nocite} command.

In order for this system to function properly, the entry that is referred to must appear in the database(s) \textit{after} all those that refer to it. It is therefore recommended to put all such referenced entries at the end of the database. Cross-references may not be nested.

### 14.2.4 Special field formats

There are some special rules for entering the texts to the fields \texttt{author}, \texttt{editor}, \texttt{title}, and \texttt{booktitle}. \LaTeX{} will process names, putting surnames first, abbreviating given names with initials, and so on, according to the instructions in the style file. Thus it is important that the program knows what is a given name and what a surname. Similarly for titles, capitalization may change depending on style and/or entry type, so \LaTeX{} must know what words are always capitalized.

#### Names

The names in the author and editor fields may be typed in either in the form \{\textit{Given Names Surname}\} or as \{\textit{Surname, Given Names}\}. That is, \LaTeX{} assumes that if there is no comma the last capitalized name is the surname, or family name; otherwise what comes before the comma is taken to be the surname. Thus the name texts "John George Harrison" and "Harrison, John George" are equivalent for Mr J. G. Harrison. However, if a person has a double surname, without a separating hyphen, the second form must be employed, or the double name must be enclosed in braces, as

"San Martino, Maria" or "Maria \{San Martino\}"

for Ms M. San Martino.

Auxiliary words to a surname that are not capitalized, such as \textit{von} or \textit{de}, may be entered in either form:

"Richard von Mannheim" or "von Mannheim, Richard"

"Walter de la Maire" or "de la Maire, Walter"

Anything enclosed in braces will be treated as a single item, something that is used in ambiguous cases, or when the name contains a comma or the word \textit{and}. An example is
"{Harvey and Sons, Ltd}"

If the name contains a Junior or some other addition, it must be entered \{Surname, Junior, Given Name\}, for example as

"Ford, Jr, Henry" or "Ford, III, Henry"

However, if there is to be no comma, then the Jr must be treated as part of a double surname, something that is not recommended at all:

"{Filmore Jr}, Charles" or "Charles {Filmore Jr}"

Accents within a name formed with a backslash command should be enclosed in braces with the backslash as the first character after the opening brace. In this way, the alphabetization and the formation of labels with the alpha bibliography style will function properly. For example,

\begin{verbatim}
author = "Kurt G{"o}del",
year = 1931
\end{verbatim}

will produce the label [Göd31] as desired. The accent text must not be enclosed any deeper than shown here.

Accents should be formed with the backslash command for most generality. Some language modifications have shorthands for accented letters (like "a instead of \"a for German) but these should be avoided in the database to ensure that the results will be universally understood.

Hyphenated first names are properly abbreviated. Thus "Jean-Paul Sartre" becomes 'J.-P. Sartre'.

If an author or editor field is to contain more than one name, the names are separated by the word and. For example,

\begin{verbatim}
author = "Helmut Kopka and Daly, Patrick William" or
AUTHOR = {Peter C. Barnes and Tolman, Paul and Mary Smith}
\end{verbatim}

If the and is actually part of the name, the whole name must be enclosed in braces, as pointed out above.

Do not insert any \~ characters between names or initials; \TeX{} will do this automatically as it deems fit.

If only initials are given for the authors given names, insert a blank between them, as P. W. Daly, not P.W. Daly. The latter will be interpreted as a single given name and will be abbreviated as ‘P. Daly’

If the author list is too long to type in all the names, it may be terminated with and others. This will be converted to the form of et al. prescribed in the style file.

\textbf{Titles}

The capitalization of the title depends on the bibliography style: usually book titles are capitalized while article titles are not. The text in the fields
14.2. Creating a bibliographic database

The general rules for capitalizing titles in English state that the first word of the title, the first word after a colon, and all other words are capitalized except articles and unstressed prepositions and conjunctions. For example:

```
> title = "The Right Way to Learn: A Short-Cut to a Successful Life"
```

Words that are always to be capitalized, such as proper nouns, must be enclosed in braces. It is sufficient to enclose only the letter that must be capitalized. The two following examples are equivalent:

```
> title = "The {Giotto} Mission to Comet {Halley}"  
> TITLE = {The {G}iotto Mission to Comet {H}alley}
```

14.2.5 Abbreviations

It is always possible to use an abbreviation in the field text in place of actual text. Some abbreviations, such as the names of the months and some standard journal names, are already available, and the user is free to define new ones for his or her personal use.

The name of an abbreviation consists of any combination of letters, numbers, and symbols except

```
" # % ' ( ) , = { }
```

An abbreviation is defined with the command

```
@string{abbrev.name = {text}}  
@string(abbrev.name = {text})
```

where `abbrev.name` stands for the name of the abbreviation and `text` is the replacement text. For example, if the abbreviation

```
@string{JGR = {Journal of Geophysical Research}}
```

has been defined, the following two field statements are identical:

```
> journal = JGR  
> journal = {Journal of Geophysical Research}
```

The name of the abbreviation is not enclosed in braces or double quotes, otherwise the name itself will be interpreted literally as the field text. In both the `@string` command and the name of the abbreviation, the case of the letters is unimportant. The above abbreviation could just as well be defined as

```
@STRING{jgr = {Journal of Geophysical Research}}  
or  @StrinG{jGr = {Journal of Geophysical Research}}
```
and it may be referred to in the fields as JGR, JGr, JgR, Jgr, jGR, jGr, jgR or jgr.
Abbreviations may be concatenated with the symbol # between them. Thus after giving

@string{yrbk = {Institute Yearbook}}

this abbreviation may be combined with other abbreviations and text, as in

\texttt{title = "Max-Planck" # yrbk # 2000}

to produce 'Max-Planck Institute Yearbook 2000'.

Standard abbreviations exist for the months of the year, the names consisting of the first three letters of the name, as jan, feb, etc. These are actually included within the .bst file, and can change their definition for styles intended for other languages. However, the name of the abbreviation remains the same, the first 3 letters of the English name.

Further predefined abbreviations are available (in the .bst file) for some standard journal names. Those produced with custom-bib are especially extensive. See Section 14.3.

The @string commands may be issued anywhere in the database between two bibliographic entries, but must already be defined before they are used. Therefore, it makes most sense to insert all the abbreviation definitions at the beginning of the database file.

\subsection{14.2.6 Using a template}

Typing in the text for bibliographic entries into a database might seem to be rather complicated since there are so many things to remember, such as which fields are required and which are optional, what their formats are, and so on. One way to simplify this task is to make up templates for the most common entry types that you use. A template is basically a complete entry with all the fields left blank. It is stored in a separate file on its own, to be inserted into the database file whenever a new entry of that type is to be made. Then the field texts are written in.

An appropriate template for the entry type @article could be

@ARTICLE{<key>,
    AUTHOR = {},
    TITLE = {},
    YEAR = {},
    JOURNAL = {},
    volume = {},
    number = {},
    month = {},
    pages = {},
}
in which the <key> is a reminder that the keyword is to go here, the required fields come first and are capitalized, while the optional fields are in lower case.

Finally, we mention that Bib\TeX{} has been written by Oren Patashnik, Stanford University, in close association with Leslie Lamport. The Bib\TeX{} installation should contain the two files btxdoc.tex and btxhak.tex for additional information about Bib\TeX{}. In particular, btxhak.tex contains instructions for writing bibliographic style files. Process these files with \LaTeX{}, if the resulting .dvi or .pdf files are not already there.

14.3 Customizing bibliography styles

It seems that every journal and publishing house has its own set of rules for formatting bibliographies. The differences are really minor, such as whether to use commas or colons here, or to put volume numbers in bold or italic typeface. Nevertheless, in spite of this triviality, the rules are rigid for each house.

Standard \LaTeX{} provides only four bibliography styles, and their differences have more to do with sorting and labeling than with such fussy details. It is possible to design one’s own .bst file, by modifying existing ones; however, this requires a knowledge of the peculiar Bib\TeX{} programming language, something that puzzles even experienced \LaTeX{} users. There are some 50 .bst files in the Bib\TeX{} directories of the \TeX{} file servers, and probably many more in contributed \LaTeX{} directories. Each is designed for a specific journal and there is no guarantee that it would be accepted by another one. What is a user to do if his or her publisher demands a certain format that just is not to be found? How can one seek a given format anyway among those offered?

Some help can be found in the custom-bib application written by Patrick W. Daly. The heart of this package is a generic (or master) bibliography style merlin.mbs containing alternative coding for a multitude of different bibliographic features or options. It is processed with the DocStrip program which is capable of including alternative coding according to menu information contained in the .mbs file itself. This means that there could be a number of .mbs files
to choose from, each one explaining its set of options to the user via \makebst.

Support for other languages is provided by means of additional .mbs files, one for each language, which contain the translations for such words as volume, editor, edition, and so on.

Some of the features provided by the supplied generic bibliographic style file are:

- numerical or author-year citations; in the latter case, one may also choose which \bibitem style is to be used;
- order of references: by citation order, alphabetical by all authors, alphabetic by author-year label;
- format of authors' names: first name plus surname, initials plus surname, surname plus initials, reversed initials only for first author, and more;
- number of author names to include before giving et al.;
- typeface to be used for the author names;
- position of date, parentheses or brackets around the year;
- format of volume, number, and pages for journals;
- capitalization of article titles as sentence or title style;
- whether to use the word and or an ampersand &;
- placement of commas with the word and;
- whether to abbreviate editor, volume, chapter, etc.;
- first and last page numbers or only first;
- optional addition of shorthand designations of common journal names;
- and much more.

Abbreviations for many journal names in physics, optics, and geophysics can be provided in those .bst files produced by custom-bib, which then may be used in Bib\TeX databases. It is unfortunate that these are to be found in the .bst files themselves, which means that they may not be universally present. Which ones are present depends on the options chosen by the producer of the style file. The file shorthnd.tex in the custom-bib installation lists all the current possible abbreviations. (Actually, it is generated by running L\TeX on shorthnd.ins, which then produces the most current list from the existing .mbs files.) To check what any particular .bst file contains, open it and look for the MACRO commands.
So far we have been assuming that the output of a \LaTeX{} project is to be printed paper or an electronic document looking like printed paper displayed on the computer monitor.

Another important type of output is presentation material, the visual support displayed in front of an audience during an oral presentation. Traditionally, this consisted of a set of slides to the projected onto a screen, but in more recent times this has been replaced by viewgraphs (or transparencies) and an overhead projector, giving the speaker much more interactive control over the presentation. However, the truly modern presentation is done electronically, with the entire presentation stored in a computer directly connected to the projector; there are no slides to fall out of the cassette, or be inserted the wrong way around, no viewgraphs to spill out onto the floor. The only thing that can go wrong is that the computer refuses the connection, needs rebooting, or the necessary display program is missing. Such teething problems are occurring less frequently now that this form of presentation is becoming standard.

In spite of the change in medium away from the $2'' \times 2''$ pieces of film, a computer presentation is still called a slide show, a term which today presupposes the electronic medium. We will use the word slide more generally to mean a page of text and/or graphics for presentation, whether it is to be printed on paper and then photographically rendered to film or transparency, or to be directly projected from the computer. The preparation of such a slide with \LaTeX{} is much the same regardless of the final projection method; direct projection does offer additional features which we also address.

We start this chapter with the standard \LaTeX{} method for slide preparation, the \texttt{slides} class, and then describe the far more advanced \texttt{seminar} class (Section 15.2). The latter is especially suitable for interesting PDF electronic presentations. We then illustrate \texttt{pdfscreen} (Section 15.3), a package designed more for documents meant for electronic viewing but which can also produce slides. Finally we show in Section 15.4 how PDF presentations may be enhanced with the help of an additional program.
15.1 Slide production with $\text{SLiTeX}$

The term $\text{SLiTeX}$ was used in the old $\LaTeX$ 2.09 system to indicate a separate specialized version of $\LaTeX$ for producing projection material. This awkward and cumbersome device was needed then because of the way fonts were hardwired into the formats. With $\LaTeX\TeX2e$, this is no longer necessary, and the entire functionality is now found within the $\text{slides}$ class, considerably streamlined over the obsolete version, running under regular $\LaTeX$, or pdf$\LaTeX$. The word $\text{SLiTeX}$ therefore should be retired, but since it is a nifty expression, we will continue to use it to distinguish the $\text{slides}$ class, which is considerably different from $\text{article}$ and the other standard classes.

15.1.1 The $\text{slides}$ class

A slide is meant to be projected before an audience and is therefore something quite different from a sheet of paper to be read held in the hand. One could of course make up the text for slides with the normal $\LaTeX$ commands, but there would be problems getting the font size right, arranging overlays, making sure that the text does not change pages unexpectedly, and the question of whether book-style fonts are suitable for projection. The $\LaTeX$ class $\text{slides}$ attempts to solve these.

A different set of fonts is used by this class, ones in which the lower case letters are relatively larger than in the normal fonts, and with a much bigger base size. This has the effect of limiting the amount of text that fits on one page/slide, but this is in fact very good practice for presentation material. The text really should be restricted to keywords and abbreviated sentences. A full page of normal text projected on to a screen will not be read by the audience.

One is not obliged to use the built-in fonts; one may still select other (sans serif) font packages, such as $\text{helvet}$ for Helvetica (Section 10.1.2) as one pleases. They will be used in larger sizes than usual.

Slides should also make use of color. The original $\text{SLiTeX}$ had a complicated method for producing color overlays (in black and white, to be copied in color) which now is totally eliminated from the $\text{slides}$ class. Instead, one simply makes use of the $\text{color}$ package of Section 6.2.

Most of the formatting commands of regular $\LaTeX$ may be used with $\text{slides}$, except for the page breaking and sectioning commands. The special features that are unique to this class are described in the next sections.

15.1.2 The slide environments

The source file for producing slides is structured much the same as that for a normal document, except that it is the $\text{slides}$ class that is invoked
15.1. Slide production with \texttt{\LaTeX} 3.25

as

\begin{verbatim}
\documentclass{slides}
preamble text
\begin{document}
slide text
\end{document}
\end{verbatim}

If colors are wanted the \texttt{color} package of Section 6.2 must be added, for example as

\begin{verbatim}
\usepackage[pdftex]{color}
\end{verbatim}

(You will have to specify your own driver as the option here, or rely on the local configuration to give it automatically, as is usually the case.)

The preamble may contain global specifications, for changing the paper size or selecting a page style, as usual. It may not contain any printable material.

Any text that appears after \texttt{\begin{document}} but before the special slide environments described below is output to an unnumbered \texttt{leading page} that comes before any slides. This may serve as a cover page for the slides.

There are three environments available for organizing different parts of a slide: the main \texttt{slide} itself, possible \texttt{overlays}, and additional \texttt{notes} to the slide.

\textbf{Slides}

A slide or viewgraph is created by means of the environment

\begin{verbatim}
\begin{slide} text and commands \end{slide}
\end{verbatim}

The contents of a \texttt{slide} environment may be any text that one pleases. Note that the \texttt{slides} class makes use of its own set of character fonts. The standard font in the normal size is roughly equivalent to the \LaTeX{} sans serif font \texttt{sffamily} in the size \texttt{\LARGE}. All the regular font commands and declarations of Section 4.1 are available, along with the other display and list environments of Chapter 4. However, there can be no page breaks within the environment, for the entire text is expected to fit on one page (slide or viewgraph). The successive \texttt{slide} environments will be numbered consecutively. If the page should overflow, a warning message is printed.

Any color commands from the \texttt{color} package may be employed, if that package has been loaded.

\textbf{Overlays}

An \textit{overlay} is an addition to a slide that can be laid on top of it to fill in certain gaps. The idea is to create suspense during a presentation, by
filling in some key words a few minutes later, or to be able to replace some text by alternatives.

The \texttt{slides} class generates overlays with the \texttt{overlay} environment, which functions exactly the same as the \texttt{slide} environment, except that the numbering is done as a sub-number of the last slide. Thus the overlays following slide 6 are numbered 6-a, 6-b, and so on.

\begin{overlay} \textit{text and commands} \end{overlay}

The \texttt{slide} environment for a viewgraph that is to have overlays must come before the \texttt{overlay} environments that go with it. Both the slide and overlay should contain identical texts, except that certain parts are printed in ‘invisible’ ink by means of the two declarations

\begin{verbatim}
\invisible \visible
\end{verbatim}

These two commands function just like font declarations. They may be set within curly braces {} to limit their scope, or used to apply to the whole environment. Normally, in the \texttt{slide}, a few words might be made invisible, while in the corresponding \texttt{overlay}, \texttt{\invisible} is declared at the start and only those words that were blanked out in the slide are made visible. This may be seen in the sample on page 329.

\textbf{Note:} the \texttt{\invisible} command does not work with fonts other than the default ones, nor with pdf\TeX. However, for pdf\TeX, there are better methods to achieve the same results (Section 15.4.1).

One application for overlays is to present alternative text. One might, for example, show a table of cost estimates, with the figures on an overlay. By exchanging this overlay for another, new numbers that can be achieved by certain procedural changes may be fitted into the same table.

\textbf{Notes}

During a presentation, it is often necessary to refer to a list of keywords or other notes between the actual projected slides. The \texttt{slides} class has the \texttt{note} environment for producing such reminders for the speaker.

\begin{verbatim}
\begin{note} \textit{text} \end{note}
\end{verbatim}

Like the overlays, notes are sub-numbered with respect to the previous slide, but with numbers instead of letters. Following slide 4, notes are numbered 4-1, 4-2, and so on.

\textbf{15.1.3 Further features}

\textbf{Page styles}

The \LaTeX\ command \texttt{\pagestyle{style}} may also be used with the \texttt{slides} class. The following styles are available:
15.1. Slide production with \TeX

plain  All slides, overlays, and notes have their number at the lower right corner.

headings
The same as plain except that if the clock option is selected (see below), a time marker is printed in the lower left corner of the notes. This is the default page style.

empty  The sheets are printed with no page numbers.

In normal L\TeX, the command \texttt{\pagestyle} is issued in the preamble and is valid for the whole document. Individual pages may be given a different style by means of the \texttt{\thispagestyle} command which remains in effect for only one page. This command should not be employed within one of the environments, but rather a new \texttt{\pagestyle} command may be given outside of the \texttt{slide}, \texttt{overlay}, and \texttt{note} environments.

Timing notes
The option \texttt{clock} can be selected with \texttt{\documentclass}; it activates two additional commands:

\begin{verbatim}
\settime{secs}
\addtime{secs}
\end{verbatim}

and prints the time in minutes at the bottom of each note. In this way, the notes contain a reminder of how far advanced the presentation should be at that point. The value of the time marker is set and incremented with the above two commands, which take a number in seconds as argument. The initial value of the timer is 0.

Selective slide processing
During the construction of a slide show, one is likely to spend considerable time working on individual slides, ignoring the others for the time being. Complicated slides need to be corrected, processed, and viewed. One can speed this up by selecting only a few slides for processing, by placing the command

\begin{verbatim}
\onlyslides{slide_list}
\end{verbatim}

in the preamble. The \texttt{slide_list} stands for a list of slide numbers in ascending order, for example, \texttt{2,5,9-12,15}, specifying the slides, or range of slides, that are to be processed.

Non-existent slide numbers may also appear in the \texttt{slide_list}. If the slide file contains, say, 20 slides, the command \texttt{\onlyslides{1,18-999}} will allow only slides 1 and 18–20 to be processed. Any overlays that belong to these slides will also be output.

Finally, the command
\onlynotes{note_list}

in the preamble limits the notes that are output to those listed in note_list. Supposing that slide 5 has three note environments associated with it, \onlynotes{5} arranges that only the note pages 5-1, 5-2, and 5-3 are printed.

The \onlyslides and \onlynotes commands function similarly to the command \includeonly described in Section 9.1.2. An interactive means of selecting slides and notes to process may be set up exactly as illustrated in Section 9.1.3 with the \typein command:

\typein[\onlyslides]{Which slides to do?}
\typein[\onlynotes]{Which notes to do?}

During processing, the message 'Which slides to do?' is typed to the monitor, and the user replies with the desired slide numbers. A similar scheme may be set up for \onlynotes.

A sample slide file

An example of input text with the slides class, with a leading page, overlay, and note, and using the clock option, is given here. The four-page result is shown on the opposite page.

\documentclass[a4paper,clock]{slides}
\begin{document}
\begin{center}\Large\bfseries
Sample Viewgraphs
\end{center}

\begin{slide}
\begin{center}\large Advantages of \texttt{slides}\end{center}
\begin{itemize}
\item Uses special fonts
\item Forces key words instead of long text\invisible
\item Supports color layers\visible
\end{itemize}
\end{slide}

\begin{overlay}\invisible\begin{center}\large Advantages of \texttt{slides}\end{center}\end{overlay}

Color commands may be used {\invisible as color layers}

\end{slide}

\begin{overlay}\invisible\begin{center}\large Advantages of \texttt{slides}\end{center}\end{overlay}
Advantages of slides

- Uses special fonts
- Forces key words instead of long text

Color commands may be used

Add the overlay only for \LaTeX 2.09
Chapter 15. Presentation Material

\begin{itemize}
\item Uses special fonts
\item Forces key words instead of long text
\item Supports color layers
\end{itemize}

Color commands may be used \{\visible as color layers\}

\begin{addtime}{300}
\begin{note}
Add the overlay only for \LaTeX{} 2.09
\end{note}
\end{addtime}

15.2 Slide production with seminar

The \texttt{slides} class of the previous section some basic support for the production of presentation material. To improve on this, Timothy van Zandt has written the \texttt{seminar} class with many additional features. This was originally meant as a 'main style' for \LaTeX{} 2.09, but has been upgraded to a \LaTeX{} 2ε class file by Sebastian Rahtz with fixes by David Carlisle and Denis Girou. They only changed the interfacing so that it would work as a class file, leaving the functionality untouched.

Unfortunately, the supplied manual for \texttt{seminar} (\texttt{sem-user.tex}) has not been updated, and is still couched in the language of \LaTeX{} 2.09; in particular it confuses packages and 'style options'. We will try to correct this here.

If \texttt{seminar} is being used for PDF output, as for an electronic presentation, it is vital to include Sebastian Rahtz' \texttt{hyperref} package (Section 10.2.4), which detects the presence of \texttt{seminar} and adjusts the PDF page sizes accordingly. This should be done even if no other \texttt{hyperref} features are utilized. There are some other refinements to make \texttt{seminar} function smoothly with pdf\LaTeX{}, explained in Section 15.2.7.

15.2.1 Overview

The \texttt{seminar} class is loaded as usual with

\texttt{\documentclass[options]{seminar}}

Here we summarize its main features which differ from \StyLeX.
The slide material is put into the `slide` environment, for landscape slides, and in `slide*` for portrait slides.

Regular Roman fonts are used, but this can be changed. The options `semhelv` and `semlcmss` switch to sans serif family with the PostScript Helvetica and the standard \texttt{SLiTeX} fonts, respectively.

Regular font sizes are used, with default 10 pt as usual. However, the slide text is magnified on output to make it more suitable for projection; the magnification factor can be set, but the default is a factor of 2, so the effective default font size is 20 pt.

A slide is started automatically when the current one is full, or may be forced with the \texttt{\newslide} command; the entire slide show could therefore be contained within a single `slide` environment.

There is a choice of framing style for the slide text; the framing may be turned off with \texttt{\slideframe{none}} in the preamble.

Everything outside of the `slide` or `slide*` environments counts as note text. The notes are printed on separate pages, numbered \(n.1, n.2, \ldots\), for those following slide number \(n\).

The options `slidesonly`, `notes`, and `notesonly` control whether only the slides, both slides and notes, or only the notes are printed. By default, both are printed.

With the `article` option (not to be confused with the class of the same name), the slide show is printed in miniature form, 2 per page, appropriate for reviewing or as a handout. There is a `notesonly*` option to be used with `article` to indicate the missing slides.

We now explain these features in more detail.

### 15.2.2 Magnification and lengths

Rather than using enlarged fonts as does \texttt{SLiTeX}, \texttt{seminar} works with ‘normal’ sizes, which are then magnified at output. One can set the magnification step with \texttt{\slidesmag{num}} in the preamble, where \texttt{num} is an integer between \(-5\) and 9. Step 0 is one-to-one, and each step indicates an additional factor of 1.2. The default is step 4 (factor 2).

By default, the basic font size is 10 pt, but this may be altered with the usual options \texttt{11pt} and \texttt{12pt}. All these are magnified on output. The font size can be changed at any point in the document with \texttt{\ptsize{pt}}, where \texttt{pt} can be one of 8, 9, 10, 11, 12, 14, or 17. If this is given within a `slide` environment, it applies only within that environment.
The article option produces a handout, with 2 slides per page, using a separate magnification which is set with \articlemag{num}; the default step is 0, no magnification.

The output magnification applies to all sizes and lengths. This means, any lines drawn with \rule or spaces inserted with \hspace or \vspace are also magnified if they specify absolute units like pt or in. The size of the slide itself remains unchanged by the magnification, since the purpose is only to increase the size of the contents relative to the slide. To set lengths to values that remain fixed relative to the slide dimensions, use \setslide length and \addtoslide length in place of \setlength and \addtolength. For example,

\newlength{\fixed}
\setslide length{\fixed}{3in}
\addtoslide length{\fixed}{1cm}
\rule{1pt}{\fixed}

draws a line of height 1 pt \times magnification and of fixed length 3 in + 1 cm.

For the article output, there are also the corresponding commands \setartlength and \addtoartlength.

The following length commands are slide lengths (do not magnify) and may be used wherever a length is required:

\semcm 1 centimeter
\semin 1 inch
\textwidth width of the text area
\textheight height of the text area
\linewidth current width of text

These are particularly useful with the \includegraphics command and its options width= and height=.

15.2.3 Landscape and portrait slides

In contrast to \LaTeX, the seminar class allows landscape and portrait slides to be mixed. However, there are several aspects that need to be considered.

The slide and slide* environments determine whether the contents should be in landscape or portrait slides, but the actual placement will only be correct when it agrees with the orientation of the entire document, as specified by the options landscape and portrait. That means, when the landscape option is chosen, the landscape slides will be properly formatted, but the portrait ones not. The reverse applies with the portrait option. What one must do is to print the landscape and portrait slides separately, each with the appropriate option, and with the \landscapeonly or \portraitonly command in the preamble.
When using the \texttt{dvips} driver, one can print the landscape and portrait slides together by including the option \texttt{semrot}, which ensures that the portrait slides are rotated into the landscape orientation. The entire document (slide show) can be printed in one go. The portrait sheets are then manually rotated to the correct orientation. This would not be acceptable for a PDF electronic presentation, where one wants the two orientations to be mixed in one file, with the text always upright. Another solution for \texttt{pdf\LaTeX} is given is Section 15.2.7.

When printing both orientations together with the \texttt{semrot} option, the head and footlines will not normally be rotated with the text for portrait slides. This can be corrected by issuing \texttt{\rotateheaderstrue}. In this way, the output is the same as when one prints landscape and portrait slides separately.

Selecting landscape orientation in itself does not force the output printer to go into landscape mode; it only sets the page dimensions so that the text is written along the longer side. The printer is informed about the landscape intentions either by a switch when the file is sent to it, or by an option to the DVI driver. Again, \texttt{dvips} can get around this, as explained on page 232. Using \texttt{seminar} and \texttt{dvips}, one should define

\begin{verbatim}
\renewcommand{\printlandscape}{\special{landscape}}
\end{verbatim}

To allow the class to instruct the printer to go landscape \emph{when necessary}. This also avoids the warning message on processing that the output must be printed in landscape mode.

\section*{15.2.4 Slide formatting}

Here we describe the various ways to control the size, layout, and other formatting aspects of the slides.

\subsection*{Slide size}

By default, the paper size is American letter size, $11 \times 8.5$ in, and the slide dimensions (slide writing area) are $8.5 \times 6.3$ in. By specifying the \texttt{a4} option the paper size becomes $29.7 \times 21$ cm and the slide dimensions $22.2 \times 15.2$ cm. (The \texttt{a4paper} option exists, but may cause an error.)

The slide size may be altered temporarily as optional arguments to the \texttt{slide} and \texttt{slide*} environments, as for example,

\begin{verbatim}
\begin{slide}[10cm,7cm]
\begin{slide*}[7cm,10cm]
\end{verbatim}

Both of the above produce a slide 10 cm wide and 7 cm high.
Margins

Normally the material within a slide is centered vertically. The declaration commands

\centerslidefalse \textit{and} \centerslidetrue

turns this centering off and on, respectively. Without centering, the text is pushed to the top of the slide.

The text is left justified by default, with a ragged right margin. This can be changed with

\raggedslides[len]

where \texttt{len} is the maximum space between the end of the line and the right margin. Omitting \texttt{len} produces a ragged right margin, while a value of 0 pt yields right justification; giving 2 em, say, creates a semi-ragged margin.

The margins between the slide area and the edge of the page are set with the commands

\slidemargins

which are all set initially to 0.6 in. Since these are commands, not lengths, they must be changed with \texttt{\renewcommand}.

Framing

The slide area can be framed, the default being a rectangular box like that produced by \texttt{\fbox} (which is exactly what it is). The frame style can be chosen with

\slideframe[\textit{cmds}]{style}

The possible values for \texttt{style} are plain (the default) or none. The optional \texttt{cmds} are settings to adjust the frame parameters, such as \texttt{\fboxrule} and \texttt{\fboxsep}. By including the fancybox package, one also has shadow, double, oval, and Oval as allowed values for \texttt{style}, with their corresponding parameters (Section 4.7.9).

All frames make use of the lengths

\slideframewidth \textit{and} \slideframesep

for the frame thickness and separation from text. These may only be altered (with \texttt{\setlength}) outside the \texttt{\slideframe} command; in the optional \texttt{cmds}, one must alter \texttt{\fboxrule} and \texttt{\fboxsep} directly.

It is possible to define one's own frame style with

\newslideframe{new\_style}[\textit{cmds}]{\texttt{\box\_cmd}\{#1\}}
where \textit{new\_style} is the name of the new style (for use with \texttt{\slidestyle}), \textit{cmds} are box parameter settings, and \texttt{\box\_cmd} is a box command, like \texttt{\fbox} or \texttt{\shadowbox}.

There is a \texttt{semcolor} option for \texttt{seminar}, but this is not based on the \LaTeX\ color package, but depends on PostScript and the \texttt{dvips} driver. With the \texttt{color} package, one can make use of the \texttt{\color} and \texttt{\textcolor} commands for setting text in color, as well as \texttt{\colorbox} and \texttt{\fcolorbox} for colored boxes and \texttt{\pagecolor} to color the whole page (page 167). Two examples for defining color frame styles are

\begin{verbatim}
\newslidestyle{gplain}{\fcolorbox{green}{white}{#1}}
\newslidestyle{rshadow}{\color{red}]{\shadowbox{\color{black}#1\color{red}}}
\end{verbatim}

The first makes use of the color frame box command \texttt{\fcolorbox}, while the second shows a trick to combine color commands with \texttt{\shadowbox}, which normally prints a black shadow box. The two \texttt{\color{red}} commands ensure all sides of the box are in red, while the \texttt{\color{black}#1} puts the box’s contents in black.

### Page styles

Page styles determine how the running head and footlines appear on each page, just as in normal text. The regular \LaTeX\ page style of Section 3.2 (plain, empty, myheadings, and headings) are all available, and the default is plain, with the page/slide number centered in the footline. A different page style may be selected with \texttt{\pagestyle{style}} at any time, or with \texttt{\thispagestyle{style}} for just the current page.

There is also an additional page style \texttt{align} that puts + marks at the four corners.

You can define your own page styles with

\begin{verbatim}
\newpagestyle{style}{\headline}{\footline}
\renewpagestyle{style}{\headline}{\footline}
\end{verbatim}

which is preferable to using the styles in the \texttt{fancyhdr} package of Section 3.2.2 since that cannot handle the magnified lengths correctly. For example, to define a page style with a centered text in the head, and a footline with a text at the left and the slide number at the right, give

\begin{verbatim}
\newpagestyle{mystyle}{\hfill Report\hfill}{March 20, 2004\hfill\thepage}
\end{verbatim}

Note here that \texttt{\thepage} prints the number for the current page, whereas \texttt{\theslide} and \texttt{\thenote} refer to the latest slide and note numbers respectively; \texttt{\thepage} is always set to whichever of them is being printed.
Font commands can be included in the new definitions, but one should be aware that the commands `\slideheadfont` and `\slidefootfont` are always inserted into the head and footlines. These are defined to be `\scriptsize`, but may be redefined by the user.

It is possible to select different page styles for the slides and notes. The `\pagestyle` command sets the style for both (as well as for the `article` output) while `\slidepagestyle` applies only to slides.

**Page breaking**

Unlike \LaTeX where each `slide` environment generates a single slide (and warnings if it is too big), the `seminar` package allows the text to break naturally into multiple slides just as normal text does into pages. A new slide may be forced with `\newslide`.

You can add some extra tolerance for the page breaking by specifying

`\extraslideheight{len}`

where `len` is the additional height allowed before a break in automatically inserted. By setting this to be large, automatic page breaks never occur, and the user has full control, either with `\newslide` or by ending the environment. Setting it to 0 pt causes pages to be broken as for regular text. The default is 10 pt.

**15.2.5 Selective output**

One can select to output just the slides, or just the notes, or both, by adding one of the options `slidesonly`, `notesonly`, or `notes` to the `\documentclass` command. There is also a `notesonly*` option to print the notes alone in `article` output but with indicators for the slides.

A problem arises if `slidesonly` has been selected. Any global specifications issued after `\begin{document}` and outside of a `slide` environment will be ignored. One might want to redefine parameters in this way; including them inside the `slide` environment makes the changes local to that one environment. To avoid this, such global commands should be placed inside a `allversions*` environment. (If you actually want to include text and not just non-printing specifications, use the `allversions` environment instead.)

Just as for \LaTeX, one can output selected slides or notes. This is useful during development of the slide show, when one may spend much time designing a complicated slide and is constantly processing it to see the results; this can become tediously slow if the other, completed, slides are also complicated or include large graphics files. As for \LaTeX (page 327), the `\onlyslides{list}` command in the preamble ensures that only those slides whose numbers are included in `list` are processed. However, there is also a `\notslides{list}` command to exclude those slides in `list`. 
The \textit{list} a set of numbers in any order, separated by commas, or a range with a hyphen. It is even possible to include \texttt{\ref} commands for slides containing the corresponding \texttt{\label}, something that is very useful during development when the slide numbers are yet finalized. For example,

\texttt{\\onlyslides{4,2,\ref{sum},10-999}}

outputs slides 2, 4, the slide containing \texttt{\label{sum}}, and all from 10 to 999. Non-existent numbers are ignored, so 10–999 really means ‘10 to the end’, unless there really are so many slides in this show.

\section*{15.2.6 Additional features}

\textbf{Counters}

The \texttt{seminar} class provides all the standard \LaTeX\ counters (Section 8.1) as well as slide and note counters, the values of which are printed with \texttt{\theslide} and \texttt{\thenote}. The page number is printed with \texttt{\thepage}, which is set to whichever of the others is current, or to the true page number for \texttt{article} output.

The footnote counter is the only one that is reset to 1 for each new slide. If you want other counters to be reset too, say equation, specify this with

\texttt{\\slidereset{list} \ or \ addtoslidereset{list}}

where \textit{list} is a comma-separated list of counters. The first command overwrites the existing list of counters, the second adds to it.

\textbf{The note environment}

As an alternative to note pages being generated by all text outside of the slide environment, one can issue \texttt{\noxcomment} in the preamble, or select option \texttt{noxcomment} in \texttt{\documentclass}, and then place all the notes in \texttt{\begin{note} \ldots \end{note}} environments.

\textbf{Overlays}

Overlays for slides can be produced by including the options \texttt{semlayer} and \texttt{semcolor}. These make use of PostScript coding and therefore only work with the \texttt{dvips} driver or equivalent.

The \texttt{\overlay{n}} indicates that the following text is to appear on overlay \textit{n}; it remains in effect until the current scope is ended, at the end of the environment or \{..\} group, or when another \texttt{\overlay} command is issued.
Overlays are not so useful for a PDF electronic presentation. Instead, one creates the same effect by building the page successively. A method of doing this is explained in Section 15.4.1.

**Adding a background image**

Slide shows are made more effective with the use of color, and with a background pattern or picture. A background color can be added with the \texttt{\textbackslash pagecolor} command from the \texttt{color} package.

However, even more effective is a background image. If this is a graphics file named \texttt{mybg.png}, one can insert it as a background to all slides with the following trick. This does require the \texttt{fancybox} package for the \texttt{\fancyput} command.

\begin{verbatim}
\newlength{bgwidth}
\newlength{bgheight}
\ifportrait
  \setslidelength{bgwidth}{\paperwidth}
  \setslidelength{bgheight}{\paperheight}
\else
  \setslidelength{bgwidth}{\paperheight}
  \setslidelength{bgheight}{\paperwidth}
\fi
\ifarticle
\else
  \fancyput(-1\semin,1\semin){\raisebox{-bgheight}{\includegraphics[totalheight=bgheight, width=bgwidth]{mybg.png}}}
\fi
\end{verbatim}

What this does is first to define and set two new lengths, \texttt{bgwidth} and \texttt{bgheight}, to be the non-magnified size of the full page, for the current orientation. The \texttt{\fancyput} command places its contents on each page, relative to a point 1 in from the upper left corner; with the above coordinates, it is shifted exactly to the upper left corner. The graphics is inserted here, shifted down by the page height, and expanded/shrunk to fit the page exactly. One must design the graphics correctly so that any distortion is minimal. The \texttt{\ifarticle} construction prevents the background image from being included with \texttt{article} output.

This should work with any type of output, PostScript or PDF or other. Of course, the type of graphics file must conform to those allowed by the DVI driver. For PostScript, it must be an \texttt{.eps} file, not \texttt{.png}.

See also page 345 for an alternative means of adding background images.
15.2. Slide production with seminar

Local configuration

The seminar class reads in the file seminar.con if it exists. This gives the user the chance to write all the specifications and new definitions that he or she always employs to this file, to have them automatically included in every seminar source file.

15.2.7 Using pdfTEX with seminar

Although the seminar class originated in the pre-LATEX 2ε days, and many of its extra features are coded for PostScript and the dvips driver without the modern graphics and color packages, there is no problem using it with today’s LATEX and those packages. This is particularly true for PDF output.

However, with pdfTEX itself, there are some extra length parameters \pdfpagewidth and \pdfpageheight which need to be properly set to get the output right. This is most simply guaranteed by including the hyperref package (Section 10.2.4) even if no other of its features is needed.

Another difficulty arises if landscape and portrait slides are mixed. If the goal is only to print to paper or transparencies, then the two orientations may be processed and output separately, as explained above. However, for an electronic presentation, this would be most inconvenient. Certainly the ability to mix the two orientations is highly desirable.

The following code makes this possible. It revises the slide* environment to interchange the PDF page dimensions, so that a true portrait page is produced. (This is not possible with printed output where a single print job must have a fixed orientation.)

\@ifundefined{pdfoutput}\{\endinput\}{\% \ifcase\pdfoutput \endinput \fi}
\newcommand*\{\pdf@revpage\}{\% \@tempdima=\pdfpagewidth \pdfpagewidth=\pdfpageheight \pdfpageheight=\@tempdima \@tempdima=\paperwidth \paperwidth=\paperheight \paperheight=\@tempdima }
\ifarticle \pdf@revpage \fi
\renewcommand\{\printlandscape\}{} \expandafter\let\expandafter\slide@str\csname slide*\endcsname \@namedef{slide*}\{\pdf@revpage\slide@str}
Store this in a file named sempdftx.sty, which can then be loaded with \usepackage. It is programmed to do nothing if pdf\TeX is not being used. Alternatively, add a line \RequirePackage{sempdftx} to the local configuration file seminar.con to load it on every seminar run.

Note: This file can be copied from the enclosed CD in the directory \books\Kopka_and_Daly, where a version of seminar.con is also available.

15.3 Electronic documents for screen viewing

Strictly speaking, this section is not about presentation material that is to be projected before an audience, but rather about documents that are deliberately intended to be read from a computer monitor. These have other requirements from printable documents: the page shape should be more like a monitor (that is, landscape instead of portrait) and navigation aids should be included especially if the viewer's regular tool bars have been switched off. And of course, internal links should be present, as well as links to external web sites and documents.

The pdfscreen package by C. V. Radhakrishnan is an effort to fulfill these requirements. But it is more flexible than that, for it also allows the entire document to be output in traditional print format. And it even contains a \texttt{slide} environment for making projection material, so it does indeed belong to this chapter.

The supplied manual can be found in texmf\doc\latex\pdfscreen, in print and screen versions. The source file manual.tex can serve as a good example of how to use the package. Figure 15.2 on the opposite page demonstrates what a title page can look like, complete with navigation panel which appears on every page.

Parameters for pdfscreen

This package should be loaded with the \texttt{article} class and with the \texttt{hyperref} package. In fact, it should be considered to be an extension to \texttt{hyperref}. And it should be loaded last, right after \texttt{hyperref}.

The options that may be invoked with the \usepackage command are

\texttt{screen} to generate the output in the screen version, for online reading;

\texttt{print} to generate the print version of the output;

\texttt{panelleft} to place the navigation panel at the left;

\texttt{panellright} to place the navigation panel at the right;
15.3. Electronic documents for screen viewing

**Figure 15.2:** Title page of a pdfscreen document

- **nopanel**
  - to suppress the navigation panel;

- **paneltoc**
  - to include a table of contents in the navigation panel;

- **nocfg**
  - to ignore any local configuration file;

- **sectionbreak**
  - to insert a new page at the start of sections.

In addition one can specify various color schemes for the panel and its buttons. Possible color options are *bluelace*, *blue*, *gray*, *orange*, *palegreen*, and *chocolate*. Also all the babel language options (Chapter 11) are recognized, for labeling the navigation buttons, although only 15 are actually supported; the others default to English.

There are additional parameters that must be set by declarations:

- \**\emlema{image}\*
  - to set the name of the graphics file used for the logo in the navigation panel;
\urlid{url} \\
\text{to set the web address (URL) of the home page button in the navigation panel;}

\screensize{height}{width} \\
\text{to set the dimensions of the screen; since there are no defaults, this must be given;}

\margins{left}{right}{top}{bottom} \\
\text{to set the margins for the document; again there are no defaults, so this must be given.}

\textbf{The navigation panel}

The navigation panel appears on each page and contains the buttons for changing pages and linking to the author's home page. A default panel is provided, but the user may redesign it as he or she pleases, by redefining the \texttt{\panel} command. For example,

\begin{verbatim}
\renewcommand{\panel}{\colorbox{panelbackground}
  {\begin{minipage}[t][\paperheight][b]{\panelwidth}
    . . . . . \\
    \Acrobatmenu{FirstPage}
    {\addButton{.2in}{$\blacktriangleleft$}}
    . . . . .
  \end{minipage}}}
\end{verbatim}

Here the \texttt{\Acrobatmenu} command is from the hyperref package (see page 247) while \texttt{\panelwidth} and \texttt{\addButton} are from pdfscreen. The \texttt{\panelwidth} length is by default 15\% of the screen width, and can be redefined. The \texttt{\addButton{wth}{text}} creates a button of width \texttt{wth} containing \texttt{text}. In the above example, clicking the button executes the viewer command to go to the first page.

There is also a command

\begin{verbatim}
\imageButton{wth}{ht}{image}
\end{verbatim}

to insert the graphics file \texttt{image} as a button of width \texttt{wth} and height \texttt{ht}.

A row of small navigation buttons can be turned on at the bottom of the page with \texttt{\bottombuttons} and turned off with \texttt{\nobottombuttons}.

\textbf{Background image}

A graphics file can be inserted as background for the whole page with \texttt{\overlay{image}}, or a uniform background color can be selected with \texttt{\backgroundcolor{color}}.

The navigation panel has as background that color previously defined as \texttt{panelbackground}, which the user may alter. Otherwise, the graphics
file *image* may be inserted as the panel background with the command `\paneloverlay{image}`.

**Making slides**

There is a *slide* environment that places its contents centered vertically within a box the full size of the text area.

**Local configuration**

Any settings that you want to use for all documents with *pdfscreen* may be written to a file named *pdfscreen.cfg*, which is read in on loading if present.

**Controlling print or screen output**

The entire file can be processed either as a document to be printed in the traditional way, or as an electronic document for reading on a monitor. This is switched by the options *print* and *screen* when loading the package.

Text and specifications that are meant exclusively for one output version must be enclosed in *print* or *screen* environments. This is especially important for the *\screensize* and *\margins* commands, which are inappropriate in *print* mode.

**Other packages loaded**

The *pdfscreen* package makes heavy use of several other packages, which it loads automatically. These must then also be installed on the system:

```
  \begingroup
  \begin{tabular}{llllll}
    hyperref & graphicx & calc & shortvrb \\
    comment   & color & amssymb & fancybox \\
    truncate & colortbl & amsbsy
  \end{tabular}
  \endgroup
```

**15.4 Special effects with PDF**

There are many fancy features that one expects today from an electronic presentation: sounds and animation, text dancing in from the sides, fades, and possibly fireworks. A PDF file can include many of these.

The *pdf\TeX* program cannot (yet) generate these directly, but there is an auxiliary program called *PPower4*, written in Java by Klaus Guntermann and Christian Spannagel. This is a post-processor designed to take as input a PDF file specially prepared with *pdf\TeX* with some extra packages, and to create a new PDF file with the additional tricks.
For example, in order to build up a page containing an itemized list, such that each item should pop up one at a time, one adds a `\pause` command after each item, runs `pdflatex`, and processes the output with PPower4. It splits the page up into several pages, each one containing the text up to the next `\pause`. By stepping through these pages, the presenter ‘constructs’ the one slide before the eyes of the audience. In fact, PPower4 can do much more, making a very sophisticated electronic slide show from pure `\LaTeX` input.

The PPower4 program and the necessary `\LaTeX` support packages are on the `\TeX`Live CD, but the latest versions can be obtained from [http://www-sp.iti.informatik.tu-darmstadt.de/software/ppower4/](http://www-sp.iti.informatik.tu-darmstadt.de/software/ppower4/). At the time of writing, it is still under development, but is already very impressive.

### 15.4.1 The PPower4 packages

The PPower4 post-processor is run with only two parameters: the names of the input and output files. There are no additional adjustments, controls, or options at processing time. Everything of that sort has already been placed in the `\LaTeX` source text itself with the help of supplied packages. These instructional commands add comments to the resulting PDF file, which may be viewed and printed as normal. It is the post-processor that reads these comments and uses them as instructions for making up the new output file.

Here we describe these packages and the marker commands that they enable. The use of the `hyperref` package is presupposed, since many effects can be achieved directly with it.

#### Page transitions

Transitions from one page to another can be done in various ways to catch attention. The simplest is the plain replacement: the old page vanishes, the new one appears. Others involve effects that look like blinds opening, or like a box opening from or towards the center, or dissolving from one to the other. These can be set with the `hyperref` package with the option `pdfpagetransition` (page 245) or set to a different effect anywhere in the document by giving

\[
\text{\texttt{hypersetup{pdfpagetransition=\{pars\}}}}
\]

where `pars` are the required parameters for the desired transition.

Marc van Dongen has prepared a file `pagetrans.tex`, available with the PPower4 collection, that defines simpler commands to execute the above. (If this were renamed `pagetrans.sty`, it would be a package and could be loaded with `\usepackage` but as is, it must be included with the `\input` command.) These commands are:
\Replace pages are simply replaced, the default;
\Dissolve one page ‘dissolves’ into the other in a mosaic fashion;
\VBlinds several horizontal lines sweep vertically across the screen, like Venetian blinds;
\HBlinds same as \VBlinds, but vertical lines move horizontally;
\HOSplit two vertical lines sweep horizontally out from the center;
\HISplit as above, but the lines move into the center;
\VOSplit two horizontal lines sweep vertically out from the center;
\VISplit as above, but the lines move into the center;
\OBox a box opens out from the center;
\IBox a box sweeps into the center;
\Wipe{angle} a single line sweeps across the screen in the direction given by \textit{angle}, which can be 0 (left to right), 90 (bottom to top), 180 (right to left), 270 (top to bottom);
\pageTransitionGlitter{angle} like \Dissolve but in a band that moves across the screen in the direction \textit{angle}, which can be 0 (left to right), 270 (bottom to top), 315 (upper left to lower right).

We stress again, these declarations are simplifications of the \hyperref commands, and function without PPowder post-processing. Once given, they remain in effect for all pages until a new transition command is declared.

\textbf{Adding a background}

We have already seen in Section 15.2.6 how a background image may be added with the help of the fancybox package, and how a solid background color can be set with the \texttt{color} package.

These and other effects can also be achieved with the background package supplied with PPowder4. This package provides the commands

\begin{verbatim}
\hpagecolor[color1]{color2} and \\
vpagecolor[color1]{color2}
\end{verbatim}

that cause the background color to blend from \texttt{color1} to \texttt{color2} horizontally or vertically. If the optional \texttt{color1} is omitted, it is the intensity of \texttt{color2} that changes across the page. Both colors must be defined by the \texttt{color} package command \texttt{definecolor}, or be one of the predefined colors. \textit{This effect requires post-processing with PPowder4!}
346  Chapter 15.  Presentation Material

The package can also add background images, without PP\textsf{ower}4 post-processing, if it is loaded with the option \texttt{bgadd}. This then includes the commands

\begin{verbatim}
\bgadd{image}  \textit{and}  \bgaddcenter{image}
\end{verbatim}

to place \textit{image} at the upper left corner or centered on the page. The \textit{image} is an \texttt{\includegraphics} command with appropriate scaling to fit the page. These commands may be issued more than once to add further elements to the background. They may all be removed with \texttt{bgclear}.

\textbf{Note:} The addition of background elements to the pages requires that the package es\texttt{o-pic} also be installed.

\textbf{Building a page successively}

\textbf{Package: pause}  The electronic equivalent to overlays is the successive building of a page of text. The PP\textsf{ower}4 \texttt{pause} package introduces the \texttt{\pause} command that places markers in the PDF output which PP\textsf{ower}4 then uses to break the page at that point, and to start a new page containing the previous contents plus whatever comes up to the next \texttt{\pause} or end of page. This works for both pdf\TeX{} and dvipdfm.

For example, we could build a numbered list with

\begin{verbatim}
We proceed as follows:
\begin{enumerate}
\item We write the list source text.\texttt{\pause}
\item We add the pause markers.\texttt{\pause}
\item We run pdf\TeX{}.\texttt{\pause}
\item We run PP\textsf{ower}4.\texttt{\pause}
\end{enumerate}
\end{verbatim}

And there we have it!

The first PDF file generated will have a single page with a small colored block where each \texttt{\pause} command appears. (This may be suppressed if the package is loaded with the \texttt{nomarkers} option.) The resulting PDF file after post-processing will have 5 pages in place of this one, each one with an additional item. As the presenter steps through the pages, each new line appears to be added to a single page, until the full page is complete.

\textbf{Building with transitions}

Rather than having the new line simply popping up out of nowhere, one can select the type of page transition at any step by giving a modified \texttt{\pause} with the name of one of the transition declarations from page 344. For example \texttt{\pauseHBr\textit{inds}} is the same as \texttt{\pause} plus \texttt{HBr\textit{inds}} except that the transitions between main pages are not affected. Note that
\texttt{\textbackslash pauseReplace} is the default, causing the new text simply to appear. These remain in effect for the following \texttt{\textbackslash pause} breaks until countermanded.

There is a \texttt{\textbackslash pauseGlitter\{angle\}} command as the equivalent to \texttt{\textbackslash pageTransitionGlitter\{angle\}}, but all others just attach the declaration name to \texttt{\textbackslash pause}.

\textbf{Building with levels}

Even more sophistication can be added by assigning level numbers to the chunks of text between the \texttt{\textbackslash pause} markers. In this way the page can be built up in any order, parts can be made to disappear or to be replaced, and the footline text at the bottom of the page can be visible from the beginning.

The level number corresponds to the sequence number of the views of the built-up page. Thus a chunk of text (that which comes between \texttt{\textbackslash pause} commands) with level \textit{n} will first appear in view \textit{n}. By default, the level number starts at 1 and increases by 1 for each \texttt{\textbackslash pause}. To ensure that the page footline is at level 1, we can write the last line of the above example as

\begin{verbatim}
And there we have it!\texttt{\textbackslash pause\textbackslash pauselevel\{=1\}}
\end{verbatim}

The \texttt{\textbackslash pauselevel} command sets the level number for the chunk in which it is located. There are several possibilities for its argument.

\begin{itemize}
\item \texttt{\{=n\}} sets the level number absolutely to \textit{n};
\item \texttt{\{=+n\}} increases it by \textit{n}; bear in mind that the preceding \texttt{\textbackslash pause} has already incremented it by 1;
\item \texttt{\{=-n\}} decreases it by \textit{n};
\item \texttt{\{=n -d\}} sets level number to \textit{n} and makes the change in level number with \texttt{\textbackslash pause} to be \textit{−d}; this would be done with say \texttt{\{=10 -1\}} to make the chunks numbered backwards from 10, causing the page to be built from the bottom up;
\item \texttt{\{:m\}} sets the maximum level to \textit{m}; at higher levels, the text vanishes; with \texttt{\{=3 :4\}} the chunk is visible only at levels 3 and 4; the \texttt{\{:m\}} may also be set relatively as \texttt{\{:+m\}} or \texttt{\{:-m\}}.
\end{itemize}

It is also possible to give multiple level specifications, as

\begin{verbatim}
\texttt{\textbackslash pause\textbackslash pauselevel\{=2 :2, =5 :6\}Text\textbackslash pause}
\end{verbatim}

to cause 'Text' to appear at level 2, to vanish for 3 and 4, and to reappear only for 5 and 6.

The level number may not be less than 1. Zero and negative numbers are treated as 1.
With some tricky playing around with level numbers and with the \TeX overlap commands \rlap and \llap, it is possible to get text or images to be replaced. The two must be made to overlap, but at different level numbers. For example:

```
We now \rlap{alter}change this word
We now
\pause\pauselevel{=1 :1}\rlap{alter}\pause
change\pause\pauselevel{=1}
this word
```

The first line shows the text without the \pause markers, in which ‘alter’ overlaps to the right over ‘change’. The next lines show the same text with levels inserted: ‘alter’ belongs to level 1, and disappears after 1; ‘change’ belongs to 2 (automatic increment), and the remaining text is set back to level 1.

**Highlighting instead of building**

An alternative to building the page by pieces is to highlight the section of text being discussed. In this mode, the entire page is visible, but printed in a dull color, like gray, while the chunk of text at the current level is in a bright color. When switching to the next view (level), that text becomes dull, and the next chunk brightens.

To get this to work, one must inform PPower4 what the normal and highlight colors are to be, and one must further indicate what text should actually change. Clearly certain parts of the page are to remain as they always are, at least the head and footlines, and possible titles. This is done with

```
\pausecolors{textclr}{dullclr}{highclr}
```

where textclr is a dummy color that is not otherwise being used. Only text indicated in this color will participate in the highlighting. For example

```
\pausecolors{cyan}{gray}{red} ...
This is the \textcolor{cyan}{text to be highlighted}
by the ...
```

The ‘text to be highlighted’ will normally appear in gray, but will turn red for that view corresponding to its level number. And only for that view! At the next view, it returns to gray. Note that the color cyan never appears; it is only a marker for PPower4. (Well, it does appear in the first PDF output, before post-processing.)

The highlighting mode is activated with \pausehighlight which makes all text at all levels visible, but with this color switching feature. The build mode can be reactivated with \pausebuild. In build mode, the
color switching still takes place, but the text does not appear until the right view is reached. It will be highlighted only for this first view.

One can add the word highlight to the argument of \pauselevel to indicate a chunk that should always be visible even in build mode, and which participates in the color switching. In highlight mode, this has no effect.

**Linking to first view**

Making an internal link with the \hyperref commands \hypertarget and \hyperlink (page 247) causes the target to be on the completed built up page, on the last view. However, one probably wants to link to the first view, before the page is built up. The pp4link package can assist here. It defines the commands

\toplink\{name\}\{text\} and \toptarget\{name\}

to establish text as a link to the target name.

This package tries to load \hyperref on its own, without any options. If you try loading \hyperref afterwards, with options, you will get a message about conflicting options and the options will be ignored. Therefore, load pp4link after \hyperref.
Letters

In addition to the three standard \LaTeX{} document classes, there is a fourth one named \texttt{letter} for formatting correspondence. As provided, this class is intended for private letters without any frills such as letterheads or business reference codes. However, local modifications are possible.

We first present the standard \LaTeX{} \texttt{letter} class, and then demonstrate how a house style can be written using our own institute style as an example. As always, it is highly recommended that each of these modifications be given its own file name so that \texttt{letter.cls} refers only to the provided \LaTeX{} version.

\subsection*{16.1 The \LaTeX{} letter class}

The \texttt{letter} document class is meant for writing letters. A single input file may contain the text for more than one letter and recipient, all from the same sender. Address labels may also be printed automatically if one wishes. Most of the normal \LaTeX{} commands function as usual within the \texttt{letter} class. One exception, however, is the sectioning commands, which will lead to the error message \texttt{Undefined control sequence} when issued. It actually makes little sense to try to divide a letter up into chapters, sections, etc. On the other hand, there are a number of special commands that may be applied only within this style.

The input text for a letter begins as for every \LaTeX{} document with

\begin{verbatim}
\documentclass[options]{letter}
\end{verbatim}

in which all the options listed in Section 3.1 may be given for \texttt{options} except for \texttt{twocolumn} and \texttt{titlepage} which hardly make any sense within a letter.

Every letter must contain the name and address of the sender, which are set for all the letters in one file by including the commands

\begin{verbatim}
\address{sender_address}
\signature{sender_name} or \name{sender_name}
\end{verbatim}
in the preamble. The \textit{sender.address} normally consists of several lines, separated by \\
commands, as in the example

\begin{verbatim}
\address{Max-Planck-Institut f"ur Aeronomie\\
Max-Planck-Str. 2\\
D--37191 Katlenburg--Lindau\Germany}
\end{verbatim}

The entry in the \textit{name} command will be used in the return address in the letterhead, if such has been programmed. The entry in the \textit{signature} command will be printed at the end of the letter below the space left blank for the writer's signature. If \textit{signature} has not been specified, the \textit{name} entry is inserted here instead. This allows a more formal \textit{name} to be used for the return address and a different form, perhaps with multiple lines, for the signature, as in

\begin{verbatim}
\name{Prof.\ M.\ Ostmann}
\signature{Martin Ostmann\Project Leader}
\end{verbatim}

When the above commands are issued in the preamble, they remain valid for all the letters in the document, except for those letters that contain new versions of these commands. Thus one letter might have a different \textit{signature} from the others. The scope of these entries extends only to the end of the environment in which they were called (see Section 8.5.4).

Two other sender entries are possible in standard \LaTeX\ letter class. They are intended to be employed in local modifications for a house style. The idea is that if \textit{address} is \textit{not} called, the preprogrammed company letterhead which might also contain the sender's room and/or telephone number is generated. Thus the commands

\begin{verbatim}
\location{room.number} \textit{and } \telephone{tel.number}
\end{verbatim}

are provided. In the standard \textit{letter} class, the entries \textit{room.number} and \textit{tel.number} are printed at the bottom of the first page only if \textit{address} is not issued.

The preamble may also contain the \textit{pagestyle} command with the usual entries plain, empty, or headings. The first is the default, putting the page number centered at the bottom of all pages after the first. The \textit{headings} page style adds the recipient's name, the date, and page number in a line at the top of all pages after the first.

After the preamble commands, the actual text begins as in all \LaTeX\ files with the command \texttt{\begin{document}}. The text consists of one or more letters, each enclosed in a \texttt{letter} environment with the syntax:

\begin{verbatim}
\begin{letter}\{recipient\} text of letter \end{letter}
\end{verbatim}

where \texttt{recipient} stands for the name \textit{and} address of the recipient, divided into several lines separated by \\
commands.

\begin{verbatim}
\begin{letter}\{Mr Donald J. Burns\}
 Ontario Institute of Physics\}
\end{letter}
\end{verbatim}
The text of letter normally begins with the command \texttt{\opening} and ends with \texttt{\closing}, between which the body of the letter appears, mixed with whatever \LaTeX\ commands are desired. The syntax of these two commands is

\begin{verbatim}
\opening{dear}
\closing{regards}
\end{verbatim}

where \texttt{dear} is the salutation, such as Dear Mr Tibs, and \texttt{regards} stands for the terminating text, for example Yours sincerely,. The \texttt{\opening} command could also contain other text, for example a subject entry line, with the true salutation as part of the following body text.

\LaTeX\ places the sender's name and address in the upper right corner of the first page with the current date set right justified below it. Then the recipient's name and address appear at the left margin, followed by the salutation and the body of the letter. The letter ends with the terminating text and the sender's name or signature left justified from the center of the line with sufficient vertical space between them for the handwritten signature.

After the \texttt{\closing}, a number of other commands may appear as part of the letter. One is \texttt{\cc} to produce a copy distribution list:

\begin{verbatim}
\cc{name1 \ \ name2 \ \ ... }
\end{verbatim}

The text ‘cc:’ (or more correctly, the text defined in \texttt{\ccname}) is printed at the left margin, followed by an indented list of names of the copy recipients.

The second additional command is \texttt{\encl} for making a list of enclosures:

\begin{verbatim}
\encl{enclosure1 \ \ enclosure2 \ \ ... }
\end{verbatim}

The word ‘encl:’ (actually the text in \texttt{\enclname}) is printed at the left margin and then the list of enclosures.

Finally, the command \texttt{\ps} may be used to add a postscript after the signature. The command itself does not generate any text, nor does it take an argument. The postscript text is everything located between the \texttt{\ps} and \texttt{\end{letter}} commands.

Normally the letter is dated automatically with the current date. However, if it is desired that the letter be back-dated, or that the date be otherwise fixed in the text, it may be set with

\begin{verbatim}
\date{date_text}
\end{verbatim}

in which case \texttt{date_text} appears where the current date would be placed.

A letter file may contain any number of \texttt{\letter} environments, one per letter. As stated already, when \texttt{\address}, \texttt{\name}, and \texttt{\signature} have
A sample letter produced with the standard \texttt{letter} class.

Max-Planck-Institut für Aeronomie
Max-Planck-Str. 2
D-37191 Katlenburg-Lindau
Germany

September 8, 2003

\texttt{\LaTeX}proof Ltd
P.O. Box 123
9876 Wordtown
Textland

Dear Sir;

We are most pleased to be able to answer your request for information about the use of \LaTeX{} for general text processing at a scientific institute.

1. After some initial trepidation on the part of the secretarial staff, which was mainly due to the first experience with a non-WYSIWYG text system, the system is now fully accepted and appreciated.

2. Much to the surprise of many secretaries, they find that they are able to set the most complicated mathematical formulas in a reasonably short time without difficulties. The same applies to the production of detailed tables.

3. Creating cross-references and keyword indices no longer causes horror, even when the author is well known for demanding constant changes.

4. Finally, the high quality appearance of the output has assisted in winning acceptance for \LaTeX{} in our house.

An additional positive note is the ability to write business letters readily, making use of the \texttt{letter} class provided with \LaTeX{}. In our institute, we have designed a special version to print our own letterhead, saving the need to have special letter paper printed.

Yours truly,

Patrick W. Daly

encl: Listing of our \texttt{mpletter.cls}
Sample output

cc: H. Kopka
B. Wand
been specified in the preamble, they remain in force for all the letters in the file. It is possible to alter any one or more of these sender entries by reissuing the command within one of the letters, before the \opening command, but then this change is valid solely for that letter. If both \name and \signature have been declared, it is the latter that is printed below the signature space.

The first page contains no page number. Subsequent pages have either a centered page number at the bottom (default) or a heading line with recipient, date, and page number at the top (page style headings).

The sample letter on the facing page was generated with the following input text.

\documentclass[a4paper,11pt]{letter}
\name{Dr P. W. Daly}
\address{Max--Planck--Institut f"ur Aeronomie\ 
Max--Planck--Str.\ 2\ 
D-37191 Katlenburg--Lindau\, Germany}
\signature{Patrick W. Daly}
\date{September 8, 2003}
\begin{document}
\begin{letter}{\TeX proof Ltd\, P.\, O. Box 123\ 
9876 Wordtown\, Textland}
\opening{Dear Sir;}
We are most pleased to be able to answer your request for information about the use of \LaTeX{} for general text processing at a scientific institute.
\begin{enumerate}
\item After some initial trepidation on the part of the secretarial . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
in winning acceptance for \LaTeX{} in our house.
\end{enumerate}
An additional positive note is the ability to write business . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . special paper printed.
\closing{Yours truly,}
\encl{Listing of our \texttt{mpletter.cls}\, Sample output}
\cc{H. Kopka\, B. Wand}
\end{letter}
\end{document}

By adding the command \makelabels in the preamble, the user can print out address stickers after all the letters have been output. The entries for the addresses are taken from the recipients' names and addresses in the argument to the letter environment. The standard \LaTeX\ letter class is designed for a page of labels \(4\frac{1}{4} \times 2\) inches, ordered in two columns. This could be altered for other formats. An address sticker
without a corresponding letter may be printed by including an empty \texttt{letter} environment of the form

\begin{letter}\{recipient\}\end{letter}

16.2 A house letter style

The sample letter above demonstrates the possibilities of the standard \LaTeX\ letter class. The height and width of the text may easily be altered with appropriate declarations in the preamble. The use of explicit English words and date style is no problem for other languages since they are all contained in special commands that can be redefined.

The \texttt{letter} class has been designed so that if the \texttt{\address} command is omitted, a company letterhead will be printed instead. This presupposes that the file has been reprogrammed at the local installation for this purpose. Each employee using this house letter style will have certain personal entries to make, such as his or her room and/or telephone number. These commands have been provided for in the standard \texttt{letter} class.

We have such a house letter style at our institute which we will illustrate here. However, it was found necessary to add some more personal entries, such as ‘Our Ref.’, ‘Your Ref.’, and email addresses. In addition, a command to print out ‘Subject:’ has also been added.

To distinguish our local house style from that of standard \LaTeX, we have named it \texttt{mpletter}. It contains most of the features of \texttt{letter}, since it in fact reads in that class file and then makes its alterations and additions. The \texttt{\address} command is not necessary, since all letters are printed with the institute letterhead, including its address. The recipient’s name and address are taken from the argument of the \texttt{letter} environment and are vertically centered in the space provided in the letterhead.

The writer’s name and telephone extension are entered with

\begin{verbatim}
\name{author}  \text{and}  \telephone{ext_number}
\end{verbatim}

which, if given in the preamble, apply to all the letters in the file. If different letters have other authors, these commands must be made in the appropriate \texttt{letter} environment \texttt{before} the \texttt{\opening} command. New entry commands specific to \texttt{mpletter} are

\begin{verbatim}
\yref{your_code}
\ymail{your_date}
\myref{our_code}
\subject{subj_text}
\end{verbatim}

which produce the words

\begin{verbatim}
Your Ref.:  Your letter of.:  Our Ref.:  Subject:
\end{verbatim}
Dear George,

Thank you for your inquiry about the latest version of the \TeX{} installation and additional packages.

The entire \TeX{} installation, with binaries and of course \LaTeX{}, along with a large number of contributed packages, is distributed annually by the \TeX{} Users Group, on their \TeX{}live CD.

I am sending you a copy of the current version of this CD, as you requested. In a separate directory named \texttt{bibtex} you will find the special bibliography formatting package files mentioned in ‘A Guide to \LaTeX’. I hope you will find these of use.

Do not hesitate to get in touch with me again if you have any further questions about the installation or running of the package.

Regards,

Patrick W. Daly

encl: 1 CD-ROM with \TeX{}live contents

cc: H. Kopka
properly positioned under the letterhead together with the corresponding text argument. If any of these commands are missing, their text will not appear in the letter.

Since ours is a German institute, there is an option \texttt{german} that translates all these words into their German equivalents. The entry commands have the same names, however.

As in the standard \texttt{letter} class, the current date is printed automatically, but may be changed to any desired text with the command

\begin{verbatim}
\date{date_text}
\end{verbatim}

if one wishes to back-date a letter or to fix the dating within the letter file itself. This is often handy if one only keeps electronic copies of the letters for the record. Otherwise, when a hard copy is run off months later, it will appear with the new current date and not with the original one.

The entry \texttt{author} that is given as the argument to the \texttt{name} command appears in the letterhead as the name of the sender. It will also appear below the space left for the signature, unless it is overridden by the \texttt{signature} command that specifies an alternative form of the name for this purpose.

The sample letter on the previous page has been generated with our house style, using the following input text.

\begin{verbatim}
\documentclass[12pt]{mpletter}
\name{Dr P. W. Daly} \signature{Patrick W. Daly}
\myref{PWD/sib}
\date{May 10, 2003}
\subject{\LaTeX{} information}
\telephone{279} \internet{daly}
\ymail{April 28, 2003} \yref{GFM/sdf}
\begin{document}
\begin{letter}{Mr George Murphy\35 Waterville Rd.\Centertown, Middlesex\United Kingdom}
\opening{Dear George,}
Thank you for your inquiry about the latest version of the \LaTeX{} installation and additional packages.

Do not hesitate to get in touch with me again if you have any further questions about the installation or running of the package.

\closing{Regards,}
\encl{1 CD-ROM with \TeX{} live contents}
\cc{H. Kopka}
\end{letter}
\end{document}
\end{verbatim}
The first page of our institute letter appears as shown, without a page number. If the text must continue beyond one page, the next page will have the heading:

MAX–PLANCK–INSTITUT FÜR AERONOMIE

To Mr George Murphy  May 10, 1998  Page 2

The recipient's name that appears in this heading is taken from the first line of the recipient argument in the \begin{letter}. This argument is split up by the \LaTeX processing so that the first line is contained in the command \toname, and the rest of the lines in \toaddress. The words 'To' and 'Page' are in the standard commands \headtoname and \pagename, and may be changed by an appropriate language adaptation option as shown in Section D.4.2.

16.3 A model letter customization

Adapting the letter.cls class file to the requirements of a company style should present few difficulties to an experienced \LaTeX programmer. Even a normal user may be able to make the necessary changes with the help of the example in this section.

We present here the class file mpletter.cls that was used to produce the sample letter on page 357. It makes heavy use of the \LaTeX programming features described in Appendix D. In order to understand it, one should be familiar with Section D.2. It will not be necessary to make any changes to the file letter.cls itself, since all modifications are in a separate class file that inputs the original.

The new class file is to be called mpletter.cls. It begins by specifying the \TeX format that it requires, and by identifying itself:

\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{mpletter}

It will be necessary to execute conditionals, so we will need the ifthen package described in Section 8.3.5. We will want a flag to decide if the letter is to be in German or not, determined by an option. Create the flag and define the option german to set it:

\RequirePackage{ifthen}
\newboolean{@german}
\setboolean{@german}{false}
\DeclareOption{german}{\setboolean{@german}{true}}

All other options that are valid in the standard letter class will also be accepted here, so simply pass them on to that class with the default option. Then process all options before loading letter itself with the a4paper option. We only have A4 paper in our institute.
This completes the preliminaries. So far, we have read in the standard letter class, along with the package ifthen, and defined a new option that is not present in the original class. Otherwise, all options and functions are unchanged, so far.

We now define the new 'name' commands to contain language-sensitive text, such as 'Subject', which are not provided for in the basic class. The actual definitions will be executed by two commands \englishnames and \germannames.

The last lines test whether the flag @german has been set (by the option german), and if so, the package german is loaded, and the German names are defined, otherwise the English names. The package german already translates the standard names commands \toname, \headtoname, and \pagename, so they are not included in \germannames.

Having settled the language problem, we now attack those commands for entering extra information in the header. Each of these stores its text argument in an internal command for future use. First the internal storage commands must be created.
16.3. A model letter customization

\newcommand\ymail[1]{\renewcommand\@ymail{\ymailname: #1}}
\newcommand\myref[1]{\renewcommand\@myref{\myrefname: #1}}
\newcommand\subject[1]{\renewcommand\@subject\{\subjectname: #1\}}
\newcommand\internet[1]{\renewcommand\@internet{#1}}
\INTERNET@linmpi.mpg.de

Set the dimensions of the text on the page and its margins. These numbers are appropriate for A4 paper (which does make the a4paper option superfluous).

\setlength{textheight}{215mm} \setlength{textwidth}{160mm}
\setlength{oddsidemargin}{0pt} \setlength{evensidemargin}{0pt}
\setlength{topmargin}{-20pt} \setlength{headheight}{12pt}
\setlength{headsep}{35pt}

The next step is to define some fixed fonts that are needed for the letterhead. We refer to the fonts with their explicit NFSS attributes; in this case they are all Computer Modern sans serif fonts in various sizes. These fonts will not change if totally different families are used for the body of the letter.

\DeclareFixedFont{xviisf}{OT1}{cmss}{m}{n}{17}
\DeclareFixedFont{xsf}{OT1}{cmss}{m}{n}{10}
\DeclareFixedFont{viiisf}{OT1}{cmss}{m}{n}{8}

The letterhead itself is divided into two fields, the left one containing the institute name in large letters, the right one the address in a smaller font. Below the first horizontal line, the left field displays the name and address of the recipient, positioned to fit in the window of an envelope, and the right one has the personal data of the letter writer, name, extension, computer address. The widths of these fields are established.

\newlength{leftfield} \setlength{leftfield}{117mm}
\newlength{rightfield} \setlength{rightfield}{43mm}

The total width of these two fields equals 160 mm, which is the same as textwidth.

Next, we place the institute name and address in several saved boxes.

\newsavebox{FIRM}\sbox{FIRM}
{\parbox[t]{\leftfield}
 \{xviisf MAX--PLANCK--INSTITUT F"UR AERONOMIE\}}
\newsavebox{firm}\sbox{firm}
{\xsf MAX--PLANCK--INSTITUT F"UR AERONOMIE}
\newsavebox{firmreturn}

\newsavebox{firmreturn}
Using these boxes as building blocks, we put together the actual head and foot of the letterhead page in two further save boxes.

The box \firmhead is fairly clear: it is a \parbox of width \textwidth containing the boxes \FIRM and \firmaddress side by side, with a line below. The foot \firmfoot is also a \parbox of the same width, but containing three columns of general institute information, set in \tabular environments.

It now remains to have the head and foot boxes placed on the first page. In the letter class, there is a special page style named firstpage that is always invoked for the first page of a letter. This must be redefined.

\renewcommand{\ps@firstpage}
{\setlength{\headheight}{41pt}\setlength{\headsep}{25pt}%
 \renewcommand{\oddhead}{\usebox{\firmhead}}%
 \renewcommand{\oddfoot}{\raisebox{-20pt}[0pt]\usebox{\firmfoot}}%
This page style must (re)define the commands \@oddhead and \@oddfoot, which are always inserted at the top and bottom of odd pages, to place our special \firmhead and \firmfoot. The even pages are unimportant, since the first page is always odd. It is necessary to enlarge \headheight and \headsep to allow the big boxes to fit in.

Subsequent pages are set with the headings or plain page styles. We want to modify the former to include the firm address once more.

\renewcommand{\ps@headings}
{\setlength{\headheight}{41pt}\
\renewcommand{\@oddhead}
{\parbox{\textwidth}{\usebox{\firm}\[5pt\]
\slshape \headtoname{} \toname\hfill\@date\hfill
\pagename{} \thepage\
\rule[3pt]{\textwidth}{1pt}}}
\renewcommand{\@oddfoot}{}
\renewcommand{\@evenhead}{\@oddhead}
\renewcommand{\@evenfoot}{\@oddfoot}

One small problem remains: the first time one of these page style commands is executed, the head and foot commands may not yet exist, causing \renewcommand to complain. We ensure that they are there at the start by predefining them with \providecommand (Section 8.3.1).

\providecommand{\@evenhead}{}
\providecommand{\@oddhead}{}
\providecommand{\@evenfoot}{}
\providecommand{\@oddfoot}{}

Now make headings the active page style.

\pagestyle{headings}

There is only one last thing to do, and that is to redefine the opening command that prints the recipient's address and the salutation. We add a bit more, including the personal data of the sender, as well as the reference information. The address goes in the left field, the personal data to the right. The references go in a line below the rule, followed by the subject line. These entries are tested first, and are only included if they are not blank. Several stored entry commands used here are part of the standard letter class, such as \toname and \toaddress. The \@date entry is either \today or whatever text was stored with \date.

\renewcommand{\opening}[1]{\thispagestyle{firstpage}\
\parbox[t]{\leftfield}
{\usebox{\firmreturn}\\n\parbox[b][3.5cm][c]{\leftfield}{\toname\toaddress}}%\n\parbox[t]{\rightfield}
{\fromname
\ifthenelse{\equal{\telephonenum}{}}{\{}\{}\Tel.: \telcode-\telephonenum}
The result of this formatting can be seen on page 357. It should be possible to modify the coding as needed for other organizations without too much difficulty.

The field for the recipient's name and address has been positioned so that it will appear in the address window of an envelope when properly folded. The smaller return address will also be visible through this window. The printing of extra address stickers is thus superfluous.

As an exercise, the user should design a document class \texttt{pletter} for writing personal letters. With the tips for formatting company letters, it should not be difficult to add a letterhead of the form

Sheila Joan McDonald

Tel.: 234-9871
31 Maple Drive
Willowtown

\textit{Hint: The name has been printed with the font cmdunh10 scaled \texttt{magstep4}, declared with \texttt{newfont}. Such a \texttt{pletter} class must remain in the personal directories of the user at a computer center. It would be rather embarrassing if everyone made use of the same personal letterhead.}
Appendices
The New Font Selection Scheme (NFSS)

When \TeX{} and \LaTeX{} were invented, the fonts available for them were very limited in number. For this reason, the original \LaTeX{} had an inflexible system of defining the fonts that were to be used, since it was not obvious that one might want to change them. The association between the high-level font commands such as \texttt{\textbackslash large} and \texttt{\textbackslash bf} and the external font name that is ultimately selected was rigidly fixed internally.

Today there are many additional fonts available, some of which should be used alongside the standard CM fonts, and others that should replace them altogether. For example, the Cyrillic fonts of Section 12.4.2 should be added parallel to the Latin fonts, but to make them operate automatically under the \LaTeX{} size commands was a complex procedure. (We know: we have done it!) Similarly, installing PostScript fonts involved calling intricate interface macros.

Another problem was the behavior of the font style and size commands. As explained in Section F.2.1, each of the font declarations \texttt{\textbackslash rm}, \texttt{\textbackslash bf}, \texttt{\textbackslash sc}, \texttt{\textbackslash sl}, \texttt{\textbackslash it}, \texttt{\textbackslash sf}, and \texttt{\textbackslash tt} activates a particular font in the current size, overriding the previous declaration. One can select either a bold or an italic font, but there is no way to select a bold, italic one. Furthermore, selecting a new size automatically switches to \texttt{\textbackslash rm}, an upright, Roman, medium font. That is, the attributes cannot be selected independently of each other.

In 1989 Frank Mittelbach and Rainer Schöpf proposed a New Font Selection Scheme (NFSS) for \LaTeX{} and a preliminary test package was ready in early 1990. A second release (NFSS2) was published mid-1993 with many substantial changes. With the official release of \LaTeX{}2ε in June 1994, NFSS became firmly entrenched in the new standard. The new font declarations and commands are described in Sections 4.1.3 and 4.1.4. Here we explain the usage in more detail; in Section A.3 we describe how new sets of fonts may be installed.
Appendix A. The New Font Selection Scheme (NFSS)

A.1 Font attributes under NFSS

According to the NFSS scheme, every character set can be classified by five attributes called *encoding*, *family*, *series*, *shape*, and *size*, which may be selected with the commands

\fontencoding{encode} \fontfamily{fam} \fontseries{wt_wth} \fontshape{form} and \fontsize{sz}{line_sp}

The *encode* attribute was new to the second release of NFSS. It specifies the layout of the characters within the font. Possible values for it are listed in Table A.1. It is unlikely that one would want to change encoding within a document, except to activate Cyrillic fonts perhaps. This feature is more for \LaTeX\ programmers to install new fonts on the system.

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Description</th>
<th>Sample Font</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>OT1</td>
<td>Original text fonts from Knuth</td>
<td>cmr10</td>
<td>492</td>
</tr>
<tr>
<td>OT2</td>
<td>Univ. Washington Cyrillic fonts</td>
<td>wncy10</td>
<td>497</td>
</tr>
<tr>
<td>T1</td>
<td>The Cork (DC/EC) fonts</td>
<td>ecrm1000</td>
<td>500</td>
</tr>
<tr>
<td>OML</td>
<td>\TeX\ math letter fonts</td>
<td>cmmi10</td>
<td>495</td>
</tr>
<tr>
<td>OMS</td>
<td>\TeX\ math symbol fonts</td>
<td>cmsy10</td>
<td>495</td>
</tr>
<tr>
<td>OMX</td>
<td>\TeX\ math extended fonts</td>
<td>cmex10</td>
<td>496</td>
</tr>
<tr>
<td>U</td>
<td>Unknown coding</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

The argument *fam* in *\fontfamily* denotes a basic set of font properties. For the Computer Modern fonts, listed in Section G.2.2, all the serif fonts belong to the family *cmr*. The family *cmm* includes all the sans serif fonts while *cmtt* contains the typewriter fonts. A number of special decorative fonts are the only members of their families. Table A.3 on page 370 lists the CM fonts according to the family and other attributes.

**Note:** The font attribute *family* has no relationship to the primitive \TeX\ concept of the same name. A \TeX\ family consists of three fonts of different sizes for use in math formulas as normal characters, and first- and second-level indices.

The argument *wt_wth* in *\fontseries* designates the weight (=boldness) and width of the characters. These are specified by 1 to 4 letters as shown in Table A.2 on the opposite page.

The argument for *\fontseries{wt_wth}* consists of the letter or letters for the weight, followed by those for the width. Thus *ebsc* indicates weight *extrabold* and width *semicondensed* while *bx* means weight *bold* and width *expanded*. The letter *m* is omitted when combined with any non-normal weight or width; if both are to be normal, it is sufficient to give *m* alone.
### Table A.2: The NFSS series attributes

<table>
<thead>
<tr>
<th>Weight class</th>
<th>Width class</th>
<th>Width class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ultralight</td>
<td>Ultracondensed 50%</td>
<td>uc</td>
</tr>
<tr>
<td>Extralight</td>
<td>Ultracondensed 50%</td>
<td>ec</td>
</tr>
<tr>
<td>Light</td>
<td>Condensed 75%</td>
<td>c</td>
</tr>
<tr>
<td>Semilight</td>
<td>Semiexpanded 87.5%</td>
<td>sc</td>
</tr>
<tr>
<td>Medium (normal)</td>
<td>Medium 100%</td>
<td>m</td>
</tr>
<tr>
<td>Semibold</td>
<td>Semiblended 112.5%</td>
<td>sx</td>
</tr>
<tr>
<td>Bold</td>
<td>Expanded 125%</td>
<td>x</td>
</tr>
<tr>
<td>Exrobold</td>
<td>Extraexpanded 150%</td>
<td>ex</td>
</tr>
<tr>
<td>Ultrabold</td>
<td>Ultraexpanded 200%</td>
<td>ux</td>
</tr>
</tbody>
</table>

In \fontshape, the argument form is one of the letter combinations \(n\), \(it\), \(sl\), or \(sc\) for selecting normal (or upright), italic, slanted, or small caps.

The \fontsize attribute command takes two arguments, the first sz being the point size of the font (without the dimension pt explicitly given) and the second line_sp being the vertical spacing from one baseline to the next. The second argument becomes the new value of \baselineskip (Section 3.2.4). For example, \fontsize{12}{15} selects a font size of 12 pt with interline spacing of 15 pt. (The second argument may be given a dimension, such as 15pt, but pt is assumed if no dimension is stated.)

Once all five attributes have been set, the font itself is selected with the command \selectfont. The new feature here is that the various attributes are independent of one another. Changing one of them does not alter the others. For example, if the selection

\fontfamily{cmr} \fontseries{bx} \fontshape{n} \fontsize{12}{15}

has been made for an upright, bold, expanded, Roman font of size 12 pt and interline spacing 15 pt, then when \fontfamily{cmss} is later selected for a sans serif font, the attributes weight and width bx, form n, and size 12(15pt) remain in effect when the next \selectfont is issued.

Alternatively, all attributes but the size may be specified and the font activated immediately with the command

\usefont{code}{family}{series}{shape}

The table on the next page (by F. Mittelbach and R. Schöpf) lists the classification of the Computer Modern character sets according to the attributes \fontfamily, \fontseries, and \fontshape. That there are so many attribute combinations without a corresponding CM font may appear to be a weakness in the NFSS system, but it must be recalled that it is designed for the future. It may also be employed with the PostScript...
fonts, which are becoming ever more popular, to exploit their complete variability.

Formally it is possible to set any combination of attributes; however, there may not exist any font matching all the attributes selected. If that is the case, then when \selectfont is called, \LaTeX issues a warning stating which font has been activated in its place. The font size attribute of the \fontsize command may normally take on values of 5, 6, 7, 8, 9, 10, 10.95, 12, 14.4, 17.28, 20.74, and 24.88, but other values may also be added. The second argument, the interline spacing, may take on any value since it is not something intrinsic to the font itself.

With the \begin{document} command, \LaTeX sets the five attributes to certain preset default values. These are normally standard encoding OT1, family cmr, medium series m, normal shape n, and the base size selected. The user may change these initial values within the preamble, or they might be set differently by a special option, say when a PostScript font has been selected.

### A.2 Simplified font selection

The attribute commands \fontencoding, \fontfamily, \fontseries, \fontshape, and \fontsize, together with the command \selectfont, are the basic tools in the New Font Selection Scheme. The user need not
employ these commands directly, but rather may make use of the higher-
level declarations presented in Sections 4.1.2 and 4.1.3. In fact, a font
declaration like \itshape is defined as \fontshape{it}\selectfont.

The high-level commands to select font sizes are:

\begin{verbatim}
\tiny (5pt) \normalsize (10pt) \LARGE (17.28pt)
\scriptsize (7pt) \large (12pt) \huge (20.74pt)
\footnotesize (8pt) \Large (14.4pt) \Huge (24.88pt)
\small (9pt)
\end{verbatim}

The sizes listed for the commands are those when 10pt (the default) has
been selected as the basic size option in the \documentclass command;
for 11pt and 12pt, they will all scale accordingly.

The family declarations and their standard family attribute values are:

\begin{verbatim}
\rmfamily (cmr) \sffamily (cmss) \ttfamily (cmtt)
\end{verbatim}

which are the three Computer Modern families, Roman, Sans Serif, and
Typewriter (Section G.2.2).

The series declarations and their initial attribute values are:

\begin{verbatim}
\mdseries (m) \bfseries (bx)
\end{verbatim}

meaning that only a medium and a bold extended series are provided as
standard.

Finally, the shape declarations and their attribute values are:

\begin{verbatim}
\upshape (n) \itshape (it)
\slshape (sl) \scshape (sc)
\end{verbatim}

to select upright, slanted, italic, and SMALL CAPS.

Note that there are no high-level declarations for the encoding attri-
butes. This is because there is normally no need to change encoding
within a document. An exception might be to use Cyrillic fonts (coding
OT2), in which case one could define

\begin{verbatim}
\newcommand{\cyr}{\fontencoding{OT2}\selectfont}
\newcommand{\lat}{\fontencoding{OT1}\selectfont}
\end{verbatim}

to be able to switch back and forth more conveniently.

The family, shape, and series attributes may be reset to their standard
values at any time with the \normalfont command, which also activates
that font in the current size.

For each of the above font attribute declarations there is also a cor-
responding font command (Section 4.1.4) that sets its argument in that
font. Thus \textit{text} is almost the same as \{\itshape text\}, the
only difference being that the command also contains the italic correction
automatically. The complete list of such commands is:
A.3 Installing fonts with NFSS

In this section, we wish to elaborate on the internal workings of NFSS, showing how a set of attributes are associated with a particular font and how special symbols are assigned to their proper positions within various encoding schemes.

A.3.1 Default attribute values

We implied in Section A.2 that the font attribute declarations like \itshape are defined as \fontshape{it}\selectfont, whereas in fact they make use of certain default attributes. Thus the true definition of \itshape is \fontshape{\itdefault}\selectfont

The default commands available are

Family: \rmdefault \sfdefault \ttdefault
Series: \mddefault \bfdefault
Shape: \updefault \itdefault \sldefault \scdefault

It is \itdefault that is defined to be the attribute it.

It is also necessary to define the standard attributes chosen when the command \normalfont is issued. These are contained in the four defaults

\encodingdefault \familydefault \seriesdefault \shapedefault

All this may sound like a complicated route linking the high-level commands to a particular font. However, it does provide flexibility and modularity. The author only needs to know that three families, two series, and four shapes are available, and does not care what they really are. A programmer defines these with the defaults at a lower level.

An example of how all the standard fonts may be replaced by PostScript ones, by simply redefining the three family defaults, is shown on page 379. Such redefinitions are much simpler than trying to alter the font declarations themselves, including \normalfont. Those definitions are in fact much more complex than implied here, whereas the default commands really are as simple as indicated.
A.3. Installing fonts with NFSS

A.3.2 Defining font commands

There are a number of commands available for defining new font declarations and commands. These are intended mainly for \LaTeX package programmers but may be used in a normal document as well.

\texttt{\DeclareFixedFont{\texttt{cmd}}{\texttt{code}}{\texttt{family}}{\texttt{series}}{\texttt{shape}}{\texttt{size}}}
defines \texttt{cmd} to be a declaration that selects the font of the specified attributes. It is rigidly fixed in all attributes. This is equivalent to \texttt{\newfont} except that the font is determined by attributes and not by name.

\texttt{\DeclareTextFontCommand{\texttt{cmd}}{\texttt{font specs}}}
defines \texttt{cmd} to be a font \textit{command} that sets its argument according to \texttt{font specs}. This command is used internally to define all the font commands like \texttt{\textbf}, which is defined with \texttt{font specs} as \texttt{\bfseries}.

\texttt{\DeclareOldFontCommand{\texttt{cmd}}{\texttt{text specs}}{\texttt{math specs}}}
defines \texttt{cmd} to be a font \textit{declaration} that may be used in math mode in the manner of \LaTeX 2.09: as a declaration, not a command. It is useful for defining commands to be compatible with the old version, but should be avoided. For example, \texttt{\it} is defined with

\texttt{\DeclareOldFontCommand{\it}{\normalfont\itshape}{\mathit}]

A.3.3 Mathematical alphabets

The font that has been activated for text processing does \textit{not} influence the characters and their fonts in math mode, since special mathematical symbol fonts are used for this purpose. If a formula is to appear in bold face, the command \texttt{\boldmath} (Section 5.4.9) must be issued, which remains in effect until it is countermanded by \texttt{\unboldmath}. Both of these declarations must be made outside of math mode.

These commands may still be employed in the same way under NFSS. However, the internal math font selection command is

\texttt{\mathversion{vers name}}
in which the argument \texttt{vers name} currently takes on values \texttt{normal} and \texttt{bold}. The declarations \texttt{\unboldmath} and \texttt{\boldmath} are defined in terms of this command. It is planned that special package files will become available to allow additional sets of math symbols.

On the other hand, mathematical alphabet commands may be issued within math mode itself, to set letters as symbols in particular fonts (Section 5.4.2):

\texttt{\mathrm \mathcal \mathnormal \mathbf \mathsf \mathit \mathtt}

These are all \textit{commands} operating on arguments rather than \textit{declarations}, unlike the \LaTeX 2.09 equivalents.

New math font alphabets may be defined by the user. For example, to define a slanted math font \texttt{\mathsl}, give

\texttt{\texttt{\DeclareOldFontCommand}{\mathsl}{\normalfont\itshape}{\mathit}]

\texttt{\DeclareOldFontCommand}{\it}{\normalfont\itshape}{\mathit}
which means that the new math font command \texttt{\textbackslash maths1} selects that font in family \texttt{cmr} with weight \texttt{m} and shape \texttt{s1}, which, with the normal font definitions, is font \texttt{cmsl} in the appropriate size. However, this font will be selected in \texttt{all} math versions, whereas it would be more suitable if a bold font were selected when \texttt{\textbackslash mathversion\{bold\}} is in effect. This is accomplished by adding to the definition of \texttt{\textbackslash maths1}

\texttt{\textbackslash SetMathAlphabet\{\maths1\}\{bold\}\{OT1\}\{cmr\}\{bx\}\{s1\}}

which redefines \texttt{\maths1} exceptionally for math version \texttt{bold} to be a font with weight \texttt{bx}. For the normal font definitions, this is \texttt{cmbxsl} in the current size.

New math versions can be created with

\texttt{\textbackslash DeclareMathVersion\{\texttt{vers.name}\}}

and the fonts belonging to it are determined by issuing \texttt{\textbackslash Set...} commands for each math alphabet or symbol font.

### A.3.4 Mathematical symbol fonts

Mathematical symbols must be defined in a totally different manner from text characters: they bear a command name (like \texttt{\textbackslash alpha}), may come from various fonts, behave differently depending on type, and can appear in different sizes. Under \LaTeX{} 2.09, the symbol names were fixed to the Computer Modern math fonts, but NFSS offers more flexibility for additional (or replacement) symbol fonts.

A symbol font name is declared with the command

\texttt{\textbackslash DeclareSymbolFont\{sym.fnt.name\}\{code\}\{family\}\{series\}\{shape\}}

which associates the name \texttt{sym.fnt.name} with the given set of attributes. This name is not a command, but rather an internal designation for use in defining symbols. The selected font is valid for all versions, but if a different font is to be associated with the same name under other versions,

\texttt{\textbackslash SetSymbolFont\{sym.fnt.name\}\{version\}\{code\}\{family\}\{series\}\{shape\}}

may be used to redefine \texttt{sym.fnt.name} for that one version.

The standard \LaTeX{} setup declares

\texttt{\textbackslash DeclareSymbolFont\{operators\}\{OT1\}\{cmr\}\{m\}\{n\}}
\texttt{\textbackslash DeclareSymbolFont\{letters\}\{OML\}\{cmm\}\{m\}\{it\}}
\texttt{\textbackslash DeclareSymbolFont\{symbols\}\{OMS\}\{cmsy\}\{m\}\{n\}}
\texttt{\textbackslash DeclareSymbolFont\{largesymbols\}\{OMX\}\{cmex\}\{m\}\{n\}}

the sequence of which is important for reasons that are built deeply into \TeX{} itself.

Once the symbol font names have been defined, they may be used to construct math alphabets and various types of symbols.
\DeclareSymbolFontAlphabet{\mathalph}{sym_fnt_name}

defines \mathalph to be a math alphabet based on the font with the internal name sym_fnt_name. This command is to be preferred over \DeclareMathAlphabet if a symbol font exists with the proper attributes for the math alphabet.

The primary command for defining symbols is

\DeclareMathSymbol{\symbol}{\type}{sym_fnt_name}{\pos}

which makes \symbol print the symbol in position pos of font sym_fnt_name. The pos is a number, in decimal (10), octal ('12) or hexadecimal ('0A) representation. The type specifies the functionality of the symbol and is one of

\begin{itemize}
  \item \texttt{\texttt{\texttt{\texttt{\texttt{\mathord}}}} an ordinary symbol}
  \item \texttt{\texttt{\texttt{\texttt{\texttt{\mathop}}}} a large operator like \sum}
  \item \texttt{\texttt{\texttt{\texttt{\texttt{\mathbin}}}} a binary operator like \times}
  \item \texttt{\texttt{\texttt{\texttt{\texttt{\mathrel}}}} a relational operator like \geq}
  \item \texttt{\texttt{\texttt{\texttt{\texttt{\mathopen}}}} an opening bracket like \{}
  \item \texttt{\texttt{\texttt{\texttt{\texttt{\mathclose}}}} a closing bracket like \}}
  \item \texttt{\texttt{\texttt{\texttt{\texttt{\mathpunct}}}} punctuation}
  \item \texttt{\texttt{\texttt{\texttt{\texttt{\mathalpha}}}} an alphabetic character}
\end{itemize}

Math alphabet commands operate only on the symbols of type \mathalpha; other types always produce the same symbol, for a given math version, within all math alphabets.

Similarly, math accent commands are established with

\DeclareMathAccent{\cmd}{\type}{sym_fnt_name}{\pos}

where type is either \mathord or \mathalpha; in the latter case, the symbol changes with the math alphabet.

Math delimiters and radicals can appear in two different sizes. They are set up with

\begin{itemize}
  \item \texttt{\texttt{\texttt{\texttt{\texttt{\DeclareMathDelimiter}}}} \cmd \type sym_fnt1\pos1 sym_fnt2\pos2}
  \item \texttt{\texttt{\texttt{\texttt{\texttt{\DeclareMathRadical}}}} \cmd sym_fnt1\pos1 sym_fnt2\pos2}
\end{itemize}

which define \cmd to print the smaller variant from position pos1 of font sym_fnt1 and the larger from position pos2 of sym_fnt2.

The sizes have not been specified in any of the above math declarations. This is because there are normally four different sizes available, depending on the math style, as explained in Section 5.5.2. However, these sizes must somehow be specified. This is done with

\DeclareMathSizes{text}{math_text}{script}{sscript}

where the four arguments are numbers giving a point size. When the normal text font is size text pt, textstyle will be in math_text size, textstyle in script, and textscriptstyle in sscript size. For example,

\DeclareMathSizes{10}{10}{7}{5}

All the \Declare... and \Set... commands may only be called in the preamble.
A.3.5 Addressing the attribute values

In some programming situations, it is necessary to make use of the current values of the attributes without knowing what they are. These are stored in the internal commands

\f@encoding \f@shape \tf@size
\f@family \f@size \sf@size
\f@series \f@baselineskip \ssf@size

The values of these commands should never be changed directly. However, they may be used to test if they possess a certain value. Since they all contain the character @ in their names, they may only be used in class or package files, and not in the main document file.

A.3.6 Defining fonts under NFSS

Under NFSS, fonts are specified within a document by giving the attributes required and then calling \selectfont. How is the set of font attributes then associated with a particular external font name such as those found in Appendix G? This is done by means of font definition commands which are usually stored in files with extensions .def and .fd.

First the declaration

\DeclareFontEncoding{code}{text}{math}

sets up a new encoding attribute named code; whenever a text font of this encoding is selected, text is executed in order to redefine accent commands or other things that are coding dependent; similarly math is called for every math alphabet of this encoding. It is possible to define default text and math with

\DeclareFontEncodingDefaults{text}{math}

This allows general text and math mode settings to be declared with this command, while more specialized ones, which are executed afterwards, appear in \DeclareFontEncoding.

If no font can be found for the specified attributes,

\DeclareFontSubstitution{code}{family}{series}{shape}

declares the values of the attributes that should be substituted; substitutions are made in order of shape, series, then family; the encoding is never substituted. If even this fails to find a valid font, then

\DeclareErrorFont{code}{family}{series}{shape}{size}

determines which font is to be used as a last resort.

A new family with a specified encoding scheme is established with

\DeclareFontFamily{code}{family}{option}
where \textit{option} is a set of commands that may be executed every time a font of this family and encoding is selected.

The main font-defining declaration that associates external font names with font attributes is

\begin{verbatim}
\DeclareFontShape{\OT1}{cmr}{m}{n}
  {<5> <6> <7> <8> <9> <10> <12> gen * cmr
   <10.95> cmr10
   <14.4> cmr12
   <17.28> <20.74> <24.88> cmr17}{}
\end{verbatim}

where \textit{option} is additional commands that are executed when one of these fonts is selected.

The \textit{font_def} contains a series of size/font associations, each consisting of a size part, a function, an optional argument, and a font argument. For example,

\begin{verbatim}
\DeclareFontShape{\OT1}{cmr}{m}{n}
  {<5> <6> <7> <8> <9> <10> <12> gen * cmr
   <10.95> cmr10
   <14.4> cmr12
   <17.28> <20.74> <24.88> cmr17}{}
\end{verbatim}

states that the medium series, normal shape members of the \textit{cmr} family are to be represented by external fonts \textit{cmr5} \dots \textit{cmr12} for sizes 5–12 pt, by \textit{cmr10} scaled to 10.95 pt for sizes near 11 pt, and so on. If the specified size is not present, the nearest size within certain limits is taken instead.

The size part consists of numbers in angle brackets, representing point sizes. The brackets may also contain ranges, as \textit{<-10>} means all sizes up to, but excluding, 10 pt, \textit{<10–14>} means 10 pt to less than 14 pt, and \textit{<24–>} indicates 24 pt and higher. The possible functions are:

\begin{itemize}
  \item \textbf{(empty)} loads the named font scaled to the requested point size; if an optional argument in square brackets precedes the font name, it acts as an additional scaling factor;
  \begin{verbatim}
  <11> [ .95 ] cmr10 \quad \text{would load \textit{cmr10} scaled to 95\% of 11 pt;}
  \end{verbatim}

  \item \textbf{gen *} generates the font name by appending the point size to the font argument;
  \begin{verbatim}
  <12> gen * cmr \quad \text{loads \textit{cmr12};}
  \end{verbatim}

  \item \textbf{genb *} generates the font name by appending the point size times 100 to the font argument (for DC and EC fonts, page 501);
  \begin{verbatim}
  <14.4> genb * ecss \quad \text{loads \textit{ecss1440};}
  \end{verbatim}

  \item \textbf{sub *} substitutes a different font whose attributes are given in the font argument as \textit{fam/ser/shp};
  \begin{verbatim}
  <-> sub * cmtt/m/n \quad \text{this is best used when there is no font of the required attributes; a message is output to the monitor and to the transcript file;}
  \end{verbatim}
\end{itemize}
Appendix A. The New Font Selection Scheme (NFSS)

\texttt{subf *} is like the empty function, but issues a warning that an explicit substitute font has been loaded;

\texttt{fixed *} loads the specified font at its normal size, ignoring the size part; if an optional argument is given, it is the point size to which the font is scaled, as in
\begin{verbatim}
\end{verbatim}

All the above functions may be preceded by an \texttt{s} to suppress messages to the monitor. Thus silent \texttt{sub *} is \texttt{ssub *}, and silent empty is \texttt{s *}.

As another example, consider the definition of bold, italic, typewriter, for which there is no font in the Computer Modern collection:
\begin{verbatim}
\DeclareFontShape{OT1}{cmtt}{bx}{it}{<-> ssub * cmtt/m/it }{}
\end{verbatim}

This substitutes (silently) for all sizes (\texttt{<->}) the medium italic typewriter attributes. Which fonts those are is determined by a \texttt{\DeclareFontShape} command with those attributes.

The font definition commands may be issued in a package file, or even in the document itself. However, the normal procedure is to store each of the \texttt{\DeclareFontEncoding} commands in a file named \texttt{codeenc.def} (for example, \texttt{ot1enc.def} for the OT1 encoding), and to place the commands \texttt{\DeclareFontFamily} and \texttt{\DeclareFontShape} in a file whose name consists of the encoding and family designations plus the extension \texttt{.fd}. For example, the shape specifications for encoding OT1 and family cmr are to be found in \texttt{ot1cmr.fd}. When a coding and family combination that is not already defined is selected, \LaTeX{} tries to find the appropriate \texttt{.fd} file for input. It is therefore not necessary to input such files explicitly, for they are loaded automatically as required.

However, it is important that the encoding be declared beforehand. If it is not already known in the current format, \texttt{\DeclareFontEncoding} must be issued, either explicitly, or by loading the \texttt{codeenc.def} file. One way to do this is to invoke the standard package \texttt{fontenc}, as for example
\begin{verbatim}
\usepackage[OT2,T1]{fontenc}
\end{verbatim}

where the desired codings are listed as options in square brackets, the last of which is made current.

The normal user will never need to worry about such problems. However, \LaTeX{} programmers will find things considerably easier for them. For example, to install PostScript fonts under NFSS is almost trivial. The common PostScript fonts are already defined as separate families in \texttt{.fd} files of their own; for example, \texttt{ot1ptm.fd} associates PostScript Times fonts to a family named \texttt{ptm}. A package to activate these fonts is supplied under the name \texttt{times.sty} containing essentially the lines

\begin{verbatim}
\usepackage[OT2,T1]{fontenc}
\end{verbatim}
A.3. Installing fonts with NFSS

This makes Times `ptm` the default Roman family, invoked with the command `\rmfamily`, Helvetica `phv` the default sans serif family (called by `\sffamily`), and Courier `pcr` the default typewriter family (activated by `\ttfamily`). The default attribute for bold face is defined to be `b` instead of the regular `bx`, since bold extended is not provided by these fonts.

Another two examples of the usefulness of NFSS are the Cyrillic fonts of the University of Washington (Section 12.4.2) and the extended EC fonts with the Cork encoding (Section G.4.3). Both of these may be activated within a document simply by selecting another encoding, `OT2` for Cyrillic, `T1` for the EC fonts. (These encodings must first be declared, for example with the `fontenc` package as illustrated on the facing page.)

A.3.7 Encoding commands

In \LaTeX, special characters and accents are addressed by means of commands, such as `\O` to print the Scandinavian letter Ø. The position of this character in the font tables depends on the encoding (character 31 in `OT1` and 216 in `T1`), so that it is necessary to redefine all such symbol commands when the encoding is altered. This is carried out with the help of certain encoding commands, which normally appear in the `codeenc.def` file along with the `\DeclareFontEncoding` command.

To define a command that functions differently in the various encodings,

\begin{verbatim}
\ProvideTextCommand{\cmd}{code}[narg][opt]{def}
\DeclareTextCommand{\cmd}{code}[narg][opt]{def}
\end{verbatim}

are available and behave just like `\providecommand` and `\newcommand` except `\cmd` has the definition `def` only when the encoding `code` is active. Thus `\cmd` may have different definitions for each encoding.

\begin{verbatim}
\DeclareTextSymbol{\cmd}{code}{pos}
\end{verbatim}

defines `\cmd` to print the character in the font position `pos` when encoding `code` is active.

\begin{verbatim}
\DeclareTextAccent{\cmd}{code}{pos}
\end{verbatim}

defines `\cmd` to be an accent command, using the character in font position `pos` as the accent symbol, when encoding `code` is active.

\begin{verbatim}
\DeclareTextComposite{\cmd}{code}{letter}{pos}
\DeclareTextCompositeCommand{\cmd}{code}{letter}{def}
\end{verbatim}

define the action of command `\cmd` followed by the single `letter` either to print the character in font position `pos` or to execute the definition `def`. These declarations are most useful with the T1 encoding, where many accented letters are separate...
symbols on their own (Section G.4.3). Thus \'{e} in OT1 prints an acute accent
(character 19) over the letter e, while in T1, it prints the single character in
position 233. This behavior is achieved with

\DeclareTextAccent{'}{OT1}{19}
\DeclareTextComposite{'}{T1}{e}{233}

The command must already have been defined for the encoding, either with
\DeclareTextAccent or with \DeclareTextCommand; in the latter case, it must
be defined to take a single argument.

All of the above definition commands create new commands for a specific
encoding. If the defined commands are invoked in some other encoding, an
error message is issued. Default definitions may be provided for all unspecified
encodings with

\DeclareTextCommandDefault{\cmd}[narg][opt]{def}
\ProvideTextCommandDefault{\cmd}[narg][opt]{def}
\DeclareTextAccentDefault{\cmd}{code}
\DeclareTextSymbolDefault{\cmd}{code}

where the first two create a default definition that applies to all unspecified
encodings, while the second two declare which encoding is to be taken as the
default.
In this appendix, we describe how the \TeX program and \LaTeX files are installed, how they are organized, what their roles are. Throughout this book we have given examples of contributed packages, but here we list those packages and other files that belong to the 'kernel', the essential installation. In Section B.6, we explain how \LaTeX ticks, what happens during a processing run, and what all the various file types mean.

B.1 Installing \LaTeX

We explain first \LaTeX installations in general before looking at the particular one provided on the enclosed \TeXLive CD.

B.1.1 \TeX implementations

One must have the \TeX program and its auxiliaries (\metafont, font files) before \LaTeX can be set up on top of it. Installing \TeX is a somewhat daunting experience, but thankfully it need not be done very often and there are many ready-to-run implementations available for practically every computer operating system. These can be obtained from CTAN under the directory systems (Figure B.4 on page 390) or from the \TeX Users Group (www.tug.org).

The system delivered with the \TeXLive CD is that developed by Karl Berry known generically as Web2c. This name derives from the fact that it converts Donald Knuth's original \TeX source files from his Web language (no relation to the World Wide Web) into the C programming language for subsequent compilation with a C compiler. The te\TeX implementation, by Thomas Esser, applies Web2c to Unix machines, while Fabrice Popineau's fp\TeX is the Win32 (Windows 95, 98, NT, 2000, XP) version.

Another excellent \TeX implementation for Windows is Mik\TeX by Christian Schenk, available from CTAN under systems/win32/miktex/.
The implementation for DOS, em\TeX{} by Eberhardt Mattes is still available from CTAN, but is no longer appropriate for modern PCs. We used to produce the first two editions of this book and found it superb for its time.

Commercial packages for \TeX{} also exist, offering graphics display of the output while you type: for Windows (Y&Y Inc., www.YandY.com/) and Textures for Macintosh (Blue Sky Research, www.bluesky.com/). A 'what-you-see-is-what-you-get' \TeX{} system is Scientific Workplace, which can be purchased from MacKichan Software Research (www.tcisoft.com/).

It should be stressed that the \TeX{} and pdf\TeX{} programs in all the above implementations are identical (for the same version number). They provide ready-to-run executable programs from the same source code, which are necessarily different for different computer types. There are also minor variations as to how the programs are run.

The \LaTeX{} files are all ASCII so they are fully compatible with all systems. One uses the already installed \TeX{} or pdf\TeX{} program to generate the \LaTeX{} formats (Section B.1.3). Even this task is normally done for you by the above implementations.

A graphics interface to \TeX{} and \LaTeX{} is not absolutely essential, but today it is almost unthinkable to do without one. This is an editor program for writing and managing the \LaTeX{} source files, and calling the various programs (\LaTeX{}, Bib\TeX{}, MakeIndex) and invoking viewers by clicking icons. For Unix, the \texttt{emacs} is normally used for this; for Windows, the two editor programs mentioned in Section 1.6.2, Winshell and WinEdt can be highly recommended.

B.1.2 The \TeX{}Live CD

The \TeX{} Users Group, TUG issues a set of CDs each year to its members containing the latest \TeX{} and \LaTeX{} installations, with executable files for Win32, Linux, Macintosh, and all known flavors of Unix. As of the 7th issue in 2002, it was necessary to split the contents over 2 CDs, separating the Unix versions from the others. Each CD is otherwise complete, missing only the executables of the other one.

The \TeX{}Live CDs are maintained by Sebastian Rahtz who has kindly agreed to allow the CD for Windows, Linux, Mackintosh to be included with this book, as it stands in June 2003. It thus corresponds to something between the 7th and 8th regular issues.

The startup window in Figure B.1 appears when the CD is inserted into your computer. If the autorun function does not work, you can run the \texttt{autorun.exe} program directly from the CD.

You then proceed with the installation by clicking Install on the upper bar. You then follow the instructions on the various pages that then appear, making selections as you go. One choice is whether to install to run from the CD, or to install to the hard disk. In the former case,
you have the entire installation available with little lost of storage, but will need to have the CD mounted in the player whenever you run \LaTeX. Processing time will also be slower.

If you elect to install only part of the system to hard disk, you may still add further packages and support programs later by clicking maintenance. Or you may run \texttt{texsetup --maintenance} which does the same thing bypassing the Welcome window. Running \texttt{texsetup --help} produces a list of all the options.

You may also click Documentation→View TeXLive doc to open the instruction manual for \TeXLive, with details for all systems. (Or you can go directly to \texttt{\textbackslash texmf\doc\tl\doc\english} and select \texttt{live.pdf} or \texttt{live.html}.)

Documentation for most of the packages and collections are available on the CD under \texttt{\textbackslash texmf\doc\...}, if the author has provided a description. We often refer to such manuals throughout this book, indicating more precisely where they are to be found. By clicking Documentation→Run TeXDocTK, you may open a documentation browser (Figure B.2) to assist finding descriptions for any module.

The \TeXLive installation program will do all the housekeeping for you,
Figure B.2: The \TeX{}Live documentation browser

placing files in the right directories, setting up font mapping files, and generating the formats. However, there might still be times when you want or have to do some of this yourself. This we explain in the next sections.

### B.1.3 Making the \LaTeX{} format

Adding or updating \LaTeX{} on to an existing \TeX{} installation by hand is relatively straight forward. One must obtain the source files, which on \TeX{}Live are located at `\texttt{\texmf\source\latex\base}` or on CTAN at `\texttt{tex-archive/macros/latex/base/}`. At CTAN, instruction files with extension `.txt` are available for the various implementations, such as `miktex.txt` and `web2ctex.txt`, as well as the general instruction file `install.txt`.

These source files are a set of packed files consisting of a large number of `.dtx` files, the integrated source and documentation files. These are unpacked by running \texttt{initex} on the batch job file `unpack.ins`. This generates the necessary `.cls`, `.sty`, `.clo`, and other files. It also constructs the fundamental file `latex.ltx` which is needed to produce the format file. On CTAN, the unpacked files are already provided in a parallel directory `unpacked`.

The next step should be to run \texttt{initex} on the `latex.ltx` file to produce the \LaTeX{} format `latex`. However, before doing that, there are a number of configuration aspects to be considered. At several points during the processing, certain files are read in, but if a file of the same name but extension `.cfg` exists, it is loaded instead. It is this device that permits one to configure the final format for local conditions or desires. The possible configuration files are:

- `texsys.cfg` offers the possibility of adding some very machine-specific adjustments for older versions of \TeX{} or for some peculiarities of the \TeX{} installation; information is to be found in the specific `.txt` file, or in `ltdirchk.dtx`;


fonttext.cfg is loaded in place of fonttext.ltx if it is present; this defines the fonts that are to be available for processing text; details can be found in fontdef.dtx;

fontmath.cfg is loaded in place of fontmath.ltx if it is present; this is the equivalent of fonttext.cfg for math fonts;

preload.cfg is loaded in place of preload.ltx if it is present; this determines which fonts are preloaded into the format; a number of other preload files may be extracted from preload.dtx, any one of which may be renamed to .cfg to be implemented;

hyphen.cfg is loaded in place of hyphen.ltx if it is present; this specifies the hyphenation patterns and their assignments; by default, the patterns in hyphen.tex are loaded into language 0; the babel system provides such a configuration file (Section 11.1) which must be used if babel is to be incorporated into the format, in which case, language.dat must be tailored to local requirements.

The only configuration file that you are likely to want to change is hyphen.cfg, especially if you are going to be using \LaTeX{} for languages other than American English.

Once any configuration files have been set up and located where \TeX{} can read them, \texttt{initex} may be run on \texttt{latex.ltx} with

\begin{verbatim}
initex latex.ltx  or  
tex -ini latex.ltx
\end{verbatim}

The resulting \texttt{latex.fmt} must be placed where format files are read. For \TeX{}Live as installed by \TeX{}Live, this is in \texttt{texmf-var\textbackslash web2c\}. Other unpacked files must be moved to \texttt{texmf\textbackslash tex\textbackslash latex\textbackslash base\}; these are

\begin{verbatim}
latexbug.tex, testpage.tex, lablst.tex, idx.tex, nfssfont.tex, small2e.tex, sample2e.tex, and docstrip.tex,
\end{verbatim}

and all files with extensions .cls,.clo,.sty,.fd,.def, and .cfg.

Files with extension .ist are moved to \texttt{texmf\textbackslash makeindex\textbackslash base\}.

To make up the \LaTeX{} format with pdf\TeX{}, run instead

\begin{verbatim}
pdfinitex -fmt=pdflatex latex.ltx  or  
pdftex -ini -fmt=pdflatex latex.ltx
\end{verbatim}

This processes \texttt{latex.ltx} base file once more, but names the resulting format file \texttt{pdflatex.fmt} rather than \texttt{latex.fmt}. This is necessary because the format files may only be used with the program that generated them, so the two must be distinguished even though they both are essentially the same thing. Move this format too to the format file directory.

The actual commands to invoke \TeX{} or pdf\TeX{} with \LaTeX{} (what we normally call ‘running \LaTeX{} or pdf\LaTeX’) are
The system should have shortcuts named `latex` and `pdflatex` that translate to the above.

The `web2c` system, for which `fpTeX` is the Win32 implementation, has a different method. It stores all the necessary configuration information in a file named `texmf.cnf` located in the `texmf\web2c` directory. This sets up environments according to the name of the program being run, and in particular, adds the format name automatically for a `TeX`-like program. Thus a `TeX` program named `latex` will use the `latex.fmt` format file if none is specified. However, there is no real program named `latex`, so on Unix systems, an alias with this name is established to point to the `tex` executable program. On Win32, the actual `TeX` executable code is in a dynamic link library `tex.dll` and very small files named `tex.exe`, `latex.exe` exist to use this one library. These `.exe` files are in fact identical, and are really only there to ensure that different formats are invoked automatically when they are called. Similarly, there are `pdftex.dll`, `pdftex.exe`, and `pdflatex.exe`. If any additional formats are ever needed for these programs, one only needs to copy one of these `.exe` files to the new name.

### B.1.4 Updating the database

`TeX` works with a very large number of files, for classes, packages, fonts, formats, and so on. A set of environments are used to indicate where the program should search for these files. With `web2c`, these are all defined in the `texmf.cnf` configuration file mentioned above. The user normally does not have to worry about this. However, to speed up the search, a database is prepared giving the exact location of all the files in the search paths. This database will be created and updated automatically if installation programs are used, such as that for `TeXLive`.

If the user should add a new package or fonts collection by hand, without going through an installation program, it will be necessary to update this database. Under `web2c` (and `fpTeX`), this is carried out with the command

```
  mktexlsr
```

Under `MikTeX`, one must run

```
  configure --update-fndb
```

or click ‘refresh now’ in the `MikTeX` options program.

It is not necessary to update the database if existing files are being replaced. It is only the names and locations of the files that go into the database, not their actual versions or sizes.
B.2 Obtaining the Adobe euro fonts

As pointed out in Section 2.5.8, the type 1 euro currency symbol fonts from Adobe Systems Inc. that are used with the europs and eurosans packages cannot be distributed on CDs like \TeX\ Live, nor on CTAN, since Adobe reserves the exclusive right to distribution, even though free of charge. \TeX\ Live and CTAN do provide all other files needed, like the font metric .tfm and font definition .fd files.

You can obtain the Adobe euro .pfb files from www.adobe.com/type/eurofont.html, downloading the appropriate file for your computer type. They must be unpacked (on Windows, for example, you just execute the downloaded eurofont.exe) and then rename them as follows:

<table>
<thead>
<tr>
<th>Windows</th>
<th>PostScript</th>
<th>\TeX</th>
</tr>
</thead>
<tbody>
<tr>
<td>_1_____PFB</td>
<td>EuroSans-Regular</td>
<td>zpeurs.pfb</td>
</tr>
<tr>
<td>_1I_____PFB</td>
<td>EuroSans-Italic</td>
<td>zpeuris.pfb</td>
</tr>
<tr>
<td>_1B_____PFB</td>
<td>EuroSans-Bold</td>
<td>zpeubs.pfb</td>
</tr>
<tr>
<td>_1BI_____PFB</td>
<td>EuroSans-BoldItalic</td>
<td>zpeubis.pfb</td>
</tr>
<tr>
<td>_3_____PFB</td>
<td>EuroSerif-Regular</td>
<td>zpeur.pfb</td>
</tr>
<tr>
<td>_3I_____PFB</td>
<td>EuroSerif-Italic</td>
<td>zpeuri.pfb</td>
</tr>
<tr>
<td>_3B_____PFB</td>
<td>EuroSerif-Bold</td>
<td>zpeub.pfb</td>
</tr>
<tr>
<td>_3BI_____PFB</td>
<td>EuroSerif-BoldItalic</td>
<td>zpeubi.pfb</td>
</tr>
<tr>
<td>_2_____PFB</td>
<td>EuroMono-Regular</td>
<td>zpeurt.pfb</td>
</tr>
<tr>
<td>_2I_____PFB</td>
<td>EuroMono-Italic</td>
<td>zpeurit.pfb</td>
</tr>
<tr>
<td>_2B_____PFB</td>
<td>EuroMono-Bold</td>
<td>zpeubt.pfb</td>
</tr>
<tr>
<td>_2BI_____PFB</td>
<td>EuroMono-BoldItalic</td>
<td>zpeubit.pfb</td>
</tr>
</tbody>
</table>

The first column is the name as unpacked on Windows, the second the internal PostScript name, and the third, the name used by \TeX. The .pfb files must be renamed to those in the third column, and placed in \texttt{texmf/fonts/type1/adobe/euro} (or better in \texttt{texmf-local}).

The font map file zpeu.map must also be added to the lists of font maps for dvips, dvipdfm, and pdftex (Section 10.1.3). The \TeX\ Live installation does all this for you; you only need to get and store the .pfb files.

B.3 \TeX\ directory structure

The \TeX system, of which \LaTeX is only one part, albeit a large one, requires a very large number of files, for fonts, formats, classes, packages, bibliographies, and much more. Section B.6 lists the types of files and their extensions, at least for what concerns \LaTeX. Other flavors of \TeX have their own types in addition.
Appendix B. The \LaTeX{} Clockwork

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{tds_tree}
\caption{The TDS directory tree}
\end{figure}

In the early days, anyone setting up a \TeX{} installation would try to create order of some sort by distributing the files into different directories (also called \textit{folders}) according to their functions. However, everyone had their own ideas how to do this. Thus changing from one installation to another required a totally new road map.

The \TeX{} Users Group, TUG, commissioned a working group to prepare a recommended standard, which published its results in 1994. The latest version of this paper is on \TeX{}Live at \texttt{\texmf\doc\tds\} in various formats. The \TeX{} Directory Structure (TDS) described there can only be a recommendation, but it is fairly universally followed today. The \TeX{}Live CD certainly conforms to it.

Figure B.3 offers an overview of TDS. The top directory is named \texttt{texmf}, to demonstrate that it holds the entire \TeX{} and \texttt{METAFONT} systems. This may not be the absolute top directory on any installation. For example, if you install to a Windows hard disk, you might find it in \texttt{\program~files\texlive\texmf}. Furthermore, there might be parallel directories named \texttt{texmf-var} (where the system writes files it creates, like font .\texttt{pk} files) and \texttt{texmf-local} for any private packages you install yourself. The idea is to keep the main \texttt{texmf} tree intact, to let the \TeX{}Live installation program manage it. These parallel trees should have the same structure, but may be incomplete.

The subdirectories under \texttt{texmf} are the main classification of the subsequent files, sorted mainly by program, such as \texttt{Bib\TeX} (Section 14.1),
dvi\textit{pdfm} (Section 10.2.2), \textit{dvips} (Section 10.1.1), \textit{MakeIndex} (Section 9.4.3), \textit{METAFONT} (Section G.3), and pdf\TeX{} (Section 10.2.3). Others are

doc containing documentation of all sorts; its subdirectories are similarly structured to those under \texttt{tex}; most of the documentation files that we have referred to in this book are to be found here;

\texttt{fonts} is subdivided according to the types of files explained in Section G.1: \texttt{pk} for the pixel files (these usually go into \texttt{texmf-var}), \texttt{tfm} for the font metric files, \texttt{vf} for virtual font files, \texttt{source} for the \textit{METAFONT}.\textit{mf} sources, \texttt{type1} for the .\texttt{pfb} or .\texttt{pfa} PostScript outline font files; the contents of each of these directories are further sorted by supplier (adobe, public, ...), and then by font family (\texttt{times}, \texttt{cm}, ...);

\texttt{source} contains the delivery files for packages and collections; for \LaTeX{}, these are the .\texttt{dtx} and .\texttt{ins} files, found under \texttt{latex}\textbackslash base; similarly for packages, such as \texttt{latex}\textbackslash geometry containing \texttt{geometry.dtx} and \texttt{geometry.ins}; these are not needed any more once installed; the subdirectories under \texttt{source} mirror those under \texttt{tex};

\texttt{tex} is the main directory for files directly related to \TeX{}; some of its subdirectories are

\texttt{generic} for files that apply to all \TeX{} applications, like hyphenation patterns;

\texttt{latex} for all the \LaTeX{} related files; this is further subdivided into directories for individual packages; only a handful are shown in the figure; \texttt{base} contains the fundamental \LaTeX{} files, the kernel;

\texttt{plain} for Plain \TeX{}, which is often just called \TeX{} (Section 1.3.2).

This directory structure allows a user to locate documentation, source, running files for a package, collection, program, on all installations. The documentation and source files can readily be removed, or not copied at all from the CD, since they play no role during the processing. This structure makes it easy to remove such unnecessary branches of the tree.

**B.4 The CTAN server**

The easiest way to obtain the \LaTeX{} software not available on the \TeX{}Live CD is via one of the network servers supported by various educational institutions. They also provide the most up-to-date standard \LaTeX{} installation, \TeX{} extensions, programs for running \TeX{}, collections of drivers for different printers, hyphenation patterns for other languages, etc. Such a server is a treasure trove for the dedicated \TeX{} and \LaTeX{} user.
There are three main network servers for TeX and \LaTeX located in the United States, Great Britain, and Germany, forming the Comprehensive \TeX Archive Network, CTAN. They are not only kept up to date but also regularly exchange contributions among themselves. They are all reachable on the Internet.

<table>
<thead>
<tr>
<th>Country</th>
<th>Internet address</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>ftp://ctan.tug.org/tex-archive/</td>
</tr>
<tr>
<td>UK</td>
<td>ftp://ftp.tex.ac.uk/tex-archive/</td>
</tr>
<tr>
<td>Germany</td>
<td>ftp://ftp.dante.de/tex-archive/</td>
</tr>
</tbody>
</table>

In addition, there are many full and partial mirrors of these sites. See the TUG home page, http://www.tug.org, for details.

The directory structure is the same for all three servers. A sketch of the main parts of the directory tree is presented in Figure B.4.

The latest version of the \LaTeX source files may be found under the subdirectories base or unpacked, where in the latter the files are already unpacked and \texttt{latex.ltx} is ready. The \LaTeX extensions and general style files that have been contributed by other users are located in the subdirectory contrib/supported, with each contribution in a subdirectory of its own. Extensions offered by the \LaTeX Team are found in the required subdirectory, which include \texttt{babel}, \texttt{graphics}, \texttt{tools}, and so on.
B.5 Additional standard files

Throughout this book we have pointed out and explained many contributed packages that expand \LaTeX\ functionality. However, there are a number of service documents, additional classes, and packages that are delivered with the basic \LaTeX\ installation, found in texmf\tex\latex\base. We list them all here, although some have been mentioned in other sections.

B.5.1 Special documents

The standard installation includes a number of special 'documents', files with the .tex extension. These are:

small2e.tex a short sample document;

sample2e.tex a longer sample document;

lablst.tex (Section 9.2.1) for printing out all the cross-reference labels, with translations and page numbers; it asks interactively the name of the document that is to be so listed;

idx.tex (Section 9.4) lists all the \index entries for a document, ordered in two columns, by page; it asks interactively the name of the document that is to be so listed;

testpage.tex to test the positioning of the text on a page; this is more a test of the printer driver, to make sure that the margins are correct and that the scaling comes out properly;

nfssfont.tex is a document to print out a font table, sample text, and various other tests; it asks for the name of the font interactively; the command \help prints out a list of commands that can be given;

docstrip.tex (Section D.7.1) this is more a program than a document; it is the basic tool for unpacking operating files from their documented source files; it not only removes comments, but also includes alternative coding depending on various options.

B.5.2 Extra classes

In addition to the standard classes that have already been discussed (book, report, article, letter) the standard installation also contains

proc for producing camera-ready copy of conference proceedings; it is a variant of the two-column article class; it contains one extra command: \copyrightspace, which leaves blank space at the bottom of the first column for typing in a copyright notice;
Appendix B. The \LaTeX Clockwork

minimal a bare-bones class for testing and debugging;

slides (Section 15.1) the former Su\TeX, for producing presentation ma-
terial;

\ltxdoc (Section D.7.2) to produce class and package documentation.

B.5.3 Standard packages

The standard packages, some of which are described elsewhere, are:

alltt (Section 4.9.1) enables the alltt environment which is much like \verbatim except that \{\} behave as normal; this means that commands included in the text are executed and not printed literally;

doc (Section D.7.2) provides many special features for documenting \LaTeX packages;

exscale allows the mathematical symbols that appear in different sizes to scale with the font size option given in the \documentclass command; normally the size of these symbols is fixed, independent of the basic size of the text;

flafter (Section 7.2, page 171) ensures that floating objects (figures and tables) do not appear before their position in the text;

fontenc (Section A.3.6, page 378) declares font encodings by loading their .def files; the names of the encodings are listed as options;

graphpap (Section 13.1.5) provides the command \graphpaper which produces coordinate grids in the picture environment;

ifthen (Section 8.3.5) allows text or command definitions depending on the state of various conditionals; one may test the state of a boolean switch, the value of a counter or length, or the text definition of a command;

inputenc (Section D.5) allows a \LaTeX document file to be prepared with a system-dependent input encoding scheme for special characters;

latexsym loads the 11 special \LaTeX symbols in the lazy fonts (Section G.2.2), which were part of \LaTeX 2.09; this file is automatically read in in compatibility mode;

makeidx (Section 9.4.3) provides the commands that may be used with the MakeIndex program to generate a keyword index;

showidx (Section 9.4) makes the \index entries visible by printing them as marginal notes;
shortverb (Section 4.9.1) provides the commands `\MakeShortVerb` and `\DeleteShortVerb`, to designate (and remove) a certain character as the equivalent of `\verb` for printing literal text. The character, preceded by a backslash, is given as an argument to these commands. After `\MakeShortVerb{\|}` has been issued, `|\cmd{x}|` is equivalent to `\verb|\cmd{x}|`, producing `\cmd{x}`;

`syntonly` provides the command `\syntaxonly` which, when given in the preamble, suppresses all output to the `.dvi` file, but continues to issue errors and warnings; the job runs about four times faster than normal; this is used to check the processing when output is not immediately needed;

t1enc makes the T1 font encoding (Section G.4.3) the standard, for use with DC or EC fonts;

textcomp makes the special symbols in the text companion fonts available in text mode (Section G.4.4);

tracefnt is a diagnostic tool for checking NFSS font selections; the options given with `\usepackage` control its action:

`errorshow` suppresses all warnings and information messages to the monitor, which are still sent to the transcript file; only error messages are printed to the monitor;

`warningshow` prints warnings and error messages on the monitor; this behavior is much the same as when the package is not used at all;

`infoshow` (default option) sends font selection information to the monitor, which is normally only written to the transcript file;

`debugshow` writes considerably more information to the transcript file about every change of font; can produce a very large transcript;

`pausing` turns all warnings into error messages so that processing temporarily stops, awaiting user response;

`loading` shows the loading of external fonts.

**B.5.4 The tools packages**

The members of the \LaTeX{}3 Team, as individuals, have written a number of packages that they collectively make available; these are to be found in the `texmf\tex\latex\tools\` directory. Although contributions, they are considered to be part of the standard installation.

Many of these have been described elsewhere in this book, but we list them all here.
afterpage (Section 7.1 on page 171) permits commands to be saved and executed at the end of the current page. This is useful for forcing stubborn floats to be output right away, with \afterpage{\clearpage}. If \clearpage were issued directly, a new page would be inserted at that point. With \afterpage, the command is reserved until the end of the page.

array (Section 4.8.4 on page 106) is a re-implementation of the \texttt{tabular} and \texttt{array} environments with several additional features.

\texttt{bm} permits individual math symbols to be printed in bold face with command $\texttt{\bm{\text{sym}}}$. Thus $\alpha + \texttt{\bm{\beta}}$ produces $\alpha + \beta$ without recourse to the \texttt{\boldmath} command which would set the whole formula in bold face. There is also a command \texttt{\DeclareBoldMathCommand{\texttt{name}}{\texttt{sym}}} that defines \texttt{name} to be \texttt{\bm{\texttt{sym}}}.

calc re-implements the counter and length commands to allow 'normal' arithmetic with them, such as
\begin{verbatim}
\setlength{\mylen}{3cm + \textwidth}
\setcounter{page}{\value{section} * 3}
\parbox{\linewidth / \real{1.6}}{...}
\end{verbatim}

Note that the arithmetical expressions can be used not only to set values but also as arguments to commands and environments.

dcolumn (Section 4.8.4) requires the \texttt{array} package; it allows decimal point alignment in the \texttt{tabular} tables.

delarray (Section 4.8.4) requires the \texttt{array} package; it permits large bracketing symbols to be put around \texttt{array} environments, for making matrices of various sorts.

enumerate (Section 4.3.5, page 74) re-implements the \texttt{enumerate} environment so that an optional argument determines the numeration style.

\texttt{fileerr.dtx} when unpacked, produces a set of small files that may be used to reply to \TeX{} when a file is not found; this makes the responses similar to those for error messages; the files are named \texttt{h.tex}, \texttt{e.tex}, \texttt{s.tex}, and \texttt{x.tex}; thus, replying \texttt{x\langle\text{return}\rangle} loads the last one, which terminates the input.

\texttt{fontsmpl} is a package to output a sample text, accents, and special characters for a given font; with the accompanying file \texttt{fontsmpl.tex}, a family such as \texttt{cmr} can be printed with all attributes and encodings.

\texttt{ftnright} puts footnotes in the right-hand column in two-column mode.
hhline requires the array package; allows more flexibility in putting horizontal rules in tables.

indentfirst (Section 3.2.4 on page 46) indents the first paragraph of sections, which normally are not indented.

layout (Section 3.2.5 on page 47) defines the command \layout which draws a diagram of the current page format, with the values of the parameters printed.

longtable (Section 4.8.4 on page 108) makes tables that extend over several pages, with breaks occurring automatically.

multicol (Section 3.2.8) introduces the environment multicols which switches to multi-column output without a page break either before or after. It is called with

\begin{multicols}{num_cols}[header text][pre_space]
Text set in num_cols columns
\end{multicols}

The optional header text is printed in a single column across the top of the multi-column part. A \newpage is issued only if the remaining space on the current page is less than a certain value, stored in the length \premulticols; the optional length argument pre_space overrides this value if given. Similarly the length \postmulticols determines whether a new page is inserted at the end of the multi-column text. Column separation and possible rule are set by the standard \LaTeX lengths \columnsep and \columnseprule as for the \twocolumn command. The text in the multi-column format is balanced; that is, the columns are all equally long.

showkeys prints out all the cross-reference keys defined by \label and \bibitem and used by \ref, \pageref, and \cite; these are marginal and interlinear notes for a draft only, to check the cross-referencing.

somedefs allows only selected commands to be defined in a package, depending on options in \usepackage.

tabularx (Section 4.8.4) defines a tabularx environment which is like tabular*, a table of desired width, except that it is the column widths that expand and not the intercolumn spacing.

theorem provides extensions to the \newtheorem command for more flexible 'theorem-like' environments; it is very similar to the \AMS amsthm package (Section 12.3.1).
Appendix B. The \LaTeX{} Clockwork

\texttt{varioref} (Section 9.2.4) defines \texttt{\vref} and \texttt{\vpageref} commands, analogous to \texttt{\ref} and \texttt{\pageref}, which check if the referenced object is only one page away, and if so, print text like ‘on the previous page’; the actual text printed can alternate between two variants.

\texttt{verbatim} (Section 4.9.1) is a re-implementation of the \texttt{verbatim} environment that prevents memory overflow for long texts; it also defines a command \texttt{\verbatiminput} to input a file and to print its contents literally, as well as a \texttt{comment} environment for a block comment in the input text.

\texttt{xr} (Section 9.2.3) is a package to allow \texttt{\ref} to cross-reference \texttt{\label} commands from other documents.

\texttt{xspace} (Section 8.3.1 on page 186) contains a device to fix the problem of command names swallowing up the blank that follows them; thus the command \texttt{\PS} defined as

\begin{verbatim}
\newcommand{\PS}{PostScript\xspace}
\end{verbatim}

may be used as \texttt{\PS file} without having to terminate the \texttt{\PS} command with \texttt{\_} or \texttt{\{\}}.

B.6 The various \LaTeX{} files

A number of different files are used during the \LaTeX{} processing: some are read in while others are created to store information for the next run. They all consist of two parts:

\texttt{root\_name.extension}

For every \LaTeX{} document, there is a \texttt{main} file, whose root name is used when the program \LaTeX{} is called. It normally possesses the extension \texttt{.tex}, as do any other files that it might read in with \texttt{\input} or \texttt{\include} commands. (If they have a different extension, it must be given explicitly.)

The other files that are created during a \LaTeX{} processing run normally have the same root name as the main file but a variety of different extensions, depending on their functions. If the main file contains \texttt{\include} commands, there will be additional files created having the same root names as the included \texttt{.tex} files but with the extension \texttt{.aux}.

Some of these files are created with every \LaTeX{} run, whereas others appear only when certain \LaTeX{} commands have been issued, such as \texttt{\tableofcontents} or \texttt{\makeindex}. The \texttt{.aux} files as well as those produced by special \LaTeX{} commands may be suppressed by including the command \texttt{\nofiles} in the preamble. This command is useful if a document is being constantly corrected and reprocessed, so that the information in the special files is not yet finalized and may therefore be
dispensed with. Files created at the \TeX level of processing will always be
generated.

Other extensions describe files that contain formatting information,
or additional instructions (coding), or databases. The root part of these
file names is not associated with any document file. An example of this
type is the article.cls file that defines the article class.
The rest of this section contains a list of the \LaTeX file extensions with
a short description of the role they play in the \LaTeX processing.

.aux This is the auxiliary file written by \LaTeX, containing information for
cross-references as well as some commands necessary for the table
of contents and other lists. There will be one .aux file created (or
re-created) for the main file, in addition to one for every file read in
by an \include command.
The auxiliary .aux files may be suppressed by issuing \nofiles in
the preamble.

.bbl This file is not created by \LaTeX, but by the program \BIBTeX. It has
the same root name as the main file. \BIBTeX actually only reads the
.aux file for its information. The .bbl file is read into the next \LaTeX
processing run by the command \bibliography and produces the
list of literature references.

.bib Bibliographic databases have the extension .bib. \BIBTeX reads them
to extract the information it needs to generate the .bbl file. The
root name describes the database and not (necessarily) any text
file. The database name is included in the text by means of the
\bibliography command.

.blg This is the transcript file from a \BIBTeX run. The root name is the
same as that of the input file.

.bst This is a bibliography style file that is used as input to \BIBTeX to
determine the format of the bibliography. The name of the .bst
file is included in the text by means of the \bibliographystyle command.

.cfg Some classes permit local configurations for paper size or other
requirements by putting specifications into a .cfg file of the same
root name as the class. The ltxdoc class (Section D.7.2) is one of
these.

.clo This is a class option file, containing the coding for certain options
that might apply to more than one class. There is no special
command to input them.

.cls This is a class file, defining the overall format of the document. It
is read in by the \documentclass command. A .cls file may also
be input from another one with the \LoadClass command.
Additional definition files of various types carry the .def extension. Examples are `t1enc.def` (to define the T1 encoding) and `latex209.def` (for compatibility mode between \LaTeX\ 2\epsilon and 2.09).

These are documented source files for the \LaTeX\ installation. If a .dtx file is processed directly, the documentation is printed. This is an explanatory manual and/or a detailed description of the coding. The class, package, .ltx, .fd or other file(s) are extracted by processing the .dtx file with DocStrip (Section D.7.1), usually by running \LaTeX\ or \TeX\ on a supplied .ins file.

This is the \TeX\ output file, containing the processed text in a format that is independent of the output printer, hence a 'device-independent' file. It too is generated by every \TeX\ process run and cannot be suppressed by \nofiles. The .dvi file must be further processed by a printer driver program that converts it into the special commands for the desired output device (printer).

Files containing the NFSS commands associating external font names with font attributes (Section A.3.6) are font description files. The root names consist of the encoding and family designations, such as `ot1cmr.fd`. When a font of a given family is requested for the first time, the corresponding .fd file is automatically read in, if it exists.

Documented sources for .fd files have the extension .fdd, and are nothing more than a special type of .dtx file. They may be processed for the documentation, or have the .fd files extracted by DocStrip.

This extension belongs to a \TeX\ format file that has been created with the special program initex (Section B.1.3). Format files contain all the basic instructions in a compact coding for quick loading. \LaTeX\ is nothing less than \TeX\ run with the format `latex.fmt'. (Plain \TeX\ uses the format file `plain.fmt'.)

This file is only generated when the command \makeglossary exists in the preamble. It has the same root name as the main file and contains only \glossaryentry commands that have been produced by \glossary commands in the input text. This file will be suppressed in spite of the presence of \makeglossary if the command \nofiles is issued in the preamble.

This file is the glossary equivalent of the .ind file. It too is generated by the MakeIndex program, but with the .glo file as input. The input and output extensions must be explicitly given. There are no standard \LaTeX\ commands to read in the .gls file. The doc package uses it to record a change history.
.idx This file is only generated when the command `\makeindex` exists in the preamble. It has the same root name as the main file and contains only `\indexentry` commands that have been produced by `\index` commands in the input text. This file will be suppressed in spite of the presence of `\makeindex` if the command `\nofiles` is issued in the preamble.

.ilg This is the transcript file from a MakeIndex run. The root name is the same as that of the input file.

.ind This file is generated by the MakeIndex program, with the .idx file as input. It is read in by the `\printindex` command which is defined by the makeidx package. It contains a `theindex` environment, built up from the entries in the input file.

.ins To facilitate the DocStrip run on a .dtx file, the necessary commands (and options) are put into an `installation` file. The extraction is then accomplished simply by processing (with \TeX or \L\TeX) the appropriate .ins file.

.ist This is an index style file, containing style settings for MakeIndex. The root name reflects the style and has no relationship to any text file. For the doc package, the files gind.ist and gglo.ist are provided for specialized index formatting.

.lof This file contains the information for the list of figures. It behaves exactly like the .toc file except that it is opened by the command `\listoffigures` instead of by `\tableofcontents`.

.log This is the transcript file, containing the protocol of the \TeX processing run, that is, all the messages that were sent to the monitor during the run as well as additional information that can be of use to \TeX experts when errors occur. This file is always generated, even when `\nofiles` has been issued.

.lot This file contains the information for the list of tables. It behaves exactly like the .toc file except that it is opened by the command `\listoftables` instead of by `\tableofcontents`.

.ltx For \L\TeX installation, certain files that are inputs to the initex program bear the extension .ltx. They form the basis for generating the format latex.fmt. The main such file is named latex.ltx.

.mf METAFONT source files (Sections G.1 and G.3) contain drawing instructions for the characters in a font. The METAFONT program converts these to bitmap .pk files in a given size and resolution for DVI driver programs.


pk

Packed pixel, or bitmap, files (Section G.1) contain images of the characters in a font. These are read in by the DVI drivers to print the characters either on paper or on the computer monitor.

sty

This is a package file, to be loaded with \usepackage. A package contains additional \LaTeX commands to define new features or alter existing ones. It should not produce any printable text. It may be input by another package with the command \RequirePackage.

tex

The file containing the user's input text should have the extension .tex. For every \LaTeX document there is at least one .tex file. If there is only one .tex file, it is also the main file that determines the root name of all the others. If there are \input or \include commands within the input text, there are other files belonging to the document with different root names but all with the extension .tex. The main file is the one whose name appears when the program \LaTeX is called. It contains the highest level \input commands and is the only file that may have \include commands.

tfm

\TeX font metric files (Section G.1) contain all the font information that \TeX needs to process a document: height, width, depth of each letter, ligature information, italic correction, and so on. The only thing that is missing is what the character looks like. This is provided by .pk files.

toc

This file contains the information for the table of contents. The command \tableofcontents reads in the .toc file, if it exists, and then outputs the table of contents. At the same time, a new .toc file is opened where all the contents information for the current processing run is written. This .toc file is closed by the \end{document} command, so that if the \LaTeX run is interrupted before completion, the .toc file is lost.

A .toc file is only generated if the document contains the command \tableofcontents and if the \nofiles command has not been issued. It has the same root name as the main file.

vf

Virtual font files (Section G.1) are read in in place of .pk files for fonts that do not really exist independently. For each character, there are instructions for the driver program, usually to take character nn from real font xx.
Errors are bound to be made at times during the preparation of a long \LaTeX{} document. The mistakes can be of various kinds, from simple typing errors for command names to forgetting that some commands must be paired or giving an incorrect syntax for a complicated command.

Errors during the \LaTeX{} processing produce a list of messages on the monitor, which appear totally incomprehensible to the beginner. Even the advanced user can have difficulty figuring out a particular error message. However, these messages contain information about fundamental structures that can aid an experienced \TeX{} programmer to see much deeper into the heart of the problem.

In addition, the error messages contain useful information even for the beginner. It is the purpose of this chapter to explain some of these messages that can be of assistance to non-programmers.

Error messages are written both to the computer monitor and to the processing transcript file, with extension \texttt{.log}. Examine this file if the messages on the monitor went by too fast.

\section{Basic structure of error messages}

Error messages have two sources: those from \LaTeX{} and those from the underlying \TeX{} program. The \LaTeX{} messages are often followed by \TeX{} messages since \LaTeX{} operates at a higher level.

\subsection{\TeX{} error messages}

We start with a simple error as an example.

\begin{verbatim}
\documentclass{article}
\begin{document}
The last words appear in \textbf{bold face}.
\end{document}
\end{verbatim}
Appendix C. Error Messages

Here the command $\textbf{\texttt{\textbackslash{}textbf}}$ has been mistakenly typed as $\textbf{\texttt{\textbackslash{}textbf}}$. During the processing, $\LaTeX$ assumes that the user wants to invoke a $\TeX$ command $\textbf{\texttt{\textbackslash{}textbf}}$. Since $\LaTeX$ does not know the $\TeX$ commands, it passes this text on to $\TeX$, which then decides that there is no such command in its repertoire. The following error message is written to the monitor:

! Undefined control sequence.
1.3 The last words appear in $\textbf{\texttt{\textbackslash{}textbf}}$

{\texttt{\textbackslash{}textbf\{\textit{bold face}\}}}.

?

The program stops at this point and waits for a response from the user. This message can be understood even by beginners. It consists of an error indicator that starts with an exclamation point !. Here the indicator is:

! Undefined control sequence, meaning that an unknown command name (control sequence) was the cause of the error. Next comes a pair of text lines, the first of which is prefixed with 1.3, meaning that the error occurred in 'line 3' of the input text. The error itself was encountered at the last symbol printed in this upper line. The lower line shows the continuation of the input line being processed when the error was found, here the words \{\textit{bold face}\}. Before continuing, $\TeX$ waits for a reaction from the user, as indicated by the question mark in the last line of the message.

Entering another ? and \langle return \rangle as response produces the following information:

Type \langle return \rangle to proceed, S to scroll future error messages, R to run without stopping, Q to run quietly, I to insert something, E to edit your file, 1 or ... or 9 to ignore next 1 to 9 tokens of input, H for help, X to quit

?

This is a list of the possible user responses:

1. \langle return \rangle: simply typing the return key tells $\TeX$ to continue processing after making an attempt to handle the error according to some preprogrammed rules. In the case of an unknown command name, the error treatment is to ignore it.

2. S scroll mode: $\TeX$ continues the processing, writing further error messages to the monitor as they are encountered, but without stopping for a user response. This is as though \langle return \rangle were pressed after all subsequent errors.

3. R run mode: $\TeX$ continues processing as with S, but does not even stop as it would in scroll mode if the file named in an \texttt{\textbackslash{}input} or \texttt{\textbackslash{}include} command is missing.
4. **Q quiet mode:** the same as with R except that no further error messages are printed to the monitor. They are, however, written to the .log file.

5. **I insert:** the mistake can be corrected by inserting the proper text. `\TeX` puts the line of text entered from the keyboard into the processing in place of the error and then continues. Such a correction applies only to the processing; the original text in the .tex file is unchanged and must be altered with an editor. Typing in `I\stop` brings the program to a halt with the current page in the .dvi file.

6. **1 ...:** entering a number less than 100 will delete that many characters and commands from the subsequent text. The program then stops to await further response from the user.

7. **H help:** an extended account of the problem is printed to the monitor, which is more informative than the brief error indicator and which may contain tips for relieving the error.

8. **X exit:** The \TeX processing is halted at this point. The current page does not appear in the .dvi file.

9. **E edit:** The process is halted as with X and a message is printed saying on which line of which file the error occurred. For some implementations, the editor may actually be called automatically, going directly to the faulty line.

The above response letters may be typed in either in capitals or in lower case. The response does not take effect until after \langle return \rangle is pressed.

Typing H or h for help on the previous sample error message produces the text:

```
The control sequence at the end of the top line of your error message was never \def'ed. If you have misspelled it (e.g., `\hobx'), type `I' and the correct spelling (e.g., `I\hbox'). Otherwise just continue and I'll forget about whatever was undefined.
```

? The error is described in more detail: the command name at the end of the upper line is unknown; if this is simply a typing error, enter the proper text with the I response, in this case I\textbf. Otherwise, press \langle return \rangle and the faulty command will be ignored. The line of text will be processed as though it had been The last word appears in bold face. Of course, no bold face can actually appear.

The basic structure of a \TeX error message is summarized as:

*Every error message begins with an error indicator, marked by a ! at the beginning of the first line. The indicator text is a brief description of the*
Appendix C. Error Messages

Following are one or more lines of input text. The last symbol of the first of these lines is the one that has caused TeX to stop and print the error message. The last line contains the text or commands that are the next to be processed. TeX waits for a response from the user. If that response is the typing of an \texttt{H} for help, a more detailed description, with possible tips, is printed to the monitor, and TeX waits for a further response.

C.1.2 LaTeX error messages

As an example of a text with a LaTeX error, we take

\begin{verbatim}
\documentclass{article}
\begin{document}
  \begin{quote} \slshape
    Text indented at both ends
  \end{quote}
\end{document}
\end{verbatim}

Here the call \texttt{\begin{quote}} was incorrectly typed as \texttt{\texttt{qoute}}. The LaTeX processing writes the error message:

\begin{verbatim}
! LaTeX Error: Environment \texttt{qoute} undefined.
See the LaTeX manual or LaTeX Companion for explanation.
Type \texttt{H <return>} for immediate help.
...\end{verbatim}

The first line of this message states that LaTeX itself discovered this error, with a brief error indicator, in this case \texttt{Environment qoute undefined}. All LaTeX error messages begin with a line like this, followed by a reference to the LaTeX manuals for detailed explanation (which is also to be found in Section C.3 of this book). The third line of text is a reminder that the response \texttt{H \texttt{<return>}} will also type additional clarification.

The line of three dots \ldots{} indicates that there are more lines of internal coding that have been suppressed. Sometimes it is desirable to look deeper behind the error message, especially for practiced TeXperts who know how to interpret them. In this case, LaTeX can be made to output the missing lines of coding by setting

\begin{verbatim}
\setcounter{errorcontextlines}{num}
\end{verbatim}

where \texttt{num} is the number of levels a macro will be decoded on an error. By default, \texttt{num} = \texttt{-1}. The knowledgeable user can set it to \texttt{99} to get more lines than he or she will ever likely to want.
The next pair of lines shows just where the processing was halted when the error was detected. As for \TeX{} messages, the current input line of text is broken at the error, with the processed part on the first line, here \verb|\begin{qoute}|, and the remainder on the next line. Here this remainder is \verb|\slshape|. The line indicator \verb|l.3| (small letter \textit{L}, not number one) shows on which line (3) of the input file the error was found.

\LaTeX{} now awaits a response from the user. Typing \texttt{H} \texttt{(return)} yields the additional information:

\begin{quote}
Your command was ignored. \\
Type I \texttt{<command>} \texttt{(return)} to replace it with another command, or \texttt{<return>} to continue without it.
\end{quote}

This means the last command in the upper line of the pair beginning with \verb|l.3| has not been read into the processing. The mistake may thus be corrected \textit{for the current process run only} with \texttt{I\begin{quote}} \texttt{(return)}. The misspelling will still be in the text file, though, and needs to be removed by an editor run later.

However, if the processing is continued by simply pressing \texttt{(return)}, then \verb|\begin{qoute}| is ignored, as though this text were not in the source file. This leads directly to another error when the command \verb|\end{quote}| is encountered, since now the matching \verb|\begin{quote}| command is missing. The actual text of this second error message is:

\begin{quote}
! \LaTeX{} Error: \verb|\begin{document}| ended by \verb|\end{quote}.
\end{quote}

See the \LaTeX{} manual or \LaTeX{} Companion for explanation. \\
Type \texttt{H} \texttt{(return)} for immediate help.

\begin{quote}
\ldots
\end{quote}

\verb|1.5 \end{quote}|

\begin{quote}
\ldots
\end{quote}

The message again contains the standard \LaTeX{} announcement, with error indicator, in the first line, followed by a reference to the manuals, and an invitation to type \texttt{H} for help. The error indicator in this case is

\verb|\begin{document}| ended by \verb|\end{quote}|.

This arises because when \LaTeX{} encountered \verb|\end{quote}|, it checked the name of the current environment. Since \verb|\begin{quote}| was missing, the matching \verb|\begin| command is the \verb|\begin{document}| statement, producing a mismatched \verb|\begin|...\verb|\end| pair.

The last pair of lines again shows just how far the processing had occurred before the error was detected. In this case, the whole of the current line has already been taken in, so the remainder line is blank.
The response \texttt{H (return)} at this point leads to the same message as for the first error. A correction with an \texttt{I} entry will not do any good since the missing \texttt{\begin{quote}} can no longer be inserted ahead of the environment text. Entering \texttt{I\begin{quote}} will replace the \texttt{\end{quote}} with \texttt{\begin{quote}}, but that does not solve the real problem. The best response now is to type \texttt{\langle return\rangle} so that the command \texttt{\end{quote}} is ignored and the processing continues.

Now both the faulty \texttt{\begin{quote}} and correct \texttt{\end{quote}} commands are removed and the processing takes place as if the quote environment had never been in the input text at this point.

If the typing error \texttt{quote} had been made in the \texttt{\end} instead of in the \texttt{\begin} command, the error message would have been:

\begin{verbatim}
! LaTeX Error: \begin{quote} on input line 3 ended by \end{quote}
See the LaTeX manual or LaTeX Companion for explanation.
Type H <return> for immediate help.
...
1.5 \end{quote}
?
\end{verbatim}

The previous explanations should be enough for the user to understand what is now being said. The error indicator here is

\texttt{\begin{quote}} on input line 3 ended by \texttt{\end{quote}}

and the last pair of lines indicate that the problem is in line 5 and that the troublesome command is \texttt{\end{quote}}. The obvious thing to do now, and the \texttt{H} message encourages this conclusion, is to make a correction with \texttt{I\end{quote}}. However, now a new message appears on the monitor:

\begin{verbatim}
! Extra \endgroup.
<recently read> \endgroup
1.5 \end{quote}
?
\end{verbatim}

With the exception of the last pair beginning with 1.5, this makes no sense at all. The error indicator \texttt{! Extra \endgroup} is seemingly meaningless. That this is a \TeX error message and not a \LaTeX one is hardly any compensation.

The frustrated user is not really to blame. The response was perfectly reasonable even if it was wrong. Only experience could say that the best action at that point was to have just pressed \texttt{\langle return\rangle}. The quote environment is closed off anyway, although any special actions associated with \texttt{\end{quote}} would be left off.

The help message at this point is
Things are pretty mixed up, but I think the worst is over. This is at least encouraging, and the user should not lose heart. The best thing is to just keep on pressing \texttt{(return)} to get through the run.

We give here two recommendations for choosing a response, one specific and one general. The specific one is:

If a false environment name appears in a \texttt{\begin{command}} command, the proper correction of the error is

\begin{command}{right\_name}

If the misspelling has occurred in the \texttt{\end{command}} command, the best way to handle this mistake is just to press \texttt{(return)}. The environment will be closed and any local declarations or definitions will be terminated. However, if there are any commands to be executed or text to be printed by the \texttt{\end{command}} command, these will be missing.

The general recommendation is

If the user knows how to correct the error with the help of the error message, this should be done with

\begin{command}{correction}

Otherwise, one may give \texttt{(return)} and wait and see what happens. Even if more peculiar (\TeX) error messages appear, one can keep on pressing \texttt{(return)} until the processing is finally at an end. The following printout should indicate where the mistake was.

Instead of continually pressing \texttt{(return)}, one may also enter S, R, or Q and \texttt{(return)} to speed up the error treatment (Section 3.1.1). Here, as with the simple \texttt{(return)} entry, the faulty commands are not just ignored. \TeX attempts to treat them by making assumptions about what the user wanted to do at this spot. Only when this is no longer possible will \TeX ignore the command completely. For example, if the indicator reads

\texttt{\begin{environment} ended by \end{environment}}

there was at least a \texttt{\begin{command}} command with a valid environment name earlier. It could be assumed that the name of the environment in the \texttt{\end{command}} command is wrong. \LaTeX thus tries to execute this command using the name of the current environment.

\section{Error messages from \TeX macros}

The majority of \TeX commands and practically all the \LaTeX commands are so-called \TeX macros. These are combinations of primitive commands grouped together under a new command name, by which they may be
Appendix C. Error Messages

called as a unit. \TeX macros are structures similar to those that may be defined by the \LaTeX\ command \texttt{\textbackslash newcommand}. Up to nine arguments may be passed to them, just as for \LaTeX\ commands. However, the corresponding \TeX\ commands for generating macros are more general than \texttt{\textbackslash newcommand}.

In fact, of the approximately 900 Plain \TeX\ commands available, only 300 are primitives, or fundamental commands. The remaining 600 are macros. If an error occurs within a macro, the other commands within it may also be affected.

Here is an example for clarification. The command \texttt{\textbackslash centerline} is a macro defined as

\begin{verbatim}
\def\centerline#1{@@line{\hss#1\hss}}
\end{verbatim}

in which \texttt{@@line} is itself a macro while \texttt{\hss} is a \TeX\ primitive, a rubber length that can be infinitely stretched or shrunk. To avoid leading the user into the murky depths of \TeX\ commands, we will simply point out that the above macro definition is more or less equivalent to the \LaTeX\ command sequence

\begin{verbatim}
\newcommand{\centerline}[1]{\makebox[\textwidth][c]{#1}}
\end{verbatim}

Now take the following sample input text

\begin{verbatim}
\documentclass{article}
\begin{document}
\centerline{This is an \texttt{\textbackslash invalid} command}
\end{document}
\end{verbatim}

in which a \texttt{\textbackslash} has been placed before the word \texttt{invalid} making a false command named \texttt{\textbackslash invalid}. During the \LaTeX\ processing, the following \TeX\ error message appears:

\begin{verbatim}
! Undefined control sequence.
<argument> This is an \texttt{\textbackslash invalid}
command
1.3 \centerline{This is an \texttt{\textbackslash invalid} command}
? \end{verbatim}

This \TeX\ message is now a bit more understandable. The error indicator is the same as for the example in Section C.1.1:

\begin{verbatim}
! Undefined control sequence.
<argument> This is an \texttt{\textbackslash invalid}
\end{verbatim}

The following pair of lines claim that the error was recognized after the \texttt{command} \texttt{\textbackslash invalid} was processed, and that the next piece of text to be read in is the word \texttt{command}. At the same time, the \texttt{<argument>} at the beginning of the upper line says that this text is the argument of some other command.

The next pair of lines are familiar: the error occurred in line 3 of the input text, after the entire line, command and argument, had been read in for processing.
C.2 Some sample errors

C.2.1 Error propagation

The example with the incorrect \begin{quote} environment showed that a simple response with the \texttt{\textbackslash return} key led to a second error message in spite of the fact that the \texttt{\textbackslash end\{quote\}} command that produced it was correct. That one wrongly corrected error leads to further errors is more often the rule than the exception.

Let us look at the input text

\documentclass{article}
\begin{document}
\begin{itemie}
\item This is the first point in the list
\item And here comes the second
\end{itemize}
\end{document}

The only mistake in this text is the misspelled environment name \texttt{itemie} in place of \texttt{itemize}. The \LaTeX{} processing first produces the same error message as for the incorrect \texttt{quote} environment earlier:

! LaTeX Error: Environment itemie undefined.

See the \LaTeX{} manual or \LaTeX{} Companion for explanation.

Type \texttt{H \textbackslash return} for immediate help.

...$

1.3 \begin{itemie}$

? 

At this point, entering \texttt{(return)} leads to a new error messages:

! LaTeX Error: Lonely \texttt{\item}--perhaps a missing list environment.

See the \LaTeX{} manual or \LaTeX{} Companion for explanation.

Type \texttt{H \textbackslash return} for immediate help.

...$

1.4 \item

\begin{quote}$

his is the first point in the list

? 

What has happened here is that without the \texttt{\begin} command, the \texttt{\item} has been issued outside of a list environment, where it is rather meaningless. (Actually it is defined to print the above error message!) It is now too late to try to insert the missing start of the \texttt{itemize} environment. Typing \texttt{H \textbackslash return} for help yields
Try typing \textless\textbackslash return\textgreater{} to proceed.  
If that doesn't work, type \text{X} \textless\textbackslash return\textgreater{} to quit.

Following this advice and pressing \textless\textbackslash return\textgreater{}, one obtains the same error message once again, but this time on line 5, for the second \texttt{\textbackslash item} command. Persevering further and pressing \textless\textbackslash return\textgreater{} once again, one is presented with

\begin{verbatim}
! LaTeX Error: \begin{document} ended by \end{itemize}.

See the LaTeX manual or LaTeX Companion for explanation.  
Type \text{H} \textless\textbackslash return\textgreater{} for immediate help.  
...
\end{verbatim}

1.6 \texttt{\textbackslash end\{itemize\}}

The \texttt{itemize} environment has now come to its end, but since it was never properly started, \LaTeX{} complains about a mismatch of \texttt{\begin{document}} and \texttt{\end{document}} commands. Now at last, typing another \textless\textbackslash return\textgreater{} leads to a proper continuation of the processing. Of course the itemized text will be incorrectly formatted, but the rest of the document will not be affected.

In this example, a single mistake in the input text generated three further error messages. This is by no means unusual. Some \LaTeX{} errors can produce hundreds of such successive errors. It is even possible that the chain of errors never ceases and that the processing never advances. In this case, there is nothing else to do but to terminate the program. This should be done with the entry \texttt{I\textbackslash stop} after the next error message. It may be necessary to give it several times before it takes effect. If this does not work, that is, if the same message appears every time, then the response \texttt{X} \textless\textbackslash return\textgreater{} will halt the program immediately.

It is better to stop the program with \texttt{I\textbackslash stop} than with \texttt{X}, since then the output will include the last page being processed. This can be of assistance in trying to deduce what the source of the error was.

The final lesson of this section is simply: \textit{even when faced with a host of error messages, don't panic! Continue to press the \textless\textbackslash return\textgreater{} key to advance the processing.}

Pressing \texttt{S} \textless\textbackslash return\textgreater{} instead will produce the same set of error messages on the monitor but without waiting for a user response in between (Section C.1.1).

\section*{C.2.2 Typical fatal errors}

Occasionally it may happen that a user forgets to include one of the commands \texttt{\documentclass} or \texttt{\begin{document}}, or the entire preamble.
The last can easily occur when a \LaTeX\ file is to be read in with \texttt{\input} or \texttt{\include}, but is mistakenly entered directly with the \LaTeX\ program call. If a file with the text

This file has no preamble.

is put directly into the \LaTeX\ processing, the following message appears on the monitor:

! \LaTeX\ Error: Missing \texttt{\begin{document}}.

See the \LaTeX\ manual or \LaTeX\ Companion for explanation.
Type \texttt{H <return>} for immediate help.
...

1.1 T

his file has no preamble.
?

with the accompanying help message after typing \texttt{H <return>}: 

You're in trouble here. Try typing \texttt{<return>} to proceed.
If that doesn't work, type \texttt{X <return>} to quit.
?

One sees from the error message that \LaTeX\ has discovered a mistake on reading the very first letter of the text. The help message is also not very encouraging. There is no point trying to continue the processing with \texttt{<return>}. Rather one should stop it with \texttt{X} or \texttt{E} right away, since nothing useful can be achieved here.

Even when the above example file contains the environment

\begin{document}

This file has no preamble.
\end{document}

it is not possible to carry out a proper processing. The \TeX\ error message now printed is

! \LaTeX\ Error: The font size command \texttt{\normalsize} is not defined:
there is probably something wrong with the class file.

See the \LaTeX\ manual or \LaTeX\ Companion for explanation.
Type \texttt{H <return>} for immediate help.
...

1.1 \begin{document}
?

Appendix C. Error Messages

The error indicator is rather peculiar since the command \normalsize never appears in the input. Only the last two lines of this message are easily understood. They state that the error was recognized on line 1 after \begin{document} was read in. The fact that this is the text of the first line is enough to show that no meaningful processing can take place, since the mandatory \documentclass command that must precede it is missing.

Here again there is no choice but to halt the program with X or E, since a continuation of the processing would be meaningless.

(The reason for the strange error indicator is that many formatting parameters are initiated by the \begin{document} statement, including the standard font. It is here that \normalfont is called internally, and that command must be defined in the class file, which is missing. Other initiating commands are defined in the \LaTeX format itself, and so they do not cause this error.)

If the name of the document class is wrongly typed, say as

\documentclass{fred}

the following message appears:

! LaTeX Error: File ‘fred.cls’ not found.

Type X to quit or <RETURN> to proceed, or enter new name. (Default extension: cls)

Enter file name:

This message should be fairly clear: the program is looking for a file with the name \texttt{fred.cls} but cannot find it, so would the user be so kind as to enter an alternative name. If the new name has the same extension as that requested (.cls), it need not be explicitly given. In this case, the name of any standard \LaTeX class, such as \texttt{article}, \texttt{report}, \texttt{book}, or \texttt{letter}, may be typed in, as well as that of any additional classes that you might have access to. Of course, it should be the class for which the document was written.

This same error message is printed whenever a file is to be read in that cannot be found on the system. The issuing command may be \texttt{\input}, \texttt{\include}, or \texttt{\usepackage}. If the file sought really does exist, perhaps it is not located where \TeX is looking for files. Usually there is a system parameter set by the installation to point \TeX to its files. If your file is somewhere else, you can trying entering its full name, with directory or path designations.

If \LaTeX is asking for a file that you know does not exist, or which is totally unknown to you, you will want to halt the processing right away or tell \LaTeX to skip the silly file. At this point, you have a problem. \LaTeX insists on a valid file name. Typing X \texttt{(return)} only produces the
message once again, this time stating that it cannot find the file x.ext. Some installations provide a dummy file named null.tex, while one of the extension packages available (Section B.5.4) offers a set of files named after the standard response letters x.tex, e.tex, r.tex, h.tex, s.tex to emulate the responses to normal error messages.

**Emergency stop:** sometimes one reaches a situation where the program cannot be halted after an error, even with I\stop or X. In such a case, one has to apply the operating system's program interrupt, usually CTRL Z.

### C.2.3 Mathematical errors

Surprisingly few errors occur as a result of incorrect application of mathematical formula commands themselves, even for a user with only a little experience. More often, the errors in mathematical formulas are fiddly ones such as forgetting a closing brace } or neglecting to switch back to text mode. Another type of common error is to use a symbol in text mode that may only appear in math mode. We will point out a few typical examples here.

One wants to produce the text: ‘The price is $3.50 and the order number is type_sample’, and types the input text:

```
The price is $3.50 and the order number is type_sample.
```

This text contains two errors, such that the first cancels the second one: the $ sign is the math mode switch for generating formulas within text (Section 5.1). The proper way to write a true dollar sign in text is as \$. Instead, in the sample text, the $ alone switches to math mode, setting everything that follows as a formula. However, the closing switch-back $ is missing, something that \TeX will first notice at the end of the current paragraph.

```
! Missing $ inserted.
<inserted text>
   $
1.4
```

If \langle return \rangle is simply pressed as a response, \TeX inserts a $ at this point. For our sample text, this would be at the end of the text before the blank line. This means that the text from the first $ to the end of the paragraph will be set as a text formula:

```
The price is 3.50 and the order number is type_sample.
```

This output should show the user right away what went wrong. Near the end of the line, there is a letter s printed as an index. This is the second error in the example. The underbar character _ is only permitted
Appendix C. Error Messages

in math mode and should have been typed here as \_. However, after the first $ had switched (incorrectly) into math mode, the _ sign became allowed, with the result that the following letter s was lowered.

If we now correct the text by replacing the $ sign with \$, then the _ symbol is no longer in math mode, producing the error message:

! Missing $ inserted.
<inserted text>
$ 1.5 ...ice is \$3.50 and the order number is type__

sample.

Pressing (return) here tells T\TeX to recover from the error by inserting the apparently missing $ sign at this point before the math command _. The processing continues, and at some place before the end of the current paragraph, T\TeX will notice again that the closing $ is not present. The same error message is printed as in the first case. The processing goes on with another (return) response, with the result:

The price is $3.50 and the order number is type_ample.

In all three cases, asking for more help with H would type to the monitor:

I've inserted a begin-math/end-math symbol since I think you left one out. Proceed, with fingers crossed.

The last sentence above is just what we recommend for mathematical error messages: keep on pressing (return) or S to get through to the end of the processing, and then look at the printed output to find the mistake.

C.2.4 Errors from multi-file texts

If the document text is split over many files that are to be read in with \input or \include commands, the line number in the error message refers to the file currently being read. A response with E (return) will call the editor program and open that file, going to the indicated line where the error was recognized. With the other responses, the editor has to be called separately, with the faulty file explicitly named.

It is possible to determine which file was being processed at the time of the error by examining the processing messages or the .log file. As the files are opened for reading, T\TeX writes an opening parenthesis ( and the name of the file to the monitor and to the transcript .log file. When the file is closed, a closing parenthesis ) is printed. Output page numbers are similarly written to both the monitor and transcript file in square brackets. For example, if the monitor shows the processing messages
C.3 List of \LaTeX error messages

1.999 \finish
?

this can be interpreted as follows: a file myfile.tex was being read, and after pages 1, 2, and 3 had been output, the file part1.tex was read in with an \input or \include command inside the file myfile.tex. Pages 4 and 5 were output and part1.tex was closed, after which another file part2.tex was opened for input. An error was discovered in line 999 of this file. If this error is corrected from the keyboard, or if the processing is otherwise continued, the monitor messages proceed as

[8] [9]) [10]
! Too many }'s
1.217 \em sample}

The closing parenthesis after page 9 indicates that the file part2.tex has been closed. The next error is on page 10 in the main file myfile.tex, since there is no closing ) for it. The error was discovered on line 217 of this file.

C.3 List of \LaTeX error messages

The \LaTeX error messages are listed here, divided into general, package, or font error messages. Within each group, they are ordered alphabetically according to the error indicators. A description of the possible causes and solutions is also given.

C.3.1 General \LaTeX error messages

The following error messages are those not involving font selection or definition, nor any of the special features for \LaTeX programming class and package files (Appendix D).

! \LaTeX Error: \ldots undefined.

The argument of a \renewcommand or \renewenvironment has not been previously defined. The corresponding \new\ldots command should be used instead.

! \LaTeX Error: \textless in mid line.

The command \textless in a tabbing environment occurred in the middle of a line. This command may only appear at the beginning of a line (Section 4.6.3).
Appendix C. Error Messages

! LaTeX Error: Bad \line or \vector argument.
The first argument of a \line or \vector command specifies the angle of the line or arrow. This message states that the selected angle entries are invalid. See pages 293–294.

! LaTeX Error: Bad math environment delimiter.
\LaTeX has encountered a math switch command in the wrong mode: either a \[ or \( command in math mode or the corresponding \] or \) in normal text mode. Either the math switches have been improperly paired or some braces {...} are incorrect.

! LaTeX Error: \begin{...} on input line ... ended by \end{...}.
\LaTeX has encountered an \end command without a corresponding \begin of the same name. This may be due to a typing mistake in the name of the environment, or to the omission of a previous \end command. A good way to avoid this error is always to enter the \end command immediately after the \begin, inserting the actual environment text with the editor in between the two. This is especially useful for long, nested environments. It also reduces the risk of typing the environment name incorrectly in the \end command.

! LaTeX Error: Can be used only in preamble.
Many commands may only be called within the preamble. These include \documentclass, \usepackage, \nofiles, \includeonly, \makeindex, \makeglossary, and several others. Certain commands that only have meaning within class or package files, such as \ProvidesClass and \ProvidesPackage (Section D.2.1), as well as many \Declare.. and \Set.. commands are also only allowed in the preamble. If one of these commands is issued after \begin{document}, this message is printed.

! LaTeX Error: Command ... invalid in math mode.
A command has been issued in math mode that only makes sense in text mode, such as \item or \circle. The font declarations \itshape, \bfseries, and so on, also produce this error in math mode, since the math alphabet commands \mathit, \mathbf should be used instead.

! LaTeX Error: Command ... already defined.
The user has tried to redefine an existing structure with one of \newenvironment, \newcommand, \newtheorem, \newsavebox, \newfont, \newlength, \newcounter, or \DeclareMathAlphabet. Either a different name must be selected or, in the case of commands and environments, the \renew... version must be employed. (Note that when an environment named sample is defined, the commands \sample and \endsample are also created.)
! LaTeX Error: Command ... undefined in encoding ....
The specified command has been defined with \DeclareTextCommand
for a certain NFSS encoding (say OT1) but was executed while another
encoding (for example T1) for which there is no definition was active.

! LaTeX Error: Counter too large.
A counter that is to be printed as a letter contains a value greater than 26.

! LaTeX Error: Environment ... undefined.
\LaTeX{} has encountered a \begin{command} with an unknown environment
name. This is probably due to a typing mistake. It may be corrected
during the processing with the response I followed by the correct name.
(This does not alter the source file, where the error remains.)

! LaTeX Error: File ‘...’ not found
Type X to quit or <RETURN> to proceed,
or enter new name. (Default extension: ...)
Enter file name:. 

A file is to be loaded with one of the inputting commands, but it cannot
be found. You may type a new file name, or quit, or proceed without it.
If the file name has been given correctly, and it does exist, maybe it is
not located where \LaTeX{} looks for files. Make sure that it is in one of the
right directories. The user is offered a chance to type in an alternative,
or correctly typed, name via the keyboard. Type in the file name, with
optional extension, and press \langle return \rangle. \LaTeX{} proposes a default extension
that is the same as that requested. There is no default for the main part
of the file name.

! LaTeX Error: Float(s) lost.
A figure or table environment or a \marginpar command was given
within a vertical box (\parbox or \minipage environment), or these struc-
tures were issued within a \LaTeX{} command that uses vertical boxes inter-
nally, such as a footnote. This error is first recognized by \LaTeX{} when a
page is output so that the actual cause could be many lines earlier in the
text. A number of tables, figures, or marginal notes may have been lost
as a result, but certainly not the one that triggered the problem.

! LaTeX Error: Illegal character in array arg.
A tabular or array environment contains an unknown column formatting
character (see Section 4.8.1) or the formatting entry in the second
argument of a \multicolumn command is wrong.

! LaTeX Error: \include cannot be nested.
An attempt has been made to call \include from a file that has already
been read in by an \include command. All \include commands must be issued in the main file (Section 9.1.2).

! LaTeX Error: LaTeX2e command ... in LaTeX 2.09 document.
This error occurs if a \LaTeXe command is used in compatibility mode (Appendix F), that is, if the document uses \documentstyle instead of \documentclass. These can be \LaTeXe, \usepackage, \ensuremath, the \verb|\rbox| environment, as well as the new syntax for adding an optional argument with \newcommand and \newenvironment.

! LaTeX Error: Lonely \item--perhaps a missing list environment.
An \item command has been given outside of a list environment (Section 4.3). Either the environment name has been misspelled in the \begin command and not corrected by keyboard entry, or the \begin command has been forgotten.

! LaTeX Error: Missing @-exp in array arg.
The column formatting argument of a \tabular or \array environment contains an @ symbol without the necessary following text in curly braces { } that must go with it (Section 4.8.1 for @-expressions), or the same thing occurs in the second argument of a \multicolumn command.

! LaTeX Error: Missing begin{document}.
Either the \begin{document} command has been forgotten or there is printable text inside the preamble of the document. In the latter case, there could be a declaration with incorrect syntax, such as a command argument without curly braces { } or a command name without the backslash \ character.

! LaTeX Error: Missing p-arg in array arg.
The column formatting argument of a \tabular or \array environment contains a p symbol without the necessary width specification that must go with it (Section 4.8.1), or the same thing occurs in the second argument of a \multicolumn command.

! LaTeX Error: No counter ‘...’ defined.
A \setcounter or \addtocounter command was called that referred to a counter name that does not exist. Most likely the name was typed incorrectly. If this error occurs while an .aux file is being read and if the name of the counter is indeed correct, the defining \newcounter command was probably given outside of the preamble. Therefore it is highly recommended always to give the \newcounter commands inside the preamble. (If the other \LaTeX counter commands are used with an undefined counter name, a long list of funny \TeX error messages appears.)
! LaTeX Error: No \title given.
The \maketitle command has been given before \title has been declared.

! LaTeX Error: Not in outer par mode.
A figure or table environment or a \marginpar command was given within math mode or inside a vertical box (\parbox or \minipage environment). In the first case, the math switch-back command was probably forgotten.

! LaTeX Error: Page height already too large.
The command \enlargethispage is trying to extend the vertical size of the page, which \LaTeX already considers too large.

! LaTeX Error: \pushtabs and \poptabs don't match.
The number of \poptabs commands in a tabbing environment does not agree with the number of previous \pushtabs given (Section 4.6.4).

! LaTeX Error: Something's wrong--perhaps a missing \item.
The most likely cause of this error is that the text in a list environment (list, itemize, enumerate, or description) does not begin with an \item command. It will also occur if the bibliography environment is given without the argument \{sample_label\} (Section 9.3).

! LaTeX Error: Suggested extra height (...) dangerously large.
The command \enlargethispage is trying to extend the vertical size of the page more than \LaTeX considers reasonable.

! LaTeX Error: Tab overflow.
The last \= command in a tabbing environment exceeded the maximum number of tab stops allowed by \LaTeX.

! LaTeX Error: There's no line here to end.
The command \newline or \\ has been issued after a \par or blank line where it makes no sense. If additional vertical space is to be inserted here, this should be done with a \vspace command.

! LaTeX Error: This may be a \LaTeX bug.
This message states that \LaTeX is fully confused. This could be the result of a previous error after which the user response was to press \langle return \rangle. In this case, the processing should be brought to a halt with I\stop or X or E and the earlier error corrected. It is also possible, although unlikely, that there is a bug in the \LaTeX program itself. If this is the first error message
in the processing, and the text seems otherwise to be satisfactory, the
file should be saved and submitted to the computing center for further
investigation.

! LaTeX Error: Too deeply nested.
Too many list environments (description, itemize, enumerate, or list)
have been nested inside one another. The maximum depth of such nesting
is dependent on the installation but should always be at least four.

! LaTeX Error: Too many columns in eqnarray environment.
The eqnarray environment may only have three columns per line. You
may have forgotten to start a new row with \\, or have placed an extra &
in the line.

! LaTeX Error: Too many unprocessed floats.
This error may occur if there are too many \marginpar commands on one
page. However, it is more likely that \LaTeX{} is retaining more figure and
table floats than it can hold. This happens if too many such float objects
have been given before they can be output (Chapter 7). In this case, the
last figure or table should be added later in the text. Another cause could
be that the figure or table cannot be located on a normal text page, but
rather on a special float page at the end of the text or after a \clearpage
or \cleardoublepage command. Since the output sequence of figures
and tables will be the same as that with which they were input, one such
float can block the entire queue. A \clearpage or \cleardoublepage
command can free the blockage.

! LaTeX Error: Undefined tab position.
A \>, \+, \-, or \< command in a tabbing environment has tried to move
to a tabulator stop that does not exist (Section 4.6).

! LaTeX Error: \verb ended by end of line.
The text of an in-line verbatim command \verb+...+ extends over more
than one line of text. This is forbidden in order to catch a common error:
the missing terminating character. Make sure that the entire text between
the initial and terminating characters is all on one input line.

! LaTeX Error: \verb illegal in command argument.
The \verb command may not be used in the argument of any other
command, except \index and \glossary. It may not be used within
section titles or footnotes, for example.
C.3.2 \LaTeX{} package errors

The special programming commands for handling class and package files (Appendix D) have their own set of error messages. If one of these features issues a serious error, there is often little the user can do about it other than report the problem to the author of the file. On the other hand, some errors are due to the improper use of the file or of the options that it provides.

Often classes and packages contain their own error or warning messages with text and meanings that are peculiar to them. These are indicated as such, along with the name of the class or package that issued them. For example, the package \texttt{mypack} could print the error

\begin{verbatim}
Package mypack Error: cannot mix options 'good'
(mypack) and 'bad'.
\end{verbatim}

A help message should also be available when \texttt{H} is typed. Obviously, such error (and warning) messages cannot be explained here, since they depend entirely on the package in question.

! \LaTeX{} Error: \texttt{\textbackslash LoadClass} in package file.

A package file has called \texttt{\textbackslash LoadClass}, which it is not allowed to do. A class file can only be loaded from another class file.

! \LaTeX{} Error: Option clash for package ....

The specified package has been requested a second time with a different set of options. A package file will only be loaded once and a second attempt will be ignored. Thus, if two \texttt{\usepackage} or \texttt{\RequirePackage} commands load the same package with different options, there is a conflict. Try to arrange for a consistent set. Typing \texttt{H} for help after this message will print out the two sets of options.

! \LaTeX{} Error: \texttt{\textbackslash RequirePackage} or \texttt{\textbackslash LoadClass} in Options Section.

These two commands may not appear in the definition of a class or package option made with the \texttt{\DeclareOption} command. Instead, the option should set some flag or other indicator that is later tested before calling the command in question.

! \LaTeX{} Error: This file needs format ‘...’ but this is ‘...’.

The \texttt{\NeedsTeXFormat} command specifies a different format from the one being used. The \textit{format} is the prestored set of instructions that determine what type of \TeX{} is being run. For \LaTeX{} 2\epsilon, the format name specified must be \LaTeX{}2\epsilon. This file cannot be processed at all with the current format in use.
Appendix C. Error Messages

! LaTeX Error: Two \documentclass or \documentstyle commands.
A document may contain only one \documentclass or \documentstyle command. If you can only see one in the main document file, check that other files being loaded do not contain an offending second one.

! LaTeX Error: Two \LoadClass commands.
The class file contains more than one \LoadClass command, something that is not allowed. The class file has been improperly written.

! LaTeX Error: Unknown option ‘...’ for package ‘...’.
An option has been specified with \usepackage but it is not defined for that package. Check the instructions for the package or look for a misprint.

! LaTeX Error: \usepackage before \documentclass.
The \usepackage command may not appear before \documentclass. A class file must be loaded before any packages.

C.3.3 \LaTeX font errors

The following error messages occur when defining or selecting fonts with the New Font Selection Scheme (Appendix A). Some of these messages indicate that the fonts have not been properly set up, in which case the font description files may be corrupted or contain mistakes. In either case, the system manager must see that a proper set of files are installed.

! LaTeX Error: ... allowed only in math mode.
A math alphabet command such as \mathbf has been used in text mode. Perhaps a $ has been forgotten.

! LaTeX Error: Command ‘...’ not defined as a math alphabet.
The name of a non-existent math alphabet has been specified in one of the \Set.. or \Declare.. commands that take a math alphabet name as argument. The math alphabet name must be declared with the command \DeclareMathVersion before it may be used.

! LaTeX Error: Command ... not provided in base \LaTeX2e.
A number of symbols that were part of the basic \LaTeX 2.09, but not in \TeX, are no longer automatically included in \LaTeX2e. Add one of the packages latexsym or amsfonts to include them.

! LaTeX Error: \DeclareTextComposite used on inappropriate command ....
\DeclareTextComposite (Section A.3.7) is used to redefine an existing
accenting command to print a single character for a certain encoding. If the accenting command does not already exist, this message is issued.

! \LaTeX{} Error: Encoding scheme ‘...’ unknown.

A font declaration or selection command that refers to a non-existent encoding scheme has been issued. There is probably a typing error involved here.

! \LaTeX{} Error: Font family ‘...+...’ unknown.

A \texttt{\DeclareFontShape} command has been issued for a font encoding and family combination that has not been previously declared by means of a \texttt{\DeclareFontFamily} command (Section A.3.6).

! \LaTeX{} Error: Font ... not found.

The font with the specified attributes could not be found, nor could an adequate substitution be made. The font defined by \texttt{\DeclareErrorFont} is used instead.

! \LaTeX{} Error: Math alphabet identifier ... is undefined in math version ‘...’.

A math alphabet (Section A.3.3) that has not been defined for the current math version has been invoked. This means that the alphabet has been created by \texttt{\DeclareMathAlphabet} with an empty shape attribute, but no \texttt{\SetMathAlphabet} declaration has been given to define it for the selected math version.

! \LaTeX{} Error: Math version ‘...’ is not defined.

The name of a non-existent math version has been specified in one of the \texttt{\Set...} or \texttt{\Declare...} commands that take a math version name as argument. The version name must be declared with \texttt{\DeclareMathVersion} before it may be used.

! \LaTeX{} Error: Not a command name: ‘...’.

The first argument of \texttt{\DeclareMathAccent} must be a command name, like \texttt{\acute}. Most likely the backslash \ character has been forgotten.

! \LaTeX{} Error: Symbol font ‘...’ not defined.

The name of a non-existent symbol font has been specified in one of the \texttt{\Set...} or \texttt{\Declare...} commands that take a symbol font name as argument. The font name must be declared with \texttt{\DeclareSymbolFont} before it may be used as a symbol font.
Appendix C. Error Messages

! LaTeX Error: The font size command \normalsize is not defined: there is probably something wrong with class file.

It is necessary for class files to define \normalsize, the basic size of standard text in the document. If the class file does not do this, if it only defines \@normalsize, then it must be repaired. This message also appears if \documentclass is missing, for an empty class file is clearly a faulty one.

! LaTeX Error: This NFSS system isn't set up properly.

Something is wrong with the font descriptions in the .fd files, or there is no valid font declared by \DeclareErrorFont. This is a serious problem, to be reported to the system manager.

! LaTeX Error: Too many math alphabets used in version ....

There is a limit of 16 math alphabets possible, a number set by T\TeX itself. Any additional math alphabet definitions are ignored.

! LaTeX Error: Unknown symbol font ‘...’.

The name of a non-existent symbol font has been used as argument to the command \DeclareSymbolFontAlphabet. The font name must first be declared with \ DeclareSymbolFont before it may be used.

C.4 \TeX error messages

This section contains a list of some of the most common \TeX error messages, ordered alphabetically according to the error indicator, together with a brief description of the possible cause. Each one begins with an exclamation point only.

! Counter too large.

As a \TeX error, this message refers to a footnote marked either with letters or symbols in which the counter has exceeded a value of 26 or 9. It may also occur if there are too many \thanks commands on a title page.

! Double subscript.

A mathematical formula contains two subscripts for the same variable, for example, x_{2_3} or x_{\{2\}_{3}}. The proper way to produce $x_{2_{3}}$ is with x_{2_3} or x_{2_{3}} (Section 5.2.2).

! Double superscript.

A mathematical formula contains two superscripts for the same variable, for example, $x^{2^3}$ or $x^{\{2\}^{3}}$. The proper way to produce $x^{2_{3}}$ is with x^{2_{3}} or x^{2_{3}} (Section 5.2.2).
C.4. \TeX error messages

! Extra alignment tab has been changed to \cr.
A line in a tabular or array environment contains more & commands than there are columns defined. The error is probably due to a forgotten \at the end of the previous line.

! Extra }, or forgotten $.
In math mode, either an opening brace { has been left off or an extra closing brace } has been included by mistake. Another possibility is that a math mode switch command such as $, \[, or \( has been forgotten.

! Font ... not loaded: Not enough room left.
The text processing requires more character fonts to be loaded than \TeX can handle owing to memory limitations. If certain parts of the document need different fonts, you can try to split it up and to process the parts separately.

! Illegal parameter number in definition of ....
This error message is probably due to a \newcommand, \newenvironment, \renewcommand, or \renewenvironment command in which the substitution character # has been applied incorrectly. This character may appear within the defining text only in the form #n, where n is a number between 1 and the number of arguments specified in the command. Otherwise the character # may only appear in the definition as \#. This error may also arise if the substitution character is applied within the last argument \end{def} (Section 8.4).

! Illegal unit of measure (pt inserted).
If this error message comes right after another error with the message

! Missing number, treated as zero.
the problem lies with this previous error (see below). Otherwise, the mistake is that \TeX expects a length specification at this point but has only been given a number without a length unit. This occurs most often when a length is to be set to zero and 0 has been typed in instead of 0mm or 0pt. If this is the case, then responding with \return produces the right result since for a zero value any unit specification is all right. This error may also occur if a length specification has been completely forgotten.

! Misplaced alignment tab character &.
The single character command & has been given in normal text outside of the tabular and array environments. Possibly the intention was to print \&, in which case \& should be typed. This may still be achieved during the processing by responding with \& to the error message.
Appendix C. Error Messages

! Missing control sequence inserted.
This error is most likely caused by a \newcommand, \renewcommand,
\newlength, or \newsavebox command in which the backslash \ is missing
from the first argument. Pressing \textit{return} as response will complete
the processing correctly since \TeX assumes a backslash is missing and
inserts it.

! Missing number, treated as zero.
This error is most likely due to a \LaTeX command that expects a number or
length as argument, which is missing. It may also occur when a command
that takes an optional argument is followed by text beginning with \[. Finally another cause can be a \protect command preceding a length or
\texttt{value} command.

! Missing \{ inserted.
! Missing \} inserted.
\TeX is totally confused when either of these error indicators is written.
The line number printed will probably not be where the source of the
error is to be found, which is a missing opening or closing brace. If the
error is not obvious, continue the processing by pressing \textit{return} and try
to deduce where it might be from the printed output.

! Missing $ inserted.
Most likely a symbol or command that may only appear in math mode
was used in normal text. Recall that all those commands described in
Chapter 5 are only allowed in math mode unless otherwise stated. If
an \texttt{mbox} command is inserted within a math formula, its argument has
temporarily exited from math to normal text mode. The message may
also occur if a blank line appears within a math formula, signalling a new
paragraph and the end of the formula without the necessary $ sign.

! Not a letter.
The word list in a \texttt{hyphenation} command contains a character that is
not recognized as a letter, such as an accent command like \'{e}. Such
words can only be hyphenated by explicitly inserting the hyphenation
possibilities (Sections 2.8.1 and 2.8.2).

! Paragraph ended before ... was complete.
The argument of a command contains a blank line or a \texttt{par} command,
something that is not allowed. Probably a closing brace } has been
omitted.

! \TeX capacity exceeded, sorry [...]\
\TeX sets up various storage buffers in the computer memory to carry out
C.4. \TeX error messages

The text processing. This message appears when one of these buffers is full and can no longer be used. The name of the buffer and its maximum size are printed in the square brackets of the error indicator. With this message, the \TeX processing is terminated. The source of this problem is hardly ever due to insufficient memory, no matter how much complicated text is being processed, but rather to an error in the text itself. The methods described in Section C.6 may be applied to try to detect the true error.

The following descriptions of the various buffers should help to decide whether the storage capacity allotted to \TeX really is too small and to explain what one might do to correct this.

**buffer size** The problem here can be that the text in the argument of a sectioning, `\caption`, `\addcontentsline`, or `\addtocontents` command is too long. The message then normally appears when the `\end{document}` command is processed, but may also arise at one of `\tableofcontents`, `\listoffigures`, or `\listoftables`. The way to avoid this is to use the optional argument for the short form of the heading text (Sections 3.3.3 and 7.4). Indeed, such a long entry in the table of contents is a nuisance anyway and should be shortened. After the correction has been made in the input text, the previous \TeX.aux file must be deleted before reprocessing.

This problem can occur on a PC if a word processing program has been used to generate the input text instead of a text editor. Some of these programs put an entire paragraph into a single line even though the text on the monitor is broken up into lines.

**exception dictionary** The list of hyphenation exceptions that have been entered with the `\hyphenation` commands has become too large. Words used less frequently should be removed and their possible word divisions indicated explicitly with the `\-` command.

**hash size** The source file contains too many command definitions or uses too many cross-reference markers. This does not mean that the input text really needs all these commands, for it may be that the user has developed a large collection of private commands that are stored in a single file and read into every document, whether they are all applicable or not.

**input stack size** An overflow of this buffer is probably due to a mistake in a command definition. For example, the command defined with

\begin{verbatim}
\newcommand{\com}{One more \com}
\end{verbatim}

produces `One more {One more {...One more \com}...}` going on for ever, since it continually calls itself. Actually, it does not go on for ever, but only until this buffer is full.

**main memory size** This buffer contains the text for the page currently being processed. It also overflows if a recursively defined command has been called. However, the more usual reasons are: (1) a large number of very complicated commands have been defined on one page; (2) there are too
many \index or \glossary commands on one page; (3) the page itself is too complex to fit within the allotted buffer space.

The solution to the first two situations is clear: reduce the number of command definitions and/or \index and \glossary commands on that page. In the third case, the cause might be a long \texttt{tabbing}, \texttt{tabular}, \texttt{array}, or \texttt{picture} environment, or a stuck float (figure or table) waiting for an output command.

To find out whether the memory overflow is really due to an overly complex page, add the command \texttt{\clearpage} just before the spot where the overflow occurs. If the error message no longer appears on the next processing run, then this page was indeed too complicated for \TeX. However, if the overflow still persists, the error is probably a mistake in the input text. If necessary, it may have to be located with the method explained in Section C.6.

If the page really is too complex for the \TeX processing, it must be simplified. However, first recall that the entire last paragraph is processed before that page is output, even if the page break occurs near the beginning of the paragraph. Introducing a \texttt{\newpage} may solve the problem and should be tried out before attempting a tedious restructuring of the text. If the error is due to a stuck figure or table, it might be alleviated by moving the float object to later in the text, or by changing the positioning argument (Section 7.1). If the whole text is not yet complete, one can try giving \texttt{\clearpage} for now in order to clear the blocked floats and then decide on a better ordering when the text is finalized.

**pool size** Most likely there are too many command definitions and/or labels, or their names are too long. Try shortening the names. This error may also occur if a closing right brace \texttt{)} has been forgotten in the argument of a counter command such as \texttt{\setcounter} or in a \texttt{\newenvironment} or \texttt{\newtheorem} command.

**save size** This buffer overflows when commands, environments, and the scope of declarations are nested too deeply. For example, if the argument of a \texttt{\multiput} command contains a \texttt{picture} environment, which in turn possesses a \texttt{\footnotesize} declaration with another \texttt{\multiput} command, and so on. Such a nesting must be simplified, unless the real problem is a forgotten closing brace \texttt{)} that merely makes the structure appear so complex.

**! Text line contains an invalid character.**

The input text contains a strange symbol that \TeX does not recognize. This could be a problem with the editor itself, that it is inserting extra characters. If an examination of the source file does not reveal the strange symbol, consult the computing center for help.

**! Undefined control sequence.**

Every \TeX user encounters this error message at some point. It is usually
the result of an incorrectly typed command name. It may be amended during the processing by responding with `I` and the proper name of the command, plus `<return>`. This does not alter the source file, which must be corrected separately after the \LaTeX run. If the command name has been entered correctly in a \LaTeX command, it may be that it was issued in an improper environment where it is not allowed (that is, not defined there).

! Use of ... doesn’t match its definition.

If ‘...’ is the name of a \LaTeX command, it is likely that one of the picture commands from Sections 13.1.3 and 13.1.4 has been called with the wrong syntax for its arguments. If the name is `\@array`, there is a faulty @-expression (Section 4.8.1) in a `tabular` or `array` environment. Possibly a fragile command was given in the @-expression without the `\protect` command preceding it.

! You can’t use ‘macro parameter #’ in ... mode.

The special symbol `#` has been used in normal text. Probably there should have been a `\#` in order to print # itself. This can be corrected during the processing with the response `I\#` and `<return>`.

### C.5 Warnings

\TeX and \LaTeX errors both bring the processing run to a temporary stop and wait for a reaction from the user, or they may halt the program completely. Warnings, on the other hand, merely inform the user that the processed output may contain some faults that he or she might want to correct. Warnings appear on the monitor along with the page number where they occur, without the program coming to a stop. They are also written to the `.log` file where they may be examined after the \LaTeX processing and possible printing. Warnings may be issued either by \LaTeX or by \TeX itself.

#### C.5.1 General \LaTeX warnings

\LaTeX warnings are indicated by the words ‘\LaTeX Warning:’ at their start, followed by the warning message itself.

\LaTeX Warning: Citation ‘...’ on page ... undefined on input line ....

The key in a `\cite` or other citation command has not yet been defined with a corresponding `\bibitem` command (Section 9.3). If this message does not disappear on the next run, then the bibliography is missing
that defining `\bibitem` entry, either because \LaTeX{} has not been run, or because that reference is not in any of the specified databases.

\LaTeX{} Warning: Command `...' has changed.
Check if current package is valid.
The `\CheckCommand` statement is used to test if a given command has a certain definition. This warning is issued if the test fails. This is used to be sure that a given package is doing what the programmer thinks it is doing.

\LaTeX{} Warning: Float too large for page by ..pt on input line...
A float (figure or table environment) is too big to fit on the page. It will be printed anyway, but will extend beyond the normal page margins.

\LaTeX{} Warning: ‘h’ float specifier changed to ‘ht’.
The float placement specifier ‘h’ for ‘here’ (Section 7.1) does not really mean exactly at the location of the float environment. It will be placed ‘here’ only if there is enough room remaining on the page for it, otherwise it appears at the top of the next available page. This message is a reminder of this fact.

\LaTeX{} Warning: inputting ‘...’ instead of obsolete ‘...’.
Some packages written for \LaTeX{} 2.09 explicitly input certain files, such as `article.sty`, that have a new equivalent with a different name (in this case `article.cls`). Dummy files with the old names that issue this message and input the right file are provided.

\LaTeX{} Warning: Label ‘...’ multiply defined.
Two `\label` or `\bibitem` commands have defined keys with the same name (Sections 9.2.1 and 9.3). Even after the correction has been made, this message will be printed once more to the monitor since the information that led to it is to be found in the `.aux` file from the last processing run. On the run after that, it should be gone.

\LaTeX{} Warning: Label(s) may have changed.
Rerun to get cross-references right.
The printed outputs from the `\ref`, `\pageref`, and `\cite` commands may be incorrect since their values have been altered during the processing. \LaTeX{} must be run once more so that the right values are used.

\LaTeX{} Warning: Marginpar on page ... moved.
A marginal note on the given page has been shifted downwards to prevent it from being too close to another one. This means that it will not be beside the line of text where the `\marginpar` command was actually given.
LaTeX Warning: No \author given.

The \maketitle command has been issued without a previous \author command. Unlike a missing \title command, this is not an error, but merely peculiar. The warning is printed just in case you did mean to specify an author.

LaTeX Warning: Optional argument of \twocolumn too tall on page.

The \twocolumn command (Section 3.2.7) starts a new page and switches to two-column formatting. The text in the optional argument is printed in one wide column above the double column text. If that text is too large to fit onto one page, this warning is printed.

LaTeX Warning: \oval, \circle, or \line size unavailable on input line ....

The size specified for an \oval, \circle, or slanted \line command in a picture environment is too small for \LaTeX{} to print.

LaTeX Warning: Reference ‘...’ on page ... undefined on input line ....

The marker name in a \ref or \pageref command has not been defined by a \label command in the previous processing run (Section 9.2.1). If this message does not disappear on the next run, the corresponding \label command is missing.

LaTeX Warning: Text page ... contains only floats.

This message points out that the float style parameters (Section 7.3) have excluded any regular text from appearing on the specified page. This is not necessarily bad, but you might want to check that page visually.

LaTeX Warning: There were multiply-defined labels.

This message is printed at the end of the \LaTeX{} run if there were any \label or \bibitem commands that used the same marker name more than once. A warning is also printed near the beginning of the run for each repeated marker name.

LaTeX Warning: There were undefined references.

This message is printed at the end of the \LaTeX{} run if there were any \ref or \pageref commands whose markers had not been defined during a previous run. A warning is also printed earlier for each undefined marker used. If it does not disappear on the next run, then the corresponding \label is missing, maybe due to a typing error in the marker text.
Appendix C. Error Messages

C.5.2 \LaTeX{} package warnings

The class and package warnings are those that check the names and versions of the loaded files against those requested, or the use of options, or whether the \texttt{filecontents} environment has written some text to a file. The commands involved are described in Section D.2.9.

Classes and packages can also issue their own warnings which are peculiar to them. These cannot be listed here, since they depend entirely on the class or package itself.

\texttt{LaTeX Warning: File ‘...’ already exists on the system.}
\hspace{1em}Not generating it from this source.

The \texttt{filecontents} environment is \textit{not} extracting text from the main file because it has discovered a file of the same name already on the system.

\texttt{LaTeX Warning: Unused global option(s):.}

Options specified in the \texttt{\documentclass} statement are \textit{global}, meaning they can apply to the class and/or to any following packages. However, if no class or package recognizes any of these options, this warning is printed, followed by a list of the unused options.

\texttt{LaTeX Warning: Writing file ‘...’.}

The \texttt{filecontents} environment (Section D.2.9) is extracting text out of the main file and writing it to a file of the given name.

\texttt{LaTeX Warning: You have requested class/package ‘...’,}
\hspace{1em}but the class/package provides ‘...’.

The name of the class or package as given by its internal identifying command \texttt{\ProvidesClass} or \texttt{\ProvidesPackage} does not agree with the file that was read in with \texttt{\usepackage} or \texttt{\RequirePackage}.

\texttt{LaTeX Warning: You have requested, on input line ..., version ‘...’ of class/package ...,}
\hspace{1em}but only version ‘...’ is available.

The date of a class or package file, as given by its internal identifying command \texttt{\ProvidesClass} or \texttt{\ProvidesPackage}, is earlier than that asked for by the \texttt{\usepackage} or \texttt{\RequirePackage} command. The class or package may not have all the features that the inputting file expects.

\texttt{LaTeX Warning: You have requested release ‘...’ of \LaTeX{},}
\hspace{1em}but only release ‘...’ is available.

The date of your \LaTeX{} version is earlier than that specified for some input file in a \texttt{\NeedsTeXFormat} command (Section D.2.1). Your \LaTeX{} may not provide all the features needed by that file.
C.5.3 \LaTeX\ font warnings

Font warnings are those involving the NFSS commands (Appendix A). They are indicated by the text: \LaTeX\ Font Warning: plus the warning text.

\LaTeX\ Font Warning: Command ... invalid in math mode.

A command that should only appear in text mode has been given in math mode. The command is simply ignored. The commands \texttt{\textbackslash mathversion}, \texttt{\textbackslash boldmath}, \texttt{\textbackslash unboldmath}, and \texttt{\textbackslash em} lead to this message. There are other commands that produce an \textit{error} message with the same text.

\LaTeX\ Font Warning: Command \texttt{\textbackslash tracingfonts} not provided.

(Font) Use the ‘tracefnt’ package.

(Font) Command found: on input line ....

The font tracing diagnostic tool \texttt{\textbackslash tracingfonts} may only be used if the \texttt{tracefnt} package (page 393) has been loaded. Otherwise, this command is ignored.

\LaTeX\ Font Warning: Encoding ‘...’ has changed to ‘...’ for symbol font ‘...’ in the math version ‘...’.

To make use of the specified symbol font in the given math version, it was necessary to change the font encoding temporarily.

\LaTeX\ Font Warning: Font shape ‘...' in size <...> not available size <...> substituted.

No font has been defined for the size and shape requested, so a substitute size is used instead.

\LaTeX\ Font Warning: Font shape ‘...' undefined using ‘...' instead.

The shape attribute that has been requested is unknown, or has not been defined, so a substitute shape will be used instead.

C.5.4 \TeX\ warnings

A \TeX\ warning is recognized by the fact that it is not an error message (not prefixed with \texttt{!}) and that the processing is not halted. The most common \TeX\ warnings are:

\texttt{Overfull \textbackslash hbox ....}

\TeX\ could not break this line in a reasonable way so part of it will extend into the right margin. The rest of the information in the message can be of assistance.
For example, with the complete warning message

\textbf{Overfull \hbox (17.2122pt too wide) in paragraph at lines 4--6} \\
\texttt{If \LaTeX\ can-not find an ap-pro-pri-ate} \\
\texttt{spot to di-vide a word at the end of the line, as right} \\
\texttt{the line is part of the paragraph in lines 4 to 6. The} \\
\texttt{font used is designated by its attributes \texttt{OT1/cmr/m/n/10}. The} \\
texttt{text of the problem line is if ... ... right here aaaaaaaaaaaaaaaaaaaaaaaa. Possible} \\
\texttt{word divisions are shown with hyphens, such as di-vide. The last word} \\
\texttt{cannot be divided, which is the cause of the problem.} \\

A way around this is to include some suggested hyphenations with the \texttt{-} command, such as aaaaaaaaa\texttt{-}aaaaaaaaaaaaaaaa. Similarly, a \texttt{\linebreak} command before the problem word, or setting the whole paragraph in a \texttt{sloppypar} environment, will get rid of this warning message.

If a badly broken line extends only a tiny bit into the right margin, say 1 pt or less, in most cases this will hardly be noticed and may be left as it is. A sample output should be printed just to verify the appearance.

\textbf{Underfull \vbox} ....

This warning occurs very rarely. \LaTeX\ could not break the page properly so that the text extends beyond the bottom of the page. More often \LaTeX\ sets less text on a page than too much. Thus this warning arises only when the page contains a very large vertical box, higher than the value of \texttt{\textheight}, such as a long table.

\textbf{Underfull \hbox} ....

This is the opposite of the \textbf{Overfull \hbox} warning. It appears when \LaTeX\ has filled a line right and left justified, but with so much interword spacing that it considers the appearance to be undesirable. This is often the result of a \texttt{sloppypar} environment, a \texttt{sloppy} declaration, or a \texttt{\linebreak} command. It may also come about after an inappropriate application of a \texttt{\} or \texttt{\newline} command, such as two \texttt{\} commands one after the other. The additional information in the warning message contains the text of the badly formatted line plus an evaluation of the ‘badness’ of the word spacing.

If in the example of the \textbf{Overfull \hbox} a \texttt{\linebreak} is included in the text at the spot ..., as right\texttt{\linebreak} here, then the warning becomes

\textbf{Underfull \hbox (badness 5504) in paragraph at lines 4--6} \\
\texttt{If \LaTeX\ can-not find an ap-pro-pri-ate} \\
\texttt{spot to di-vide a word at the end of the line, as right} \\
\texttt{which states that the paragraph in lines 4 to 6 contains an output line in which} \\
\texttt{the interword spacing may be unacceptably wide. The text of this line reads If} \\
\texttt{... ..., as right and is set in the font \texttt{OT1/cmr/m/n/10}. The evaluation}
badness 5504 is \TeX's estimate of how unacceptable the spacing is: \textit{the smaller this number the better}.

Underfull \texttt{\vbox} ....

The page has been broken with head and foot justified, but \TeX judges the amount of inter-paragraph spacing to be possibly unacceptable. The \texttt{badness} number here corresponds to the quantity with the same name from the \texttt{Underfull \texttt{\hbox}} warning.

## C.6 Search for subtle errors

At some point or another you will encounter an error message for which you cannot identify any cause, try as you may. For such devious errors, we recommend the following search strategy:

1. Copy the file twice into a \textit{previous} and a \textit{working} copy (in addition to the original, which remains untouched during this search).

2. In the working copy, find the outermost environment where the error occurred and remove one or more inner environments. If there are no inner environments, shorten the remaining text. Process the file with \LaTeX once more.

3. If the error still occurs, copy the shortened working copy to the previous copy and repeat step 2. If the outer environment in step 2 is \texttt{\begin{document} ... \end{document}}, the shortening may be carried out by simply inserting \texttt{\end{document}} at some earlier point.

4. If the error is no longer in the shortened working copy, copy the previous copy back to the working copy. The error is still present in this version. Remove less of the text than last time and repeat steps 2 to 4.

5. If the error is found to be in the next innermost environment, repeat the procedure for this environment with steps 2 to 4.

With this strategy, the error may be localized to one command or to the innermost environment with only a small remaining structure. If the error still cannot be identified in spite of its being precisely localized, seek help from a more experienced colleague or from the computing center. However, it is normally possible to recognize the mistake once the position of the error has been found.

It does happen that even though the error has been corrected, the same error message appears on the next \LaTeX process run. This is because of the internal transfer in information through the \LaTeX auxiliary files, which
are always one run behind on the current situation. For example, if there is a mistake in one of the sectioning commands, then after it has been eliminated, the faulty entry still exists in the .toc file. If the document contains the \tableofcontents command, \LaTeX will read in that .toc file on the next run, and issue the error message once more, since a new .toc file is only created after a successful processing.

In this case, the .toc file should also be edited and the error removed. If that is not possible, the file should be erased and the corrected source file processed twice with \LaTeX. If the error was in \caption, \addcontentsline, or \addtocontents, the same applies to the corresponding .lof or .lot file.

Occasionally the .aux file itself must be erased to prevent an error in it from being repeated even after the .tex file has been repaired. Here one must be careful that the command \nofiles has not been given in the preamble, for then a corrected .aux file will not be generated after a further \LaTeX process run.
In this appendix, we present the special commands that were designed for class and package files, with several examples of useful packages. We also explain the system for integrating documentation with coding in a single file, the system used for the core \LaTeX files and standard packages.

\section*{D.1 Class and package files}

\subsection*{D.1.1 The \LaTeX concept, an open system}

The wealth of contributed \LaTeX programming was probably never anticipated by Leslie Lamport when he released \LaTeX. It is now a fact of life, and indeed one of the great strengths of the system. The \LaTeX Team not only accommodates such ‘foreign’ extensions, it actually supports and encourages them, as witnessed by the copious presentations of such packages in \textit{The \LaTeX Companion} (Goossens \textit{et al.}, 1994) as well as in this book.

And this is the way it should be. The extensions have been written by people who needed them, who realized that \LaTeX was missing something vital for them. On the other hand, to add all of them to the basic \LaTeX installation would overload it with features that 90\% of the users would never require. The philosophy now is that \LaTeX provides a fundamental core, or kernel, which is extended first by the standard class files, and then by the myriad of contributed packages and other classes.

It is the role of the \LaTeX Team to establish guidelines for programming, to ensure that packages do not clash needlessly with the kernel, or with each other, and to provide a basis of stability so that useful packages continue to operate through further updates to the kernel and the standard classes. These \LaTeX features for class and package control, together with a set of programming tools, offers an enhanced degree of reliability and durability, both among packages and against future updates of the kernel.
D.1.2 Levels of commands

There are a number of levels of commands with varying degrees of security for the future:

**user commands** (highest level) described in this and the other manuals, consisting of lower case letters, such as `\texttt{}`, are part of the \LaTeX\ external definition to be supported forever;

**class and package commands** with longish names of mixed upper and lower cases (like `\NeedsTeXFormat`) are intended mainly for programmers, and are also guaranteed; most are preamble-only commands, but there is otherwise no real restriction to class and package files;

**internal \LaTeX\ commands** containing the character `@` in their names can only be used in class and package files; they are not guaranteed forever, although many of them are indispensable for special effects; a programmer makes use of them at the risk that some day his or her package may become obsolete;

**low-level \TeX\ commands** also have names with lower case letters, and no `@`; they should be safe against future evolution of \LaTeX, but even this is not absolutely certain; they should be avoided where possible, as explained below;

**internal private commands** are those used within a contributed class or package file; it is recommended that they all be prefixed with some upper case letters representing the package name and `@` in order to avoid clashes with other packages; for example, `\SK@cite`, from the `showkeys` package.

A question that confronts \LaTeX\ programmers is to what extent the internal \LaTeX\ commands may be used in class and package files. There is always a danger that such commands may vanish in later versions, since they have never been documented in the official books (Lamport, 1985, 1994; Goossens *et al.*, 1994). Like the \TeX\ commands discussed in the next section, their use cannot be forbidden, but one must be aware that a certain degree of risk will accompany them.

The guidelines issued by the \LaTeX\ Team strongly recommend employing the high-level \LaTeX\ commands whenever possible.

- Use `\newcommand` and `\renewcommand` instead of `\def`; if one of the \TeX\ defining commands must be used, because a template is required, or because it must be `\gdef` or `\xdef`, issue a dummy `\newcommand` beforehand to test for a name clash. If it is unimportant whether the command name already exists, issue a dummy
\providecommand followed by \renewcommand. The ability to define commands with one optional argument at the high level removes one reason for wanting to reach down to the lower ones.

- Use \newenvironment and \renewenvironment instead of defining \myenv and \endmyenv.

- Assign values to lengths and glues (rubber lengths) with the \setlength command, rather than by simply equating.

- Avoid the \TeX box commands \setbox, \hbox, and \vbox; use instead \sbox, \mbox, \parbox and the like. With the extra \LaTeX optional arguments, the need for the \TeX equivalents is greatly diminished, and the \LaTeX versions are far more transparent. Moreover, the \LaTeX boxes will function properly with the color package, while the others are unpredictable.

- Issue error and warning messages with \PackageError and \PackageWarning rather than with @latexerr or @warning; the former also inform the user of the source of the message, instead of labeling them all as \TeX messages.

- There is no suggestion that one should exclusively use the \ifthenelse command from the ifthen package (Section 8.3.5) in place of the \TeX conditionals. It seems that this package is offered to simplify employing conditionals, in a manner more consistent with \LaTeX syntax. Although most of the examples in this book use it rather than the \TeX versions, we never employ it ourselves in our own programming.

Adhering to these and similar rules will help ensure that a package will remain fit through future extensions of the \LaTeX kernel.

### D.1.3 \TeX commands

Why should primitive \TeX commands be shunned? To define commands with \def rather than \newcommand must be just as good, and is often unavoidable. Is there really a chance that it might be removed from a future \LaTeX version? The primitives are the building blocks on which all flavors of \LaTeX are constructed. Surely they must remain!

This is not really the point. The primitive \TeX commands form the bedrock of any format, and anything defined with them will always do exactly what the programmer expected. However, the equivalent \LaTeX tools could actually do more as time goes on. The \newcommand checks for name clashes with existing commands, for example. It might even be possible that a debugging device that keeps track of all redefinitions could be added later; any commands defined with \def would be excluded from...
such a scheme. Even now there is something like this to keep track of all files input with middle and high-level commands.

Another example of how low-level programming can go astray is the case of robust commands. Many commands are intrinsically fragile, meaning that they are prematurely interpreted when used as arguments of other commands, but they may be made robust by prefixing them with \protect. In \LaTeX 2.09, several fragile commands were defined to be robust by including the \protect in the definition, as for the \LaTeX logo command:

\begin{verbatim}
\def\LaTeX{\protect\p@LaTeX}
\def\p@LaTeX{...}
\end{verbatim}

The true definition is in the internal \p@LaTeX, not the external \LaTeX. Since the original definition for the logo actually possessed some flaws, several packages included an improved version. Those that simply redefined \LaTeX itself made the command fragile; those that were cleverer only redefined \p@LaTeX, with the result that they are totally left behind in \LaTeX 2ε, where commands are made robust in a completely different (and much better) manner. (Incidentally, the internal definition of the \LaTeX logo today has been greatly improved.)

In spite of the desirability of employing only official \LaTeX commands, there are many occasions when either the internal \LaTeX commands or the \TeX primitives just must be used. The risk of future incompatibility must be taken in order to have a workable package now. However, one should not take this risk lightly, where a high-level equivalent is available.

\section{\LaTeX programming commands}

All the commands described in this section are new to \LaTeX 2ε. They are not essential to class and package files, but they do extend their usefulness and guarantee that they are employed properly.

\subsection{File identification}

Three commands test that the external environment in which the class or package has been inserted is correct. The first of these is

\begin{verbatim}
\NeedsTeXFormat{\texttt{format}}[\texttt{version}]
\end{verbatim}

The first statement in a class or package should be the declaration of the \TeX format needed. Although there are existing formats with other names, only the one named \LaTeX 2ε actually recognizes this statement. All others will immediately issue the error message

\begin{verbatim}
! Undefined control sequence.
\end{verbatim}
1.1 \NeedsTeXFormat {LaTeX2e}

which all by itself is fairly informative.

What is perhaps of more use is the optional version argument, which must contain the date of issue in the form yyyy/mm/dd. If a package makes use of features that were introduced in a certain version, its date should be given, so that if it is used with an earlier version of \LaTeXX, a warning is printed. For example, the command \DeclareRobustCommand did not exist in the preliminary test release of \LaTeXX, but was first introduced with the official release of June 1, 1994. Thus any package containing this command should begin with

\NeedsTeXFormat {LaTeX2e} [1994/06/01]

The form of the date is important, including the zeros and slashes.

This declaration is not limited to class and package files: it may also be issued at the start of the document itself to ensure that it is processed with the right \LaTeX. It must, however, be given in the preamble.

The next two commands identify the class or package file itself:

\ProvidesClass {class} [version]
\ProvidesPackage {package} [version]

In both cases, the version consists of three parts: date, version number, and additional information. The date is in the same format as above, while the version number can be any designation without blanks, and the additional information is text with or without blanks. An example is

\ProvidesPackage {shortpag} [1995/03/24 v1.4 (F. Barnes)]

Only the date part is actually checked by \LaTeX against the date specified in the calling \usepackage command. The version number and additional information are printed out if \listfiles has been requested. However, the above format is necessary for the \GetFileInfo command in the doc package (Section D.7.2).

Both the \documentclass and \usepackage commands (as well as \LoadClass and \RequirePackage) may take an optional argument to specify the earliest acceptable release date for the class/package. For example, with

\documentclass [12pt] {article} [1995/01/01]

if the article class file loaded contains

\ProvidesClass {article} [1994/07/13 v1.2u
   Standard \LaTeX document class]

a warning message is printed. The same procedure applies to the commands \usepackage and \ProvidesPackage.
Appendix D. \LaTeX{} Programming

This system of version checking allows a document to insist that suitable versions of the class and package files are loaded. It assumes, however, that all later versions are fully compatible with earlier ones.

There is a further identifying command for general files, those to be loaded with \texttt{\input}.

\begin{verbatim}
\ProvidesFile{file_name}[version]
\end{verbatim}

There is no checking of the name or version in this case, but both pieces of information will be printed by \texttt{\listfiles}.

D.2.2 Loading further classes and packages

In the main document file, classes are read in by means of the initializing \texttt{\documentclass} command, and packages with \texttt{\usepackage}. Within class and package files, the commands

\begin{verbatim}
\LoadClass[options]{class}[version]
\RequirePackage[options]{package}[version]
\LoadClassWithOptions{class}[version]
\RequirePackageWithOptions{package}[version]
\end{verbatim}

must be used instead. The first allows one class file to load another, with selected options, if desired; the second permits class and package files to load other packages. Only one \texttt{\LoadClass} command may appear within any class file; it may not be called from a package file. Neither command may be invoked in the document file. The \texttt{packages} argument may be a list of several package names, separated by commas.

The \texttt{...WithOptions} variants load the class or package with all those options that were specified for the current one, something that is often required.

How the optional \texttt{version} arguments interact with the corresponding \texttt{\Provides..} command has been explained in the previous section; how the \texttt{options} argument is treated is described below.

D.2.3 Processing options

Both classes and packages may take options which are defined with

\begin{verbatim}
\DeclareOption{option}{code}
\end{verbatim}

where \texttt{option} is the name of the option and \texttt{code} is the set of instructions that it is to execute. Internally, a command named \texttt{\ds@option} is created. Often the code does nothing more than set flags, or input an option file. (\texttt{\RequirePackage} may \texttt{not} be used within the option code!) Two examples from \texttt{article.cls} are

\begin{verbatim}
\DeclareOption{fleqn}{\input{fleqn.clo}}
\DeclareOption{twocolumn}{\setboolean{\@twocolumn}{true}}
\end{verbatim}
A default option is defined with \DeclareOption*, which takes no option name, specifying the code to be executed for all requested options that are undefined.

There are two special commands that may be used only within the code of the default option definition:

\CurrentOption contains the name of the option being processed;

\OptionNotUsed declares \CurrentOption to be unprocessed.

For example, to have a class file emulate \LaTeX 2.09 behavior where all undefined options load a .sty file of the same name, define

\DeclareOption*{\InputIfFileExists{\CurrentOption.sty}{}}{\OptionNotUsed}

which first checks if there is a .sty file of the requested name, and if not declares the option to be unused. Requested options that have not been used (processed) are listed in a warning message.

The options are then processed with the commands

\ExecuteOptions{option_list}
\ProcessOptions
\ProcessOptions*

where \ExecuteOptions calls those commands defined for the options in option_list. This is normally done to establish certain options as being present by default. \ProcessOptions executes all the requested options in the order in which they were defined and then removes them from the list. Options are therefore executed only once by this command. The *-version is similar, except that the options are executed in the order requested.

It is also possible to transfer options to a class or package file with

\PassOptionsToClass{options}{class.name}
\PassOptionsToPackage{options}{package.name}

where options is a list of valid options recognized by the specified class or package file. These commands may be used within the definition of other options. The class or package named must later be loaded with \LoadClass or \RequirePackage.

If the default options for class and package files have not been altered by \DeclareOption*, the standard procedure for handling options that have been requested but are undefined is

- all options requested in the \documentclass statement are designated global; they are considered to apply to all subsequent packages, but not to classes loaded with \LoadClass; if they are not defined in the class, no error or warning is issued;
Appendix D. \LaTeX Programming

- all options requested with other commands, including \LoadClass and the \PassOptionsTo..., are local; if they are not defined in that class or package, an error is issued;

- if there are global options that are defined in neither the class nor any of the packages, a warning is issued;

- options, global and local, are executed in the order in which they are defined in the class and packages, unless \ProcessOptions* has been called in which case they are executed in the order in which they are listed.

D.2.4 Deferred processing

Sometimes, to achieve certain special effects or to avoid possible conflicts with other packages, it is desirable to have some commands executed at the end of the package or class, or at the beginning or end of the document. This can be accomplished with

\AtEndOfClass{cmds}
\AtEndOfPackage{cmds}
\AtBeginDocument{cmds}
\AtEndDocument{cmds}

The first two store away \textit{cmds} to be carried out at the end of the class or package file. They can be used by local configuration files that are read in at the beginning, but contain modifications that should be made at the end so that they are not overwritten by the defaults. The last two declarations store away the \textit{cmds} to be executed with \texttt{\begin{document}} and \texttt{\end{document}} respectively. All of these may be issued more than once, in which case the \textit{cmds} are processed in the order in which they were issued.

The \textit{cmds} stored with \texttt{\AtBeginDocument} are inserted into the processing stream effectively within the preamble, but after the command \texttt{\begin{document}} has done almost everything else that it does. Thus the \textit{cmds} may be considered to be part of the main body but preamble-only commands are also allowed.

D.2.5 Robust commands

Commands may actually be fragile, meaning that if they are used in the arguments of other commands, they could be prematurely interpreted, causing unexpected problems. This happens with moving arguments, those that appear somewhere else other than where they were given: in the table of contents and in running headlines. Complex commands with conditionals or redefinitions are likely to be fragile.
Many intrinsic commands in \LaTeX{} 2.09 were fragile, and it was necessary to precede them with \protect when they appeared in the argument of a \texttt{\section} command, for example. In this case, the command name is transferred rather than its translation. Some commands could be made \textit{robust} using the trick shown on page 440 for the \LaTeX{} command itself.

In today's \LaTeX{}, almost all regular commands are robust. However, the commands a user may define with \texttt{\newcommand}, \texttt{\renewcommand}, and \texttt{\providecommand} (Section 8.3) may very well be fragile. Alternatively, one can define the commands with

\begin{verbatim}
\DeclareRobustCommand{\com_name}[\text{narg}][\text{opt}]{\text{def}}
\end{verbatim}

which has the same syntax as the other defining commands. If the command to be defined already exists, a message is written to the transcript file, and the old definition is overwritten.

Another command with the same syntax simply checks the current definition of \texttt{\com_name}:

\begin{verbatim}
\CheckCommand{\com_name}[\text{narg}][\text{opt}]{\text{def}}
\end{verbatim}

and issues a warning message if the actual definition is not the same as \texttt{\text{def}}, with the same number of arguments, and so on. This is used to ensure that the state of the system is as one expects, and that no previously loaded packages have altered some important definition.

Both \texttt{\DeclareRobustCommand} and \texttt{\CheckCommand} may be called at any point in the document.

### D.2.6 Commands with ‘short’ arguments

Normally, the arguments to user-defined commands are allowed to contain new paragraphs, with the \par command or with a blank line. In \TeX{} jargon, these commands are said to be ‘long’. This is not the standard behavior for commands created with the \TeX{} \texttt{\def} command, where the arguments must be short in order to act as a test for forgotten closing braces.

With the version from December 1, 1994, \LaTeX{} provides *-forms of all the defining commands:

\begin{verbatim}
\newcommand* \renewcommand* \\
\newenvironment* \renewenvironment* \\
\providecommand* \DeclareRobustCommand* \CheckCommand*
\end{verbatim}

which create user-defined commands with ‘short’ arguments in the same way as does \texttt{\def}.

It is recommended that one should almost always take the *-version of these defining commands, unless there is some very good reason to expect that the possible arguments may be ‘long’, that is, contain new paragraphs. Long arguments should be the exception, not the rule.
Appendix D. \LaTeX Programming

D.2.7 Issuing errors and warnings

Classes and packages may be programmed to issue their own error messages and warnings. This is useful to indicate which file is responsible for the message.

Error messages are generated with

\ClassError{\textit{class name}}{error text}{help}
\PackageError{\textit{package name}}{error text}{help}

where \textit{error text} is the message printed to the monitor and to the transcript file, and \textit{help} is additional text printed after the user responds with H. If the texts contain command names that are to be printed literally, they must be preceded by \texttt{\protect}; spaces are generated with \texttt{\space}, and new lines with \texttt{\MessageBreak}. For example,

\PackageError{ghost}{\%
   The \texttt{\protect\textwidth\space is too large}\MessageBreak
   for the paper you have selected}
   {Use a smaller width.}

produces the error message

! Package ghost Error: The \textwidth is too large
(ghost) for the paper you have selected.

See the ghost package documentation for explanation.
Type H <return> for immediate help.

Typing H <return> produces

Use a smaller width.

after which \LaTeX halts again to wait for a response as described in Section C.1.

Warnings may also be issued from classes and packages in a similar way. The difference is that there is no \textit{help} text, and the processing does not stop for a response. The line number of the input file where the warning occurred may be optionally suppressed.

\ClassWarning{\textit{class name}}{warning text}
\ClassWarningNoLine{\textit{class name}}{warning text}
\PackageWarning{\textit{package name}}{warning text}
\PackageWarningNoLine{\textit{package name}}{warning text}

For example, with the warning

\PackageWarning{ghost}
   {This text is haunted}

one obtains the message

Package ghost Warning: This text is haunted on input line 20.
and the processing continues. Warnings may be split into several lines with the \MessageBreak command, just like error texts.

Two last commands of this type are

\ClassInfo{class_name}{info_text}
\PackageInfo{package_name}{info_text}

which write their texts only to the transcript file, and not to the monitor. They are otherwise just like the corresponding NoLine warnings.

D.2.8 Inputting files

Files other than classes and packages may also be input, in which case it is often desirable to make sure that they exist beforehand. Or, alternative actions might be taken depending on the existence of a certain file. These goals are met with

\IfFileExists{file_name}{true}{false}
\InputIfFileExists{file.name}{true}{false}

Both these commands test for the presence of the specified file name in the area that \LaTeX{} is looking for files, and execute true if it is found, otherwise false. In addition, \InputIfFileExists reads in that file after executing true.

These commands are not restricted to the preamble, nor to class or package files. In fact, the regular \input command is defined in terms of them.

Many special classes make use of these commands to read in a local configuration file. For example, the class ltxdoc contains

\InputIfFileExists{ltxdoc.cfg}
{\typeout{Local config file ltxdoc.cfg used}}
{}

just before \ProcessOptions is called. This allows one to have a local configuration that might specify

\PassOptionsToClass{a4paper}{article}

for a European installation, without altering the files that are processed with the ltxdoc class.

D.2.9 Checking files

Although not really part of programming, two \LaTeX{} features to keep track of input files are described here. The first of these, already mentioned in Section 9.1.1, is the command

\listfiles
which may be given in the preamble, even before \documentclass. It causes a list of all input files to be printed at the end of the processing, along with their version and release data. In this way, one has a record of just which files were included, something that may be of use when deciding to send a document file to another installation for processing there. Since any non-standard files may also have to be included, these may be more readily identified from such a listing.

For example, the simple document file

```latex
\documentclass{article}
\usepackage{ifthen}
\listfiles
\begin{document}
\input{mymacros}
This is \TeX.
\end{document}
```

produces the listing

*File List*

article.cls 2001/04/21 v1.4e Standard \LaTeX\ document class
size10.clo 2001/04/21 v1.4e Standard \LaTeX\ file (size option)
ifthen.sty 2001/05/26 v1.1c Standard \LaTeX\ ifthen package
mymacros.tex
***********

In this case, the local file mymacros.tex contains no version information because it is missing a \ProvidesFile command.

What should one do if a local file, such as mymacros.tex above, is needed for the processing of a document file that is to be sent elsewhere? One could send it along with the main file, but that requires giving the recipient more instructions on what to do. Or, its contents could simply be included in the main file, for shipping purposes. For a package file, this is not so easy, since internal commands containing the @ sign would cause trouble, and the options would not be handled properly. For this purpose, we have the environment

```latex
\begin{filecontents}{file name}
file contents
\end{filecontents}
```

which may only appear at the very beginning of the document, before the \documentclass command. It tests to see if there is a file on the system with the name file name, and if not, it writes its contents literally to a file of that name. This may be a package file that is subsequently input with \usepackage. In this way, the missing non-standard files can be ported together with the main document file.

If we extend the above simple example by including at the very start
D.2. \LaTeX programming commands

\begin{filecontents}{mymacros}
\newcommand{\te}{the end}
\end{filecontents}

the newly written file mymacros.tex contains

\begin{verbatim}
%% \LaTeX{}2e file 'mymacros'
%% generated by the 'filecontents' environment
%% from source 'mydoc' on 2003/01/31.
%%
\newcommand{\te}{the end}
\end{verbatim}

Note that the filecontents environment adds some comment lines
to explain where the new file came from. If this is undesirable, the
filecontents* environment may be used instead.

D.2.10 Useful internal commands

Notwithstanding the guidelines in Section D.1.2 that recommend avoiding internal
\LaTeX{} commands, there are a number that are fairly fundamental, and indeed form
many of the building blocks of the \LaTeX{} kernel and many standard packages. Since
they are still internal commands, they are not guaranteed for all future updates.
However, if they were to vanish, many of the interesting extension packages
provided by the \LaTeX{} Team itself would have to be drastically overhauled. We
merely present them briefly here for the sake of the bolder user.

\@namedef{\cmd}{\def}
\@nameuse{\cmd}

define and execute a command named \cmd, where the backslash is not included
in the command name. This name may contain any characters, even those
normally forbidden in command names.

\@ifundefined{\cmd}{\true}{\false}

executes \texttt{true} if the command \cmd does not exist, else \texttt{false}. Again, the backslash
is not included in \cmd, and any characters may appear in the command name.
This test is often used to define commands conditionally, a task that has been
taken over by \providecommand. It may also be employed to determine whether
the main class is \texttt{article}-like or not: \@ifundefined{\chapter}{{\true}}{{\false}} tests
for the existence of the \texttt{\chapter} command.

\@ifnextchar{\true}{\false}

tests if the next character is \texttt{char}, and if so, executes \texttt{true}, else \texttt{false}. This
command is traditionally used to define commands with optional arguments,
where \texttt{char} is \texttt{[}. The extended syntax of \texttt{\newcommand} offers a high-level means
of achieving this.

\@ifstar{\true}{\false}
tests if the next character is a star *, and if so, executes true, else false. It is used to define *-forms of commands and environments, something that still cannot be done at the high level.

\@for \obj := \list \do {cmds}

where \list is a command that is defined to be a list of elements separated by commas, and \obj is successively set equal to each of these elements while the code cmds is executed once for each element. For example,

\newcommand{\set}{start,middle,end}
\@for \xx:=\set \do {This is the \xx. }

prints 'This is the start. This is the middle. This is the end.'

### D.2.11 Useful \TeX\ commands

Many of the most sophisticated features of \LaTeX and its packages can only be programmed with the help of Plain \TeX commands. These are described not only in \textit{The \TeXbook} (Knuth, 1986a), but also in the excellent reference manual by Eijkhout (1992), \textit{\TeX by Topic, a \TeXnician's Reference}.

We do not intend this book to a manual for \LaTeX; nevertheless, there are a few common \LaTeX commands that appear in many packages, and even in the examples to follow. A brief description of what they do will aid the understanding of these codings. A true \TeXpert or \TeXnician can skip this section altogether.

\def\cmd#1#2...{definition}

is the standard defining command in \TeX. It is the equivalent of \texttt{\newcommand*} except that there is no check for name clashes, and the arguments are specified differently. For example, a command \texttt{\Exp} to write scientific notation can be defined as

\def\Exp#1#2{\ensuremath{#1\times10^{#2}}}

or as \texttt{\newcommand*{\Exp}[2]{\ensuremath{#1\times10^{#2}}}}

In both cases, \texttt{\Exp{1.1}{4}} produces $1.1 \times 10^4$. However, \texttt{\def} can go further: it can put the arguments in a template, such as

\def\Exp#1(#2){\ensuremath{#1\times10^{#2}}}

to allow the more convenient notation \texttt{\Exp{1.1}{4}}, something that cannot be produced with \texttt{\newcommand}. The \texttt{\def} command is often used when a command is to be defined without knowing (or caring) whether its name already exists, or when a template is needed.

\gdef \edef \xdef

are variations on \texttt{\def}: the first makes a \textit{global} definition, valid even outside the current environment or \{\ldots\} bracketing; the second is an \textit{expanded} definition, such that any commands in it have their meanings and not the command itself inserted in the definition; the last is a combination of the other two, expanded and global.
D.3 Sample packages

We present here some demonstration packages to illustrate the programming commands of the previous section. These packages are not trivial and are all useful in their own right, even though their functionality is covered by others described in this book.
D.3.1 Modifying the text page size

The standard \LaTeX \ classes set the text size parameters \textwidth and \textheight according to the size option specified in \documentclass, such as a4paper or legalpaper. They also adjust the margins so that the text is centered both horizontally and vertically. The value of \textwidth is restricted, however, so that there are at most 60–70 characters per line, this being considered optimal by the rules of typography.

Often one wants to ignore this limitation and use the paper to its maximum extent. The geometry package from Section 3.2.6 does this and much more. Here we present a simpler package for this single function.

The fullpage package presented here uses the values of \paperwidth and \paperheight, which have been set according to the paper size option, to produce a margin of one inch on all sides. It even considers what the page style is so that room for head and footlines may be taken into account. Optionally, it may set an even narrower margin of 1.5 cm.

Before proceeding, it might be useful to review the page format parameters shown in the figures on pages 48 and 602.

The package file begins by stating that it needs \LaTeX\ and by identifying itself. The package information contains the date in prescribed form, the version number, and additionally the initials of the author. The ifthen package is then loaded since conditionals will be required.

\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{fullpage}[2002/02/15 1.1 (PWD)]
\RequirePackage{ifthen}

Next the options are prepared. These are to be in and cm for margins of 1 in and 1.5 cm respectively (1 cm is really too narrow). A special length is used to store the margin value. This is a private, internal command, conforming to the convention mentioned on page 438.

\newlength{\FP@margin}
\DeclareOption{in}{\setlength{\FP@margin}{1in}}
\DeclareOption{cm}{\setlength{\FP@margin}{1.5cm}}

Furthermore, the four standard page styles are also to be option names. They will set two internal boolean switches and the page style.

\newboolean{FP@plain}
\newboolean{FP@empty}
\DeclareOption{plain}{\setboolean{FP@plain}{true}
  \setboolean{FP@empty}{false}
  \pagestyle{plain}}
\DeclareOption{empty}{\setboolean{FP@plain}{true}
  \setboolean{FP@empty}{true}
  \pagestyle{empty}}
\DeclareOption{headings}{\setboolean{FP@plain}{false}
Finally, the default set of options is executed, and then the selected options are processed, in this case, in the order specified. The reason for this is that if more than one page style has been given, the last one should dominate.

\ExecuteOptions{in,plain}
\ProcessOptions*

Now the calculation can begin. First, for plain and empty, there are no headlines, so set the relevant parameters to zero. This is when FP@plain is \texttt{true}. Then set the space reserved for the footline to zero for the empty page style (FP@empty is \texttt{true}). For headings and myheadings, this space is maintained since these styles normally have a plain page (with footline) on the first page.

\ifthenelse{\boolean{FP@plain}}{\setlength{\headheight}{0pt}\setlength{\headsep}{0pt}}{}
\ifthenelse{\boolean{FP@empty}}{\setlength{\footskip}{0pt}}{}

With all the margins and headline and footline spacings set, the actual calculation can be made. Recall that the driver program leaves a 1 inch margin at the left and above, which must be subtracted from \texttt{topmargin} and \texttt{oddsidemargin}.

\setlength{\textwidth}{\paperwidth}
\addtolength{\textwidth}{-2\FP@margin}
\setlength{\oddsidemargin}{\FP@margin}
\addtolength{\oddsidemargin}{-1in}
\setlength{\evensidemargin}{\oddsidemargin}
\setlength{\textwidth}{\paperwidth}
\addtolength{\textwidth}{-2\FP@margin}
\setlength{\textheight}{\paperheight}
\addtolength{\textheight}{-\headheight}
\addtolength{\textheight}{-\headsep}
\addtolength{\textheight}{-\footskip}
\addtolength{\textheight}{-2\FP@margin}
\setlength{\topmargin}{\FP@margin}
\addtolength{\topmargin}{-1in}

Remember that \texttt{paperheight} and \texttt{paperwidth} are set to the full dimensions of the paper by the paper size option. If this is wrongly specified, of course, the final result will be incorrect.

This completes the coding for the package file fullpage.sty. An example of how it might be invoked is
D.3.2 Redesigning the head and footlines

Something that is often demanded by \LaTeX{} users is the possibility of altering the head and footlines that appear on each page. Standard \LaTeX{} does provide a limited number of page styles (Section 3.2) but these are not always sufficient. The most flexible of them is the myheadings page style which allows the author to determine the text of the running heads with \markright{} and \markboth{}. However, the general format, including font style and placement of the page number, is still fixed by \LaTeX{}.

The fancyhdr package by Piet van Oostrum (Section 3.2.2) solves all this very nicely for most cases. Nevertheless we present this example to show how it is done internally, and to illustrate how some other special effects can be achieved.

We wish to modify the headline for documentation of large projects requiring particular information in the running head. Apart from a short version of the title and the section and page numbers, they might demand the document serial number, date, and version number. It would be useful to have a generalized documentation page style that allows these entries to be specified outside of the page style definition itself.

Since this coding is only feasible for the article class, we will create a new class, called dochead, that inputs article.cls and then defines a new page style. Actually this class could contain many other special features of which our new page style is only one.

Again we start by declaring the \TeX{} version that we need (release 4 from December 1, 1995), and by identifying the class file.

\NeedsTeXFormat{LaTeX2e}[1995/12/01]
\ProvidesClass{dochead}[2002/04/27 1.5 (PWD)]

This class has no options of its own, passing all specified ones on to the underlying article class by means of the \LoadClassWithOptions command, which only became available with release 4.

\LoadClassWithOptions{article}

Now come the commands that are used to enter the four pieces of information for the headline: short title, date, serial number, and version. Each of these commands stores its argument in an internal \DH@ command, each of which is initially set to be empty. Finally, the entering commands are declared to be allowed only in the preamble, since it would be disastrous if they were called after the main text had started.

\newcommand*{\DocTitle}[1] {\renewcommand*{\DH@title}{#1}}
\newcommand*{\DocDate}[1] {\renewcommand*{\DH@date}{#1}}
The \@onlypreamble command is an internal \LaTeX{} one.

**Note:** The commands \MakeUppercase and \MakeLowercase (not used here) set the case of their arguments. The former is used here to ensure that the document number always appears in upper case letters.

We now define the new page style, dochead, which means we must create a command \ps@dochead, to be executed by \pagestyle{dochead}. What this command must do is redefine the four internals \@oddhead, \@evenhead, \@oddfoot, and \@evenfoot. We make the footlines empty, and the even and odd headlines identical. That headline is to be a minipage, the full width of the page, containing a table with the relevant document information.

\newcommand*{\ps@dochead}{% 
  \renewcommand*{\@oddhead}{% 
    \begin{minipage}{\textwidth}\normalfont 
      \begin{tabular*}{\textwidth}{@{}l@{\extracolsep{\fill}}l@{\extracolsep{\fill}}l}\DH@number & Version & \DH@version \DH@title & Section & \thesection \Date: \DH@date & Page & \thepage \end{tabular*}\vspace{0.5ex} \rule{\textwidth}{0.6pt}% 
    \end{minipage}} 
  \renewcommand*{\@evenhead}{\@oddhead} 
  \renewcommand*{\@oddfoot}{} 
  \renewcommand*{\@evenfoot}{} 
}% 
\pagestyle{dochead}

The last line activates the new page style.

It is also necessary to increase the size of \headheight and \headsep because our headline is much higher than normal. We pick a height of 3.5 times a regular line.

\setlength{\headheight}{3.5\baselineskip} 
\setlength{\headsep}{3em}

We now add a further refinement. Such documentation normally wants the pages to be numbered within the sections. Therefore, we want to redefine the command \thepage which prints the page number from the page counter (Section 8.1.4), and we need to modify the \section command to start a new page and to reset the page counter. This is done by storing the current definition of \section and then redefining it.
Appendix D. \LaTeX Programming

\let\DH@section=\section
\renewcommand*{\thepage}{\thesection-\arabic{page}}
\renewcommand*{\section}{\newpage\setcounter{page}{1}\DH@section}

Notice that the new \section command calls the old one now stored in our own internal command \DH@section.

If the above is written to a file named dochead.cls, the following source text could be used to invoke it:

\documentclass[12pt,a4paper]{dochead}
\DocTitle{Spacecraft Cleanliness}
\DocNumber{Esa--xy--123}
\DocDate{2003 Feb 26}
\DocVersion{4.2}
\begin{document}

The headline on the fourth page of Section 3 would then appear as below.

| ESA–XY–123 | Version: 4.2 |
| Spacecraft Cleanliness | Section: 3 |
| Date: 2003 Feb 26 | Page : 3-4 |

With this example as a model, it should not be difficult to create similar headlines for other applications with different entries.

D.3.3 Reprogramming the sectioning commands

Another aspect of \LaTeX formatting that one might want to alter is the appearance of the sectioning titles. The fonts used, as well as their sizes and the spacing around them, are all rigidly set in the class files. We know users who feel so frustrated by the predefined sectioning that they type in their own by hand, including the numbers, something that completely violates the \LaTeX concept of logical markup.

In fact, all the sectioning commands from \section to \subparagraph are defined by means of a generalized internal section command with the following syntax:

```
@startsection{sec-name}{level}{indent}{pre-skip} {post-skip}{style}* [short title] {title}
```

where both the * and [short title] are optional. A command like \section is defined to be \@startsection with all the arguments up to and including style; the remaining optional *, short title, and title are given as arguments to \section itself (Section 3.3.3). The meanings of these arguments are:
sec-name: the name of the section level, for example subsection; this name is used to select the appropriate counter and to enter the title correctly in the table of contents and page headlines;

level: is the number in the sectioning hierarchy as described on page 56, 1 for \section, 2 for \subsection ...;

indent: is the amount of indentation from the left margin;

pre-skip: is a rubber length, the absolute value of which is the space inserted above the title; if this is negative, the first paragraph following the section title is not indented;

post-skip: is a rubber length; if positive, it is the space inserted below the title; if negative, its absolute value is the space between the title and the subsequent run-in text;

style: font declarations issued when printing the section title, such as \bfseries and \Large;

*: optional; if present, the section counter is not incremented, the section is not numbered, no entry is made in the table of contents;

short title: optional alternative text for the table of contents and page headline; may only be given if the * is absent;

title: the text printed as the section title; if short title is missing, this text is also entered in the table of contents and page headline.

The font style of the title is determined by the declarations in style; these must be declarations, like \bfseries, and not commands with arguments like \textbf. However, with the release from June 1, 1996, the last item in the style list may indeed be a command with an argument. This permits something like \bfseries\MakeUppercase to force the title to be bold face capital letters.

Rubber lengths with generous stretch and shrink should be given for pre-skip and post-skip to allow \LaTeX more freedom in avoiding bad page breaks.

The section number is written with the command \@seccntformat that takes one argument, the name of the sectioning counter (sec-name above) that is to be printed. This may be redefined for special effects, as illustrated in the example below.

Since both \@startsection and \@seccntformat contain @, they are internal commands that can only be used in a class or package file. We give here a brief package, mysects.sty, which redefines all the sectioning commands. The values of pre-skip and post-skip are those in the standard article.cls.
The date of the fifth \LaTeX release is specified because we want to use a command in the font styles. We start with the \texttt{section} command which is to be centered, \texttt{Large} and upper case.

\renewcommand{\section}{\@startsection{section}{1}{0pt}%-3.5ex plus -1ex minus -.2ex\%\{2.3ex plus 2ex\}\{\centering\normalfont\Large\MakeUppercase\}}

The subsections and subsubsection titles are in small caps and bold slanted fonts respectively.

\renewcommand{\subsection}{\@startsection{subsection}{2}{20pt}%-3.25ex plus -1ex minus -.2ex\%\{1.5ex plus .2ex\}\{\normalfont\large\scshape\}}
\renewcommand{\subsubsection}{\@startsection{subsubsection}{3}{10pt}%-3.25ex plus -1ex minus -.2ex\%\{1.5ex plus .2ex\}\{\normalfont\normalsize\bfseries\scshape\}}

The paragraph and subparagraph titles are to be run-in types (negative \textit{post-skip}).

\renewcommand{\paragraph}{\@startsection{paragraph}{4}{3.25ex plus 1ex minus .2ex\%\{-1em\}\{\normalfont\normalsize\underline\}}
\renewcommand{\subparagraph}{\@startsection{subparagraph}{5}{\parindent}%-3.25ex plus 1ex minus .2ex\%\{-1em\}\{\normalfont\normalsize\itshape\}}

Finally, redefine the section numbers to have a period after them.

\renewcommand{\@seccntformat}{\@nameuse{the#1}. \quad}

As an example of output with \texttt{mysects}, consider the short document:

\documentclass{article}
\usepackage{mysects}
\setcounter{secnumdepth}{3}
\begin{document}
\section{Historical Outline}
\subsection{Medieval Life}
\subsubsection{The Role of the Priest}
\paragraph{In the Church}
The priest fulfills several functions.

\subparagraph{At the Altar}
Here the priest was master of ceremonies.

which produces output:

1. HISTORICAL OUTLINE

1.1. MEDIEVAL LIFE

1.1.1. The Role of the Priest

In the Church The priest fulfills several functions.

At the Altar Here the priest was master of ceremonies.

Because the counter \texttt{secnumdepth} was set to 3, numbering only goes to the third level.

D.4 Changing preprogrammed text

D.4.1 Changing explicit names

There are a number of titles that appear automatically in \LaTeX, such as ‘Contents’, ‘Bibliography’, ‘Chapter’. For works in languages other than English, it is necessary that they be replaced by their translations. And even in English, an author might prefer say ‘Summary’ in place of ‘Abstract’. All such explicit words are to be found in certain name commands that may be redefined as one pleases.

These name commands are not defined in the basic \LaTeX\ format itself, but rather in the various class files, as they are needed. Thus \texttt{\chaptername} exists only in classes \texttt{book} and \texttt{report}, but not in \texttt{article}. This means, if a package were to redefine the word ‘Chapter’ for all classes, it must do so with something like

\begin{verbatim}
\providecommand*{\chaptername}{}
\renewcommand*{\chaptername}{Chapitre}
\end{verbatim}

The standard set of name commands and their initial values are:

(defined in book, report, and article classes)

\begin{verbatim}
\contentsname {Contents}
\listfigurename {List of Figures}
\listtablename {List of Tables}
\indexname {Index}
\figurename {Figure}
\tablename {Table}
\end{verbatim}
The redefinition of these name commands is a fundamental part of the multilingual babel system (Chapter 11). The commands are not redefined directly but rather by means of certain \captionslanguage such as \captionsgerman and \captionsenglish that allow for convenient switching back and forth between different languages. For example,

\newcommand*{\captionsgerman}{%
\renewcommand*{\contentsname}{Inhaltsverzeichnis}
...\renewcommand*{\alsoname}{siehe auch}}

\section*{The date}

The \today command for the current date is another one that outputs explicit English words. Its standard definition conforms to the American style of giving dates, that is ‘January 15, 2004’. If one wants to redefine this, either for the British style (15th January 2004), or for another language altogether, the best method is to follow the example of the other names commands: do not redefine \today directly, but create commands that allow one to switch back and forth.

For example, we can define \dateUSenglish and \dateenglish making use of the internal \TeX counters \year, \month, \day and the \TeX
D.5. Direct typing of special letters

\ifcase command (page 451).

\newcommand*{\dateUSenglish}{{\renewcommand*{\today}{%}
  \ifcase\month \or
  January\or February\or March\or April\or May\or June\or
  July\or August\or September\or October\or November\or
  December\fi \space \number\day, \number\year}}

\newcommand*{\dateenglish}{{\renewcommand*{\today}{%}
  \number\day \ifcase\day \or
  st\or nd\or rd\or th\or th\or th\or th\or th\or th\or th\or th\or
  th\or th\or th\or th\or th\or th\or th\or th\or
  st\or nd\or rd\or th\or th\or th\or th\or th\or th\or th\or
  st\fi \space \ifcase\month \or
  January\or February\or March\or April\or May\or June\or
  July\or August\or September\or October\or November\or
  December\fi \space \number\year}}

Definitions for other languages can be modeled after these examples.

\section{D.5 Direct typing of special letters}

In Section 2.5.9, we explained the \texttt{inputenc} package that permits direct typing of accented and special letters, provided one has an appropriate keyboard and display font. For example, on a German keyboard set up for Windows, typing the key marked ‘ß’ inserts character 223, which is displayed on the author's monitor as the 'eszet' letter for which the regular \LaTeX input is \texttt{\ss}. The source file however, receives only the code number 223, which \LaTeX would normally reject as unknown. The \texttt{inputenc} package with the \texttt{ansinew} option tells it that character 223 is to be treated as \texttt{\ss}. This will have the same result even if the file is displayed on a monitor set up for a different coding.

Table D.1 lists the current set of encoding options for the package. Others may be added in the future. For each of these options, there is corresponding \texttt{.def} file defining the extended characters with the commands

\begin{verbatim}
  \DeclareInputText\{pos\}\{text\} \quad or \quad \DeclareInputMath\{pos\}\{math\}
\end{verbatim}

which assign text or math coding to the character pos. For example, \texttt{latin1.def} contains

\begin{verbatim}
  \DeclareInputText\{223\}\{\ss\}
\end{verbatim}

to translate the input character code 223 to the \LaTeX command \texttt{\ss}.

Some \TeX installations automatically contain such conversions for their own local input coding scheme; files written under such a system without
Table D.1: Input coding schemes for \inputenc package

<table>
<thead>
<tr>
<th>Coding Scheme</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ascii</td>
<td>7-bit ASCII encoding, characters 32–127 only</td>
</tr>
<tr>
<td>latin1</td>
<td>ISO Latin-1 encoding (Western Europe)</td>
</tr>
<tr>
<td>latin2</td>
<td>ISO Latin-2 encoding (Eastern Europe)</td>
</tr>
<tr>
<td>latin3</td>
<td>ISO Latin-3 encoding (Catalan, Esperanto, Galacian, Maltese)</td>
</tr>
<tr>
<td>latin4</td>
<td>ISO Latin-4 encoding (Scandinavian, Greenland Inuit, Lappish)</td>
</tr>
<tr>
<td>latin5</td>
<td>ISO Latin-5 encoding (Turkish)</td>
</tr>
<tr>
<td>latin9</td>
<td>ISO Latin-9 encoding (with euro symbol)</td>
</tr>
<tr>
<td>decmulti</td>
<td>DEC Multinational encoding</td>
</tr>
<tr>
<td>cp850</td>
<td>IBM 850 code page (Western Europe)</td>
</tr>
<tr>
<td>cp852</td>
<td>IBM 852 code page (Eastern Europe)</td>
</tr>
<tr>
<td>cp437</td>
<td>IBM 437 code page (North America)</td>
</tr>
<tr>
<td>cp437de</td>
<td>Variant on 437, with German ß replacing Greek β in position 225</td>
</tr>
<tr>
<td>cp865</td>
<td>IBM 865 code page (Scandinavia)</td>
</tr>
<tr>
<td>applemac</td>
<td>Macintosh encoding</td>
</tr>
<tr>
<td>next</td>
<td>NeXt encoding</td>
</tr>
<tr>
<td>ansinew</td>
<td>Windows ANSI encoding</td>
</tr>
<tr>
<td>cp1252</td>
<td>Windows 1252 code page (same as ansinew)</td>
</tr>
<tr>
<td>cp1250</td>
<td>Windows 1250 code page, for Central and Eastern Europe</td>
</tr>
</tbody>
</table>

the \inputenc package will be processed correctly on their own system, but will generate garbage on another one. Thus this package should always be added when special symbols are directly typed in, especially if there is any possibility that the file may be processed on another computer system.

D.6 Alternatives for special symbols

A number of symbols in the text fonts cannot be addressed explicitly, but only as ligatures, for example ¿ as ?‘. Other symbols that one might want in text are only available in math mode. \LaTeX provides some \text... commands, listed in Table D.2 on the opposite page, to print these and other characters directly.

D.7 Managing code and documentation

This section describes two additional advanced features for maintaining \LaTeX code and its documentation. There is no need to employ these at all, for any home-written packages and classes will function just as well without them. They do add a degree of sophistication and security, by tracing the history of the code's development, and by including the current documentation within the code itself. Here 'documentation' means both the user's manual and the programmer's description of what each step of
Table D.2: Alternative commands for special symbols in text mode

<table>
<thead>
<tr>
<th>Command</th>
<th>Symbol</th>
<th>Replaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ligatures</td>
<td></td>
<td></td>
</tr>
<tr>
<td>\textemdash</td>
<td>—</td>
<td>---</td>
</tr>
<tr>
<td>\textendash</td>
<td>–</td>
<td>--</td>
</tr>
<tr>
<td>\textexclamdown</td>
<td>!</td>
<td>‘!‘</td>
</tr>
<tr>
<td>\textquestiondown</td>
<td>?</td>
<td>‘?‘</td>
</tr>
<tr>
<td>\textquotedblleft</td>
<td>&quot;</td>
<td>‘“‘</td>
</tr>
<tr>
<td>\textquotedblright</td>
<td>&quot;</td>
<td>‘”‘</td>
</tr>
<tr>
<td>\textquotesingle</td>
<td>‘</td>
<td>‘‘</td>
</tr>
<tr>
<td>\textquote双行</td>
<td>‘</td>
<td>‘‘</td>
</tr>
<tr>
<td>Math symbols</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| \textbullet       | •      | \textbullet$ \bullet$
| \textperiodcentered| .     | \textperiodcentered$ \cdot$
| \textbackslash    | \     | \textbackslash$ \setminus$
| \textbar          | |     | \textbar$ |$
| \textless         | <      | \textless$ <$
| \textgreater      | >      | \textgreater$ >$
| Miscellaneous     |        |          |
| \textvisiblespace | ·      | \verb*+ +
| \textasciicircum  | ^      | \textasciicircum$ ^$
| \textasciitilde   | _      | \textasciitilde$ _$
| \textsuperscript{123} | ^123 | \textsuperscript{123}$ ^{123}$
| Special symbols   |        |          |
| \textcompwordmark | (ligature break) | \textcompwordmark$ \textcompwordmark$
| \textregistered   | R      | \textregistered$ ^R$
| \texttrademark    | ™      | \texttrademark$ ^TM$
| \textcircled{x}   | X      | \textcircled{x}$ X$

The \LaTeX installation files are written in this way. There are about 40 such files that are processed with the DocStrip utility (next Section) to extract the code from these files and merge it into one file latex.ltx. The documentation for each contributory file can be viewed alone, or a giant document can be created for the entire set. On the other hand, the standard class files, with much common coding, along with their option files, are all extracted from the one file classes.dtx, with a single documentation.

More practically, many packages are provided as .dtx files containing the package code plus user manual and programmer's documentation. A DocStrip batch job with extension .ins is also given, which when processed by \LaTeX (or Plain \TeX for that matter) extracts the actual .sty file, along with a drive .drv file. \LaTeXing the driver file generates the documentation. This is usually superfluous since the .dtx file itself is so constructed that
Appendix D. LaTeX Programming

it behaves like a driver file. However, the .drv file can be edited by the user to select, say, just the user manual and to suppress the programmer notes.

The next sections describe first the extraction program DocStrip, which can be used for many other applications as well (e.g. custom-bib, Section 14.3), and then the integrated documentation scheme in the following section.

D.7.1 The DocStrip utility

Since the \TeX{} program is a programming as well as a text formatting language, it can be exploited to provide a number of utility 'programs' to manipulate files. Such programs are immediately portable: if you have \TeX{}, you can run them. One example of such a program is the DocStrip utility for extracting functioning code out of one or more source files containing documentation, as comments. The program was originally written by Frank Mittelbach and further developed by Johannes Braams, Denys Duchier, Marcin Woliński, and Mark Wooding.

The basic idea was to copy one file to another leaving off all the comment lines. From this simple concept came two additional features that extended the application of DocStrip: alternative lines of coding could be selectively suppressed or included depending on options chosen at processing time; and multiple files, or modules, could be input to form a single output file. This means that one source file can be the home of several different \LaTeX{} packages, and that a single package file can be constructed out of many different components, as for the main \LaTeX{} file \texttt{latex.ltx} mentioned above.

Running interactively

The simplest way of running DocStrip on a file is to invoke it with \TeX{} (or \LaTeX{}, but \TeX{} is faster), as

```
tex docstrip
```

which produces the response

```
**********************************************************
* This program converts documented macro-files into fast *
* loadable files by stripping off (nearly) all comments! *
**********************************************************

****************************************************
* First type the extension of your input file(s): *
\infileext=
****************************************************
```
One replies by entering the input extension (usually .dtx); then one is asked in turn for the output extension, the options wanted, and finally, for the root name of the file(s) to be processed. Execution follows.

There are some limitations to this method, since the input and output files have the same root name, differing only in the extension, and initial and final comments added to the output are fixed. The more flexible means of running the utility is with a batch job.

**Running as a batch job**

A DocStrip batch job is a file containing instructions for the utility. For example, suppose the fullpage package illustrated in Section D.3.1 is put into a documented source file called fullpage.dtx, and the actual package file fullpage.sty is to be extracted with the option package, while a documentation driver file fullpage.drv is obtained with option driver; the batch file, named fullpage.ins, could look like

```latex
\input docstrip
\preamble
This is a stripped version of the original file.
\endpreamble

\postamble
This is the end of the stripped file.
\endpostamble
\declarepreamble\predriver
This is a documentation driver file.
\endpreamble
\declarepostamble\postdriver
End of documentation driver file.
\endpostamble
\keepsilent
\askforoverwritefalse
\generate{\file{fullpage.sty}\from{fullpage.dtx}{package}}
  {\file{fullpage.drv}\usepreamble\predriver
  \usepostamble\postdriver
  \from{fullpage.dtx}{driver}}
\endbatchfile
```

The first and last lines are vital: first the file docstrip.tex is loaded, defining all the special DocStrip commands; then after the instructions have been executed, \endbatchfile ensures an orderly termination.
The commands \texttt{\preamble} and \texttt{\postamble} allow one to insert explanatory comments at the beginning and end of the extracted file. The preamble is often a copyright notice and/or a caveat that the extracted file should never be distributed without the source. Additional labeled pre- and postambles may be declared with \texttt{\declarepreamble} and \texttt{\declarepostamble}, and activated with declarations \texttt{\usepreamble} and \texttt{\usepostamble}.

The instructions \texttt{\keepsilent} and \texttt{\askforoverwritefalse} are optional. The former suppresses processing information during the run while the latter turns off the warning that an existing file may be overwritten.

The main command is \texttt{\generate} which specifies the files to be created with a series of \texttt{\file} commands, each taking two arguments: the name of the new file and a list of instructions for its production. These instructions can be declarations like \texttt{\keepsilent} or \texttt{\usepreamble}, but the main one is the command \texttt{\from}. This again takes two arguments: the name of the input file and a list of the options to be applied. In the above example, each generated file has only a single input source, but multiple input files are possible with a series of \texttt{\from} commands.

The created \texttt{fullpage.sty} file now contains

\begin{verbatim}
 This is file `fullpage.sty',
generated with the docstrip utility.
The original source files were:
 fullpage.dtx  (with options: `package')
 This is a stripped version of the original file.
 Copyright (C) 1994 Patrick W. Daly
 \NeedsTeXFormat{LaTeX2e}
 \addtolength{\topmargin}{-1in}
 This is the end of the stripped file.
 End of file `fullpage.sty'.
\end{verbatim}

It is possible to have a master batch job to process individual ones, in which case the master must input them with \texttt{\batchinput}, and not with \texttt{\input}.

Rules for removing lines

The \texttt{\generate} and \texttt{\file} commands cause the input file(s) to be transferred to the output file, line by line, according to the following rules:

1. any lines beginning with a single % sign are removed;
2. any lines beginning with double \%\% signs are retained;

3. any line beginning with \%<opt> (or \%<+opt>) will have the rest of its text transferred if opt is one of the selected options; otherwise it is removed;

4. any line beginning with \%<!opt> (or \%<-opt>) will have the rest of its text transferred if opt is not one of the selected options; otherwise it is removed;

5. all lines between \%<*opt> and \%</opt> are retained or removed depending on whether opt is a selected option or not.

Options can be combined logically, negated, and grouped:

- \texttt{a\&b} (a and b);
- \texttt{a|b} (a or b);
- \texttt{!a} (not a);
- \texttt{(a|b)&c} (c and one of a or b)

For more information on \texttt{DocStrip}, see the documentation which can be obtained by processing \texttt{docstrip.dtx} with \LaTeXX.

### D.7.2 Documenting \LaTeXX coding

The documenting of software products is extremely important: it provides on the one hand a manual for the user, and on the other details about the coding for the programmer. The original \LaTeXX styles, as well as the basic \texttt{latex.tex} file, were heavily commented by Leslie Lamport, but only with straightforward normal text. It was Frank Mittelbach's \texttt{doc} package that first allowed sophisticated, integrated documentation of the source codes.

By \textit{integrated} documentation, we mean that the descriptions and the coding are to be found merged together in a single source file. Thus two processes are necessary, one to extract the actual coding on its own (the \texttt{DocStrip} utility of Section D.7.1) and another to print the documentation (the \texttt{doc} package). The basic idea behind this package is that the comments are in fact regular \LaTeXX text, with some extra features to allow automatic indexing and to record the program's development. The coding itself appears in a special type of \texttt{verbatim} environment.

The documentation is produced by running a special driver file through \LaTeXX. Such a driver for a source file named \texttt{fullpage.dtx} would be, in its simplest form,

\begin{verbatim}
\documentclass{article}
\usepackage{doc}
\begin{document}
\DocInput{fullpage.dtx}
\end{document}
\end{verbatim}
Appendix D. \LaTeX\ Programming

The \texttt{\textbackslash DocInput} command reads in the specified file, but it first alters the function of the \% character from 'comment' to 'do nothing'. This means all comment lines in \texttt{fullpage.dtx} become real text to be processed!

We describe some of the extra features that the \texttt{doc} package makes available for the 'comments' in the .dtx files, illustrated by an example, the source file \texttt{fullpage.dtx} for the \texttt{fullpage} package presented in Section D.3.1. As usual, a more complete manual can be acquired by processing \texttt{doc.dtx}.

**The description part**

The documentation consists of two parts: the \textit{description}, which is a manual for the end user, and the \textit{coding}, a detailed explanation of how the software works, including the lines of code themselves. It is possible to suppress the coding part and to print only the description by issuing

\begin{verbatim}
\OnlyDescription
\end{verbatim}

in the preamble.

Special commands for the description part are:

\begin{verbatim}
\DescribeMacro{\macro_name}
\DescribeEnvironment{\env_name}
\end{verbatim}

which are placed at the start of the text that illustrates a new high-level command (macro) or environment. These commands do two things: they place the macro or environment name in a marginal note at that location (for easy reference when reading) and they insert an entry in the index.

Because documentation needs to use the \texttt{\verb} command frequently for printing input text, some abbreviations are provided with

\begin{verbatim}
\MakeShortVerb{\c} and \DeleteShortVerb{\c}
\end{verbatim}

which first turn the character \texttt{c} into shorthand for \texttt{\verb\c}, and then restore its normal use. For example, after \begin{verbatim}\MakeShortVerb{\|}, \DeleteShortVerb{\|}\end{verbatim} prints \texttt{\mycom}. (These commands can be made available for any document by loading the package \texttt{shortvrb}, which is actually extracted from the \texttt{doc} package.)

If the comment character \% has been deactivated, how can one put comments into the documentation text? One way is to make use of the \TeX\ conditional \texttt{\iffalse} to form a block, or \textit{meta} comment, as

\begin{verbatim}
\iffalse
% These lines are ignored even when the percent character is inactive
\fi
\end{verbatim}

The other method is to use \texttt{^\^A} in place of \%, a special \texttt{doc} feature.

The description part is terminated with
\StopEventually{final text}

where final text is to appear at the very end of the article; if only the
description part is printed, final text is printed immediately and the
documentation is ended.

The coding part

The coding part should normally contain the more specialized material
that is of no interest to the everyday user. The special commands that
may be used here are

\begin{macro}{macro_name} text and code \end{macro}
\begin{environment}{env_name} text and code \end{environment}

both of which again insert a marginal note and make an entry in the index.
They also organize any \changes commands, as explained below.

The most important environment in the coding part is macrocode,
which prints its contents as in verbatim, optionally with a code line
number. The form of this environment is somewhat special:

\begin{macrocode}
lines of code
\end{macrocode}

The four spaces before the \end{macrocode} are obligatory; those before
the \begin{} are not necessary, but it is good practice to insert them for
symmetry. This environment also counts all the backslashes within it for
a checksum test, and makes an index entry for every command name that
it finds. So it is something more than a mere verbatim environment!

The coding part is brought to an end with

\Finale

which carries out the checksum test and prints the stored final text from
\StopEventually. There may actually be more text following it, which
is only printed when both description and coding parts are output.

Index of macros and record of changes

The \doc package makes automatic entries into an index by means of
the two \Describe commands and the two environments presented
above. As well, all commands that appear in the coding are indexed.
However, the indexing is turned on only if one of

\CodelineIndex or \PageIndex

is given in the preamble. The first references the indexed commands to
the number of the code line where they appear, the second to the page
number. In the second case, the code lines are not numbered, unless the declaration `\CodelineNumbered` is also issued.

Since not all commands in the coding really need to be indexed, especially those that are part of standard \LaTeX{} and \TeX{}, the command

\DoNotIndex{list of command names}

is given, often repeatedly, near the beginning, to exempt the listed commands from being indexed.

The automatic indexing of all the commands in the code slows down the processing considerably. Once the index `.idx` file has been produced, future runs do not need to repeat this effort (unless there have been changes to the code). The command

\DisableCrossrefs

will suppress this indexing, but it may be countermanded by an earlier `\EnableCrossrefs` which neutralizes the disabling command.

The text of the index is generated from the `.idx` data file by the `MakeIndex` program (Section 9.4.3), which must be run with the special indexing style `gind.ist` as

```
makeindex -s gind.ist filename
```

The indexing style file `gind.ist` can be extracted from `doc.dtx`.

To print the index, the command

\PrintIndex

is placed where it should appear. Often this is part of the `final text` in `\StopEventually`. An up-to-date index can only appear after `MakeIndex` has been run between two \LaTeX{} processings.

A record of changes to the software can be made by inserting

\changes{version}{date}{text}

throughout the documentation, in both the description and coding parts. To form the change history list, the command

\RecordChanges

must be placed in the preamble. This enables the change entries to be placed in a glossary file, which is then processed by `MakeIndex` as

```
makeindex -s gglo.ist -o filename.gls filename.glo
```

(The indexing style file `gglo.ist` is also extracted from `doc.dtx`.) The change history is then printed in the documentation where

\PrintChanges

is located, again often as part of `final text`. The texts of the `\changes` commands are ordered, first by version number, then by the name of the macro or environment in which they appear.
D.7. Managing code and documentation

Integrity tests

If the source file is to be sent over electronic networks, there is a danger that it might be corrupted or truncated. Two tests are possible to check for this. By placing

\CheckSum{num}

near the start of the documentation (before the coding anyway), all the backslashes in the macrocode environments will be added up, and the total compared with the number \textit{num} by the \texttt{\textbackslash Final\texttt{e}} command. If \textit{num}=0, the true total will be printed on the monitor; otherwise, if the sum does not agree with \textit{num}, an error is printed, with the two values.

The other test checks that the character set has not been corrupted by passing through computer systems with different character codes.

\CharacterTable
{Upper-case \texttt{\A\B\C\D\E\F\G\H} . . .
  . . . . . Tilde \texttt{\textbackslash\textasciitilde}}

The argument must agree exactly with that expected by \texttt{doc} (except for inactive \texttt{\%} signs and multiple spaces). It should be copied from the \texttt{doc.sty} or \texttt{doc.dtx} files.

Obtaining the file information

File information may be obtained with the command

\GetFileInfo{filename}

which defines \texttt{\filename}, \texttt{\filedate}, \texttt{\fileversion}, and \texttt{\fileinfo} from the optional release information to be found in the \texttt{\Providesxxx} command identifying the specified file (Section D.2.1). The idea is that the .dtx file should contain this information once, and only once, but it needs to be known to print it in the title of the article. These \texttt{\filexxx} commands may be used for this purpose. The release information must conform to the sequence \texttt{date}, blank, \texttt{version}, blank, \texttt{text}.

The \texttt{ltxdoc} class

A special class called \texttt{ltxdoc} is provided to assist running the \texttt{doc} package. It invokes the \texttt{article} class with the \texttt{doc} package, and then issues commands

\AtBeginDocument{\MakeShortVerb{\|}}
\CodelineNumbered
\DisableCrossrefs
and defines a number of other useful commands for aiding the documentation. See the description by processing \texttt{\texttt{ltxdoc.dtx}}. It also provides for local configuration: if \texttt{ltxdoc.cfg} exists, it is read in. This can pass paper size or other formatting options to \texttt{article}, \texttt{issue} \texttt{OnlyDescription} by means of \texttt{AtBeginDocument}, and so on.

**A sample .dtx file**

To illustrate these features, we show part of the source file \texttt{fullpage.dtx}. The initial lines ensure that all the files that can be extracted from it (the package \texttt{.sty} and the documentation driver \texttt{.drv}) receive their proper identifying commands. The date and version information appears only once, but is transferred to both extracted files.

\begin{verbatim}
% \iffalse (This is a meta-comment)
%% Copyright (C) 1994 Patrick W. Daly
\NeedsTeXFormat{LaTeX2e}
%<*dtx>
\ProvidesFile {fullpage.dtx}
%</dtx>
%<package>\ProvidesPackage{fullpage}
%<driver>\ProvidesFile{fullpage.drv}
% \fi
%\ProvidesFile{fullpage}  
[2002/02/15 1.1 (PWD)]
\end{verbatim}

It is the last \texttt{\ProvidesFile{fullpage}} that enables the proper functioning of \texttt{\GetFileInfo}; the first one is only a dummy to absorb the information line when the \texttt{.dtx} file is read directly.

Next, the driver part is given. This is what \LaTeX{} sees when it processes the file directly.

\begin{verbatim}
% \iffalse
%<*driver>
\documentclass[a4paper,11pt]{article}
\usepackage{doc}
\EnableCrossrefs
\RecordChanges
\CodelineIndex
\begin{document}
 \DocInput{fullpage.dtx}
\end{document}
%\fi
\end{verbatim}

Now the checksum and list of commands that are not to be indexed are given, followed by the start of the article.

\begin{verbatim}
% \CheckSum{73}
% \DoNotIndex{\addtolength,\boolean,\ExecuteOptions,\ifthenelse}
\end{verbatim}
We advance to the end of the description and start of the coding part.

The end of the file finishes the coding and calls \Finale.

\Finale
Today it is no longer sufficient to produce professional-looking printed output on paper; one has to be able to get it online as well, that is, make it available in electronic form of some kind. To some extent, PostScript output fulfills this requirement in that it may be considered electronic paper. However, as we point out in Section 15.3, true electronic documents are something quite different.

The Internet used to be a nice, quiet neighborhood where academics could exchange simple, plain text emails or obtain known files by FTP, until it was invaded by the rowdy World Wide Web with its corporate identities, slick advertising, energetic yuppies, anarchists, and revolutionaries. In other words, it went cosmopolitan. It is here in this glittering marketplace that electronic documents have to compete for attention.

An electronic document is not something that is simply sent to a printer or leisurely viewed on a monitor: it is interactive, in that it guides the reader to other parts of itself or to other documents located anywhere else on the Web. With these links, the viewer can jump to other relevant passages with a mere mouse click. Colored illustrations are naturally part of such documents, but so are sounds and movies, as well as means of sending feedback to the author. This is a totally new medium for information exchange, more radical than the Gutenberg revolution from handwritten manuscripts to printed books.

In Section 1.2 we explain the structural similarities between \LaTeX{} and the Web languages HTML and XML. Here we point out the possibilities for conversion from \LaTeX{} to the others, and then how \TeX{} can be used as an engine to render XML documents in \LaTeX{} formats. Detailed descriptions of the conversion programs are to be found in the supplied documentation, and in the \LaTeX{} Web Companion (Goossens and Rahtz, 1999). This topic is far too extensive to be dealt with in this book.
Appendix E. \LaTeX and World Wide Web

E.1 Converting to HTML

E.1.1 The \LaTeX2HTML program

The \LaTeX2HTML translator, written by Nikos Drakos with additions by a large number of contributors, is a comprehensive Perl script that converts a \LaTeX source file into HTML with the help of several other programs, notably \LaTeX itself, dvips, Ghostscript, and the netpbm library of graphics utilities.

When \LaTeX2HTML processes a \LaTeX file, it creates a new subdirectory with the same name as that file, to which it writes the resulting HTML output and any generated images as .gif files. The program interprets the \LaTeX input text in the same way as \LaTeX itself does, but instead of producing typesetting instructions in the .dvi file, it writes appropriate HTML code to an .html file. For example, \texttt{\section{Introduction}} is interpreted as \texttt{<H1>Introduction</H1>} for the output.

This means that \LaTeX2HTML is essentially duplicating the \LaTeX processing, an enormous undertaking. Since the document classes contain varying commands or have common ones behaving differently, there must be Perl scripts for each one (\texttt{article.perl}, and so on) to program these commands properly. Additional packages loaded with \texttt{\usepackage} can also define new commands, or alter the functionality of existing ones; the translator must be informed about these by means of corresponding Perl scripts. For example, the \texttt{natbib} package described in Section 9.3.4 defines citation commands \texttt{\citet} and \texttt{\citep}, which need to be made available to \LaTeX2HTML in a file \texttt{natbib.perl}. Most of the tools packages of Section B.5.4 are included as Perl scripts, as are many other popular contributed packages. In other words, the entire \LaTeX2HTML installation must mirror the \LaTeX one.

This process is somewhat simplified by the fact that both formats are markup languages written in pure typewriter text. However, this similarity soon reaches its limits with the many \LaTeX features not available in HTML, such as complex math, included figures, cross-referencing. In the other direction, HTML exhibits hyperlinks, both internally and to external documents, something not provided by normal \LaTeX.

\LaTeX2HTML attempts to reproduce math with the limited HTML possibilities, but failing that, formulas are handled the same as figures and other environments that cannot be directly rendered: they are converted to GIF image files along the route \texttt{.tex → .dvi → .ps → .pbm → .gif}, which are then included in the HTML file as in-line images.

Cross-references and citations are automatically provided with hyperlinks to their targets; the keyword index also links back to the text. Other features of HTML can be included by means of special commands defined in the \texttt{html.sty} package. For example:

- explicit hyperlinks, internal and external
• conditional text for the HTML or \LaTeXX versions only
• raw HTML code
• segmentation of the HTML output
• finer image control
• ... 

This package thus provides the \LaTeX writer with a more convenient interface for producing hypertext documents; furthermore, the same document can still be processed with \LaTeX to generate the 'paper' version at any time. The conditional text mentioned above allows the electronic and paper versions to exhibit some differences.

\LaTeXX2HTML is available on the CTAN servers (Figure B.4 on page 390) under the support directory. It can also be downloaded from the \LaTeXX2HTML home page http://cbl.leeds.ac.uk/nikos/tex2html/doc/latex2html/latex2html.html, where more information as well as a manual can be obtained.

### E.1.2 The \TeXX4ht program

An alternative program is that by Eitan M. Gurari, called \TeXX4ht. It produces much the same results as \LaTeXX2HTML, but by a different route. Rather than trying to interpret the \LaTeX input text itself, it processes instead the .dvi file to extract the HTML output. It is therefore applicable both to \LaTeX and Plain \TeX, as its name implies.

In reality, it is not as simple as that. Since the DVI output is normally only a list of typesetting instructions, containing no logical markup information at all, it is necessary to process the \LaTeX input with the matching \TeXX4ht.sty package that redefines all the regular \LaTeX commands to write appropriate code to the .dvi file. This is accomplished by means of the \special command, a basic \TeX command for writing instructions to a particular DVI driver. In this case the driver is the \TeXX4ht program; no other driver will be able to process this .dvi file.

The \TeXX4ht processor writes the HTML output according to various options specified in the \LaTeX input. For example, the HTML output can be segmented by chapters or sections, tables of contents may be added to each segment, links are automatically established for cross-references. As with \LaTeXX2HTML, manual links to external documents or internal points can be made, raw HTML coding included, explicit images included. It is also possible to turn off the HTML conversion in order to produce a normal .dvi file; conditional input for whether HTML is on or off is also available.

Any symbols, tables, equations, or imported figures that cannot be directly rendered in HTML are written to a new DVI file, with extension
.idv, one item per page. Finally, a log file is written containing conversion instructions for each such object; this file is so constructed as to act as a script, or batch job, for creating GIF images. Just how this conversion is realized and how the instructions are formatted depend on the local installation and can be configured by the user. However, the usual route is something like .idv -> .ps (dvips) -> .ppm (Ghostscript) -> .gif (pbm). An example of \TeX{}X4ht output with a GIF representation of a math equation can be seen in Figure E.1 on page 482.

Since \TeX{}X4ht needs to redefine all the \LaTeX{} markup commands, one could argue that it too is duplicating \LaTeX{} just like \LaTeX{}2HTML. To some extent this is true, although there is a very important difference: it is only the markup, not the formatting commands, that needs to be massaged. Most classes and additional packages will need no extra configuring to be processed by \TeX{}X4ht. A package like \texttt{natbib} which adds new citation commands \texttt{\cite{t} and \citep} will totally confuse \LaTeX{}2HTML without the additional Perl script \texttt{natbib.pl}, whereas under \TeX{}X4ht these commands will produce the correct output text in the HTML file without any additional definitions. (However, the automatic links between the citations and list of references will be missing, since this is a markup feature, but the printed text will be all right.)

\TeX{}X4ht is available from the \TeX{}Live CD, or from \texttt{http://www.cis.ohio-state.edu/~gurari/\TeX{}4ht/mn.html}.

\LaTeX{}2HTML and \TeX{}X4ht are only two examples of procedures for creating HTML output from \LaTeX{}. In both cases, one should not try to convert arbitrary \LaTeX{} files, but should view them as means of producing normal \LaTeX{} and HTML output from a common source file. This source file must be constructed accordingly, possibly with conditional texts for the two outputs, with manual hyperlinks, formatting options for HTML, and so on.

The use of GIF bitmap images for rendering items not readily available in HTML, while producing the visual features on screen, does not allow these objects to be searched for or otherwise further processed. Such images also do not participate in automatic font size changes and other Web manipulations.

\textbf{E.2 The Extensible Markup Language: XML}

It has long been recognized that HTML has many severe limitations. Its simplicity may have been responsible for its rapid rise in popularity, but that simplicity also restricts it to the type of documents for which it was originally intended. Today the Web is offering much more than at the beginning, with databases, search engines, data verification. HTML is not programmable, so that browser suppliers have found it necessary to add additional features of their own, which only function with their browser.
E.2. The Extensible Markup Language: XML

Standardization is thus destroyed. What is needed is a means of adding new features to an HTML file so that the browser can be informed of what they mean. This would be like adding new features to a \LaTeX file by means of a package. It is to this end that the Web Consortium began the development in 1996 of the Extensible Markup Language, or XML.

However, XML is not so much a language as a specification for defining languages in a standard manner. An XML interpreter must be able to determine whether an XML document conforms to the general rules (mainly that all tags are properly closed), in which case it is considered to be well-formed, and to the specific syntax of the tags that it uses (as defined in a DTD, Document Type Definition), in which case it is also valid. There remains the question of how those tags are to be rendered (how are sections formatted, what to do with cross references, how to include graphics), which is relegated to style sheets documented in the Extensible Stylesheet Language, XSL. Here we have true separation between content and format, between logical and typographical markup.

If all this sounds overwhelming, it is! Furthermore, the DTDs are considered to be too cumbersome and various alternatives have been developed, themselves conforming to XML (which the DTDs do not). This whole area is in transition, but the rewards will be high when a reliable standard for electronic data exchange is achieved. Data in this sense is very generic, and includes textual documents. We recommend the books by (Marchal, 2000) and (Williamson, 2001); or one can check the Web Consortium home page http://www.w3.org/.

E.2.1 XML and \LaTeX

What does all this have to do with \LaTeX? Just as the similarity between \LaTeX and HTML allow relative easy conversion, so does the essential similarity with XML. The \TeX4ht system has already done the real work of converting \LaTeX structure into HTML tags; it is therefore simply a matter of changing the output for each of those tags, something that is put into corresponding configuration files.

Since XML itself is not a language with defined tags, how do we know what those tags are to be? Here we rely on existing DTDs for textual documents. These can be the DocBook DTD developed for software user manuals and other computer documentation, or the TEI DTD, the Text Encoding Initiative meant essentially for general text, including images and sounds.

The \TeX4ht collection on the \TeXLive CD includes configurations for producing XML output from \LaTeX for both of these DTDs. The preamble to the \LaTeX source file should then include

\begin{verbatim}
\usepackage[html]{tex4ht}
\end{verbatim}

for HTML output
\usepackage[xhtml,docbook]{tex4ht} for XML, DocBook output
\usepackage[xhtml,tei]{tex4ht} for XML, TEI output

One might ask why does the author not write an XML file directly as the source document? First, one might very well want to convert older documents recorded in \LaTeX{} to XML. Secondly, one might find it simpler to write in \LaTeX{} (not everyone agrees) since no one would write an XML file by hand (everyone agrees) but would rely on some application to do it behind the scenes. Certainly \LaTeX{} source text is easier to read, even when intersperse with commands, than an XML file.

\section*{E.2.2 MathML and \LaTeX{}}

One of the strengths of \TeX{} and thus \LaTeX{} over other text processing systems is the ability to handle mathematics in a way that is acceptable to mathematicians. HTML has never been able to even begin to compete in this area. The only way conversion programs like \LaTeX{}2HTML and \TeX{}4ht can render equations like that in Figure E.1 is to convert it to an image file. This is hardly a satisfactory solution.

The MathML project attempts to define an XML language to encode mathematics. We refer to chapter 8 of the \LaTeX{} Web Companion (Goossens and Rahtz, 1999) for a description. \TeX{}4ht can produce output in MathML by specifying
\begin{verbatim}
\usepackage[xhtml,mathml]{tex4ht}
\end{verbatim}
Symbols that do not exists in the browser fonts will still be converted to image files, but instructions are inserted to carry out the placement for fractions and superscripts and subscripts. Unfortunately, most browsers do not recognize MathML (yet). Again, once this becomes established there will be a standard for recording and exchanging documents with complex mathematics.

\section*{E.2.3 Rendering XML with \TeX{}/\LaTeX{}}

Another role that \LaTeX{}, or more properly the \TeX{} program, can play in the XML world, is to translate it to a form that can be viewed or printed as a finished document. This is known as \textit{rendering}.

David Carlisle has written macros in a file \texttt{xmltex.tex} to act as an XML parser for output in \TeX{} or \LaTeX{}. Sebastian Rahtz has written an additional macro package \textit{Passive\TeX}, to incorporate an XSL style sheet for the TEI DTD into \texttt{xmltex}. With this combination, one can produce DVI or PDF output from a TEI-coded XML file.

There are a number of ways of going about this. The simplest is to write a 'wrapper' file for processing \texttt{document.xml} as
E.3. The techexplorer Hypermedia Browser

\def\xmlfile{document.xml}
\input xmltex.tex

and then to process this with either \LaTeX or pdf\LaTeX. The Passive\TeX macros will be found automatically, if they are on the system.

Another means is to generate an xml\texttt{tex} format, as described in Section B.1.3. This would be done with

\begin{verbatim}
initex &latex xmltex.tex or
tex -ini &latex xmltex.tex
\end{verbatim}

to generate a format named \texttt{xmltex.fmt}. One can then invoke it with

\begin{verbatim}
tex &xmltex document.xml
\end{verbatim}
or simply with \texttt{xmltex document.xml} if an alias or batch file for the command \texttt{xmltex} has been defined to equate to the real command.

Instead of using the \TeX program, one can also make use of pdf\TeX instead, to generate PDF instead of DVI output.

E.3 The techexplorer Hypermedia Browser

A completely different approach to Web viewing of \TeX and L\TeX documents is offered by the IBM techexplorer Hypermedia Browser, a plug-in for Netscape Navigator and Microsoft Internet Explorer under Windows. When properly installed, it is launched automatically within the parent browser window when one opens a file with one of the extensions .tex, .bbl, .ltx, .latex, or .tcx.

The techexplorer interprets the \TeX and L\TeX commands directly and displays the results on the monitor. The original page height and line width are ignored, the paragraphs being fitted to the size of the window as is normal for any browser. Fonts, tables, environments, but most importantly mathematics are reproduced very well. Figure E.1 on the next page demonstrates a single formula represented by both \TeX4ht and techexplorer. Even user commands defined with \texttt{\def} and \texttt{\newcommand} are recognized, provided they occur within the same file. However, no external packages may be loaded, nor are the class files accepted. (Both \texttt{\usepackage} and \texttt{\documentclass} are simply ignored.)

Hypertext features can be included with additional \TeX-like commands that establish internal and external links, pop-up menus, multiple-file documents (segmentation) with automatic links between them, and inclusion of images. In other words, a techexplorer document can have the full range of hypertext features of HTML. Here one must make the same comment as for L\TeX2HTML and \TeX4ht: it is not primarily intended for viewing arbitrary L\TeX documents, but rather for preparing hypertext Web pages by means of \TeX or L\TeX. The only problem here is that most
Appendix E. \LaTeX{} and World Wide Web

Figure E.1: Example of a math formula represented by \TeX{}4ht (as a GIF image of the original \LaTeX{} output) and by \texttt{techexplorer} rendering.

We wish to demonstrate this interesting formula, in two representations:

A) as converted to GIF by \TeX{}4ht:

\[
\Gamma(\theta) = \int_0^\infty \frac{\partial f(\theta, \omega, x)}{\partial x} \, dx
\]

B) as interpreted by \texttt{techexplorer}:

\[
\Gamma(\theta) = \int_0^\infty \frac{\partial f(\theta, \omega, x)}{\partial x} \, dx
\]

One may readily compare the two outputs.

Internet users will probably not have \texttt{techexplorer} installed, so that they will only be presented with the input text, and not its rendering.

One additional application of \texttt{techexplorer} which is not available to the other programs is to embed \TeX{} commands, especially mathematics, inside a regular HTML file. If the commands are stored in a file \texttt{formula.tex}, they could be embedded with

\[
\texttt{<EMBED SRC="formula.tex" WIDTH=300 HEIGHT=70>}
\]

or they can be entered directly as

\[
\texttt{<EMBED TYPE="application/x-techexplorer"}
\texttt{TEXDATA="$$\Gamma(\theta)=\int_0^\infty \frac{\partial f(\theta, \omega, x)}{\partial x} \, dx$$" WIDTH=300 HEIGHT=70>}
\]

The result, shown in the lower part of Figure E.1, is a very acceptable display of the math formula in the middle of an HTML file. (This file was generated by \TeX{}4ht from a \LaTeX{} file, with the above \texttt{techexplorer} text included as raw HTML code.)

The \texttt{techexplorer} is obtainable from \url{http://www.software.ibm.com/enetwork/techexplorer}. It comes in a free introductory version, as well as in a professional version for a nominal charge.
Since 1994, $\LaTeX\ 2\varepsilon$ has replaced $\LaTeX\ 2.09$ as the official version. Although the newer version is backward compatible with the previous one, this is only to allow older source files written for $\LaTeX\ 2.09$ to be processed under $\LaTeX\ 2\varepsilon$; it is not intended that newer files should make use of the obsolete syntax.

Human nature being conservative as it is, many authors continued to use the commands they were familiar with, even today. The purpose of this chapter is to explain those commands for users who may come across such legacy documents, and to indicate to traditional $\LaTeX$ users which commands they should no longer be employing. It is not meant to be a guide for using $\LaTeX\ 2.09$!

At first sight, the differences seem very slight indeed. In fact, the real changes are mainly internal, allowing packages to be handled more systematically and fonts to be dealt with in a more flexible manner. The advantages of $\LaTeX\ 2\varepsilon$ for class and package writers are considerable, as explained in Appendix D. Regular users benefit indirectly by being able to employ these extensions, and by being able to activate fonts other than the original Computer Modern families.

F.1 The 2.09 preamble

As for $\LaTeX\ 2\varepsilon$, the document preamble in $\LaTeX\ 2.09$ contains overall specifications for the entire document, including general layout.

F.1.1 Style instead of class

$\LaTeX\ 2.09$ works with style rather than class files, so the first command in the source file is

$$\documentstyle[options]{style}$$
Appendix F. Obsolete $\LaTeX$

rather than $\documentclass$. Possible values for $style$ are $article$, $report$, $book$, or any other main style files that might exist locally. These styles may have various options associated with them, like $11pt$, $twoside$, and so on, which can be listed in the set of $options$. Not all the options available with the $\LaTeX\ 2\varepsilon$ class files are recognized by the style files.

The $options$ list has an additional function in $\LaTeX\ 2.09$: for any option in the list that is not recognized by the style file, a file with that name and extension $\text{.sty}$ is loaded, if it exists. The original idea of this mechanism was to allow certain options applicable to all styles to have the common coding stored in an extra file, rather than repeating it within each main style file. It was this that later led to the concept of $packages$.

There is no $\usepackage$ command in $\LaTeX\ 2.09$; packages can only be loaded with the $options$ list, and no options may be specified for the packages.

To start a document with the $article$ style and $12pt$ option, with the $parskip$ and $makeidx$ packages, one gives

$\documentstyle[12pt,parskip,makeidx]{article}$

F.1.2 Compatibility mode

$\LaTeX\ 2\varepsilon$ is designed to be able to process older $\LaTeX$ documents with exactly the same output as with $\LaTeX\ 2.09$. To do this, it recognizes the $\documentstyle$ command which switches it into $compatibility$ mode, changing the functionality of many $\LaTeX\ 2\varepsilon$ features. For example, the $\usepackage$ command becomes inoperable, issuing an error message.

With a true $\LaTeX\ 2.09$ installation, the above $\documentstyle$ example would load the files $article.sty$, $parskip.sty$, and $makeidx.sty$. Under $\LaTeX\ 2\varepsilon$, in compatibility mode, it first tries to load the main style as a class file with the extension $\text{.cls}$, in this case $article.cls$, but if that fails, it then looks for the extension $\text{.sty}$. This is to handle local main style files that have no class file equivalent. The packages $parskip.sty$ and $makeidx.sty$ are loaded as well. However, they too may recognize the compatibility mode and behave differently as they would with regular $\LaTeX\ 2\varepsilon$.

F.2 Font selection

The other major difference between the two versions is font selection. The New Font Selection Scheme (NFSS) and font attributes described in Section 4.1.3 are missing in $\LaTeX\ 2.09$, as are the font commands $\textbf$, $\texttt$, and so on, and the font attribute declarations such as $\bfseries$ and $\ttfamily$. (They are still available even in $\LaTeX\ 2\varepsilon$ compatibility mode.)
F.2.1 Old font declarations

In \LaTeX{} 2.09, only the two-letter font declarations that originally came from Plain \TeX{} are possible.

\begin{verbatim}
\rm Roman \it Italic \sc SMALL CAPS \\
\bf Bold face \sl Slanted \sf Sans Serif \\
\tt Typewriter
\end{verbatim}

These are all declarations, changing the font until another font declaration is given, or until the current environment is ended. They are normally used as \texttt{\textbackslash{}bf bold face} to yield \textbf{bold face}.

The emphasizing declaration \texttt{\textbackslash{}em} also belongs to this group, but it has been taken over as part of \LaTeX{}2\epsilon proper.

These old declarations have a different behavior from that of their NFSS counterparts: they rigidly select a particular font instead of altering only one attribute, but retaining the current size. Compare:

\begin{verbatim}
\{\sl slanted \{\bf bold\}\}
\{\slshape slanted \{\bfseries bold\}\}
\end{verbatim}

It is not possible to obtain slanted or italic bold face with \LaTeX{} 2.09 even though such fonts are available.

These two-letter declarations are also retained in the standard \LaTeX{}2\epsilon classes, and not just in compatibility mode. Note carefully that wording: they are not part of \LaTeX{}2\epsilon itself, but are only included as part of the standard class files. This means other class files might not provide them, or that they might even be removed in some later version (although this is most unlikely). Therefore, their use is not to be encouraged at all.

F.2.2 Font size declarations

In \LaTeX{} 2.09, the font size declarations also reset all the other font attributes (Section 4.1.3) to their defaults, that is, Roman, upright, medium weight. By contrast, in \LaTeX{}2\epsilon, these attributes remain unchanged. Compare the results of \texttt{\{\sl slanted \{\Large \texttt{larger}\}\}}:

\begin{verbatim}
\LaTeX{} 2.09: slanted larger
\LaTeX{}2\epsilon: slanted larger
\end{verbatim}

In \LaTeX{}2\epsilon compatibility mode, the size declarations behave as they do in \LaTeX{} 2.09.

F.3 Obsolete means obsolete

Many features have been added to \LaTeX{}2\epsilon, and a number of commands have been given extended syntax, in the form of additional optional arguments. We have no intention of indicating them anymore in this book as
this is a manual for the newer version only. (Previous editions did make a distinction.)

Apart from the restrictions pointed out in the above sections, all of the \LaTeX\ $\varepsilon$ features will work in compatibility mode. Nevertheless, we stress once more, compatibility mode is only intended for the processing of older source files. Every effort has been undertaken to ensure that the results are identical to those of a processing with the true \LaTeX\ 2.09 itself.

Compatibility mode (\texttt{\documentstyle in place of \documentclass}) issues a very strong warning:

\begin{verbatim}
Entering \LaTeX\ 2.09 COMPATIBILITY MODE
*************************************************************************************************
!!WARNING!!  !!WARNING!!  !!WARNING!!  !!WARNING!!

This mode attempts to provide an emulation of the \LaTeX\ 2.09 author environment so that OLD documents can be successfully processed. It should NOT be used for NEW documents!

New documents should use Standard \LaTeX\ conventions and start with the \texttt{\documentclass} command.

Compatibility mode is UNLIKELY TO WORK with \LaTeX\ 2.09 style files that change any internal macros, especially not with those that change the FONT SELECTION or OUTPUT ROUTINES.

Therefore such style files MUST BE UPDATED to use Current Standard \LaTeX: \LaTeX\ 2e.

If you suspect that you may be using such a style file, which is probably very, very old by now, then you should attempt to get it updated by sending a copy of this error message to the author of that file.

*************************************************************************************************

Take this message seriously, and stick to \texttt{\documentclass}.\end{verbatim}
TEX Fonts

Computers work exclusively with numbers, or more precisely, only with bits which may be interpreted as numbers. They do not know the difference between the letter A and an apple, or that either even exists. For text processing, all symbols, both input and output, need to be represented as numbers somehow. The association between number and symbol is called the encoding or layout. The latter term derives from the common method of illustrating the encoding in the form of a table.

Encoding tables are by no means standard. The standard ASCII scheme is just one of several, and it is limited to 7 bits, or 128 characters. There are 8-bit (256 characters) versions as well, in fact, a large number exist, tailored to different computer systems and languages. The question of coding the input for \LaTeX documents with more than 7 bits are addressed in Sections 2.5.9 and D.5.

In this appendix, we look at how the underlying \TeX program deals with output fonts, their nomenclature, and their encoding tables.

G.1 Font metrics and bitmaps

When \TeX decides to output character \textit{nn} in a particular font, all it needs to know is how much room to leave for it. \TeX does not care what the character looks like, for that is the task of the DVI driver afterwards.

Information about the characters in each font are stored in various files, all bearing the root name of the font but with different extensions. This information is divided into files that contain only the sizes of each symbol, and those with the actual drawing or image data.

\texttt{.tfm} \TeX font metric files are the only font files read in by \TeX itself. They contain the sizes of the characters, such as width, height, and depth. For slanted fonts, they also possess the 'italic correction' for each letter. Furthermore, they specify for which letter combinations a different spacing is required, such as AV instead of AV,
or for which a ligature is available. Finally, the .tfm files provide information about the slope of the characters (zero for an unslanted font), the standard word spacing and its stretch and shrinkage, the width of the \texttt{em} and \texttt{quad} spacings, and the spacings at the end of a sentence. Mathematical and symbol fonts require even more information which is also included in their .tfm files.

\section*{.pk}
Compressed pixel files contain bitmaps (images) of the symbols in each font, in one size and one resolution. The DVI driver program uses these to send the output to a printer or previewer. These files are normally generated automatically by the \texttt{METAfont} program (Section G.3) as needed.

Since these files might exist several times for a given font, in different sizes or resolutions, many installations distinguish them by adding the resolution to the extension, as \texttt{cmr10.300pk} or \texttt{cmr10.600pk} for 300 and 600 dpi versions, respectively. On systems that cannot support extensions with more than three characters, both files are named \texttt{cmr10.pk} but are stored in directories \texttt{dpi300} and \texttt{dpi600}.

Font magnification is achieved by selecting an appropriate resolution. If \texttt{cmr10} is to be printed on a 300 dpi printer at double its normal size, the 600 dpi version is used instead.

\section*{.mf}
\texttt{METAfont} source files contain the drawing instructions for each of the fonts. These are not read by the DVI driver but rather by the \texttt{METAfont} program which converts them into .pk bitmap files at the required size and resolution.

\section*{.vf}
Virtual font files are an alternative to .pk files. Instead of having bitmaps for each character, they contain instructions that refer to characters in different, real fonts, or that tell the driver to draw a black box and issue a warning message (that the character does not exist, for example).

Virtual fonts are normally associated with PostScript fonts and are therefore described in Section G.5. However, there is no reason for them not to be used with bitmap fonts as well.

\section*{G.2 Computer Modern fonts}
When Donald E. Knuth invented the \TeX{} program, he also provided it with an extensive set of character fonts, which he named \textit{Computer Modern}, rather than relying on the fonts available on any given printer. At that time, the printer fonts were not so good, and certainly were not uniform.
With the supplied fonts, \TeX{} could produce identical, high-quality results on all printers.

\LaTeX{}, of course, has inherited these fonts, so that they have almost become a trademark for documents produced by \TeX{} or \LaTeX{}. This is not really necessary, for \LaTeX{} need not be married to any particular set of fonts, especially with the New Font Selection Scheme (Appendix A) which simplifies font installation enormously. The main fonts used in this book, for example, are Lucida Bright, Lucida Sans, and Lucida Sans Typewriter, designed by Bigelow & Holmes and distributed by Y&Y Inc.

G.2.1 Font families

Typography is the study and classification of typefaces, something that goes back to Gutenberg's invention of movable type (not of the printing press, which was invented by the Chinese) five and a half centuries ago. Since that time, many families of fonts have been created, bearing classical names like Baskerville, Garamond, Univers, etc. Each member of such a family has the same overall design, or basic look, but vary by being slanted, italic, bold, or thin; and of course, they come in different sizes.

Font families are classified according to certain criteria that often determine to what use they will be put.

**Serif fonts:** are those that have little horizontal lines, or *serifs*, at their edges, to guide the eye better. Experience has shown these to be best for general reading, and so they are regularly used for the main body of text. The NFSS terminology refers to these as *Roman* fonts.

**Sans serif fonts:** are those that are lacking any serifs. Such fonts with their starker appearance are often employed for titling or headlining. Compare sans serif with regular text.

**Fixed fonts:** are those with a uniform letter width, something that has evolved from the typewriter and has been carried over to computer listings. Classically, such fonts have no business in book printing, where *proportional* fonts (the letter *i* is narrower than *m*) have always dominated.

**Decorative fonts:** are ones that stand out because of some unusual characteristic. They are intended to catch attention and to attract the eye. Such families are not complete, with a full range of shapes and widths, and are employed mostly by advertising.

**Mathematical fonts:** are collections of special symbols needed for mathematical works. Their further classification cannot be compared to that of text fonts at all.
A book designer must decide what kinds of typeface families are needed. He or she might decide on a Roman (serifed) font for the main body, a sans serif one for headlines, and then, if it is a book containing computer code, select a fixed font for setting those parts. Finally, symbol fonts will be needed if the work contains mathematical sections.

The Computer Modern set of fonts provides all these classes of families. However, since they were produced before the NFSS attribute system of classification was established, the CM font nomenclature does not conform perfectly with this scheme. The NFSS system frees the user from having to think about the CM font names, for specifying the attributes is sufficient. Of course, the NFSS installation must provide the font definition .fd files which translate any set of attributes into a real font name, or into some acceptable substitute.

G.2.2 Classification of CM fonts

All the \TeX font names begin with the letters `cm', which stands for ‘Computer Modern’, followed by one to four letters describing the style of font, and finally one or two digits specifying the design size in points. This is the root name of the font which must be given in the \texttt{\newfont} command (Section 4.1.5) if it is to be activated directly.

The CM fonts can be classified as text, math, decorative, or other special symbol fonts, as described in the rest of this section. Each type of font has an encoding scheme shown in the accompanying layout tables.

Text fonts

The Computer Modern fonts appropriate for straightforward text can be classified into the three families Roman, sans serif, and typewriter. Within each of these families there are upright, slanted, italic, small caps, and bold variants.

Table G.1 presents the names of these fonts together with the NFSS family/series/shape assignments from Appendix A. They all have the same coding (OT1). The * in each name is the design size specification which takes on values of 5, 6, 7, 8, 9, 10, 12, 17. Only cmr* is available in all these sizes, while some are only to be found in size 10. (This table contains much the same information as in Table A.3 on page 370 but in a different form.)

One sees several apparent inconsistencies in these font names, such as cmti* for the italic upright font. Why is this not cmri*? The answer is that this is a text \textit{} font, as opposed to the \texttt{math italic} font cmmi* described below. These inconsistencies have arisen because the CM fonts were not created with the NFSS classification system in mind, since that was established a decade later.
Further inconsistencies exist in the encoding schemes for the CM text fonts. They are all nominally OT1, but there are slight deviations among them. The proper OT1 encoding is displayed in Layout 1 on the next page for font cmr10. The text italic fonts are identical to the upright ones, except that the dollar sign in position 36 is replaced by the pound sign (Layout 3 for cmti10). The slanted fonts, on the other hand, are exactly the same as the upright ones. Other deviations are to be found in the small caps (Layout 2) and all the typewriter fonts (Layout 4).

Math fonts

The Computer Modern fonts provide three types of mathematical fonts, each with its own encoding scheme.

The mathematical italic fonts cmmi* contain Latin and Greek letters, in upper and lower case, plus a number of extra symbols. Since variable names in formulas are set in italics, these are basically italic fonts. The encoding scheme is designated OML, for old math letters, and is displayed in Layout 5. It is available in design sizes 5–12 pt in normal weight, but only in 10 pt size in bold face, cmmib10. These fonts are used for the letters math alphabet of Section A.3.4.

Symbol fonts cmsy* provide the rest of the math symbols for formulas, except for those that appear in variable sizes. The encoding scheme is named OMS for old math symbols and is shown in Layout 6. It comes in sizes 5–10 pt in normal weight, and in 10 pt size in bold face, cmbsy10. The symbols math alphabet uses these fonts.

Variable sized symbols are to be found in the font cmex10, with encoding OMX, for old math extension. Layout 7 illustrates this set. There is no bold version of this font. It belongs to the largesymbols math alphabet of Section A.3.4.

### Table G.1: Computer Modern text fonts

<table>
<thead>
<tr>
<th>Style</th>
<th>Series/shape</th>
<th>Roman</th>
<th>Sans serif</th>
<th>Typewriter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>cmr</td>
<td>cmss</td>
<td>cmtt</td>
</tr>
<tr>
<td>Upright</td>
<td>m/n</td>
<td>cmr*</td>
<td>cmss*</td>
<td>cmtt*</td>
</tr>
<tr>
<td>&quot; slanted</td>
<td>m/sl</td>
<td>cmssl*</td>
<td>cmssslit*</td>
<td>cmssltt*</td>
</tr>
<tr>
<td>&quot; italic</td>
<td>m/it</td>
<td>cmti*</td>
<td>cmtis*</td>
<td>cmtt*</td>
</tr>
<tr>
<td>&quot; small caps</td>
<td>m/sc</td>
<td>cmcsc*</td>
<td>—</td>
<td>cmcsc*</td>
</tr>
<tr>
<td>Bold</td>
<td>b/m</td>
<td>cmb*</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Bold extended</td>
<td>bx/m</td>
<td>cmsgbx*</td>
<td>cmssbx*</td>
<td>—</td>
</tr>
<tr>
<td>&quot; &quot; slanted</td>
<td>bx/sl</td>
<td>cmbxsl*</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>&quot; &quot; italic</td>
<td>bx/it</td>
<td>cmbxiti*</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>
Appendix G. \TeX\ Fonts

Font Layout 1: The character font \texttt{cmr10}. This is the standard character assignment for the \texttt{OT1} encoding scheme.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>'00x</td>
<td>(\Gamma)</td>
<td>0</td>
<td>(\Delta)</td>
<td>1</td>
<td>(\Theta)</td>
<td>2</td>
<td>(\Lambda)</td>
</tr>
<tr>
<td>'01x</td>
<td>(\Phi)</td>
<td>8</td>
<td>(\Psi)</td>
<td>9</td>
<td>(\Omega)</td>
<td>10</td>
<td>(\digamma)</td>
</tr>
<tr>
<td>'02x</td>
<td>1</td>
<td>16</td>
<td>J</td>
<td>17</td>
<td>`</td>
<td>18</td>
<td>`</td>
</tr>
<tr>
<td>'03x</td>
<td>', 24</td>
<td>B</td>
<td>25</td>
<td>(\alpha)</td>
<td>26</td>
<td>(\alpha)</td>
<td>27</td>
</tr>
<tr>
<td>'04x</td>
<td>`</td>
<td>32</td>
<td>`</td>
<td>33</td>
<td>`</td>
<td>34</td>
<td>`</td>
</tr>
<tr>
<td>'05x</td>
<td>( 40 )</td>
<td>41</td>
<td>*</td>
<td>42</td>
<td>+</td>
<td>43</td>
<td>,</td>
</tr>
<tr>
<td>'06x</td>
<td>0</td>
<td>48</td>
<td>1</td>
<td>49</td>
<td>2</td>
<td>50</td>
<td>3</td>
</tr>
<tr>
<td>'07x</td>
<td>8</td>
<td>56</td>
<td>9</td>
<td>57</td>
<td>:</td>
<td>58</td>
<td>:</td>
</tr>
<tr>
<td>'10x</td>
<td>@</td>
<td>64</td>
<td>A</td>
<td>65</td>
<td>B</td>
<td>66</td>
<td>C</td>
</tr>
<tr>
<td>'11x</td>
<td>H</td>
<td>72</td>
<td>I</td>
<td>73</td>
<td>J</td>
<td>74</td>
<td>K</td>
</tr>
<tr>
<td>'12x</td>
<td>P</td>
<td>80</td>
<td>Q</td>
<td>81</td>
<td>R</td>
<td>82</td>
<td>S</td>
</tr>
<tr>
<td>'13x</td>
<td>X</td>
<td>88</td>
<td>Y</td>
<td>89</td>
<td>Z</td>
<td>90</td>
<td>[</td>
</tr>
<tr>
<td>'14x</td>
<td>`</td>
<td>96</td>
<td>a</td>
<td>97</td>
<td>b</td>
<td>98</td>
<td>c</td>
</tr>
<tr>
<td>'15x</td>
<td>h</td>
<td>104</td>
<td>i</td>
<td>105</td>
<td>j</td>
<td>106</td>
<td>k</td>
</tr>
<tr>
<td>'16x</td>
<td>p</td>
<td>112</td>
<td>q</td>
<td>113</td>
<td>r</td>
<td>114</td>
<td>s</td>
</tr>
<tr>
<td>'17x</td>
<td>X</td>
<td>120</td>
<td>Y</td>
<td>121</td>
<td>Z</td>
<td>122</td>
<td>~</td>
</tr>
</tbody>
</table>

Font Layout 2: The character font \texttt{cmcsc10}. The differences from Layout 1 are symbols 11–15, 25, 60, and 62. The ligatures that normally appear in 11–15 have been replaced by some extra symbols.
### Font Layout 3

The character font `cmti10`. The only difference from Layout 1 is symbol 36 (£ instead of $). All other text italic fonts have this same pattern.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>'00x</td>
<td>Γ 0</td>
<td>Δ 1</td>
<td>Θ 2</td>
<td>Λ 3</td>
<td>Ξ 4</td>
<td>Π 5</td>
<td>Σ 6</td>
</tr>
<tr>
<td>'01x</td>
<td>Φ 8</td>
<td>Ψ 9</td>
<td>Ω 10</td>
<td>Ψ 11</td>
<td>Ω 12</td>
<td>Ψ 13</td>
<td>Ω 14</td>
</tr>
<tr>
<td>'02x</td>
<td>′ 16</td>
<td>† 17</td>
<td>′ 18</td>
<td>′ 19</td>
<td>′ 20</td>
<td>′ 21</td>
<td>′ 22</td>
</tr>
<tr>
<td>'03x</td>
<td>′ 24</td>
<td>′ 25</td>
<td>′ 26</td>
<td>′ 27</td>
<td>′ 28</td>
<td>′ 29</td>
<td>′ 30</td>
</tr>
<tr>
<td>'04x</td>
<td>′ 32</td>
<td>′ 33</td>
<td>′ 34</td>
<td>′ 35</td>
<td>′ 36</td>
<td>′ 37</td>
<td>′ 38</td>
</tr>
<tr>
<td>'05x</td>
<td>(40)</td>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
</tr>
<tr>
<td>'06x</td>
<td>0 48</td>
<td>1 49</td>
<td>2 50</td>
<td>3 51</td>
<td>4 52</td>
<td>5 53</td>
<td>6 54</td>
</tr>
<tr>
<td>'07x</td>
<td>8 56</td>
<td>9 57</td>
<td>: 58</td>
<td>: 59</td>
<td>i 60</td>
<td>- 61</td>
<td>: 62</td>
</tr>
<tr>
<td>'10x</td>
<td>@ 64</td>
<td>A 65</td>
<td>B 66</td>
<td>C 67</td>
<td>D 68</td>
<td>E 69</td>
<td>F 70</td>
</tr>
<tr>
<td>'11x</td>
<td>H 72</td>
<td>I 73</td>
<td>J 74</td>
<td>K 75</td>
<td>L 76</td>
<td>M 77</td>
<td>N 78</td>
</tr>
<tr>
<td>'12x</td>
<td>P 80</td>
<td>Q 81</td>
<td>R 82</td>
<td>S 83</td>
<td>T 84</td>
<td>U 85</td>
<td>V 86</td>
</tr>
<tr>
<td>'13x</td>
<td>X 88</td>
<td>Y 89</td>
<td>Z 90</td>
<td>/ 91</td>
<td>&quot; 92</td>
<td>/ 93</td>
<td>&quot; 94</td>
</tr>
<tr>
<td>'14x</td>
<td>96</td>
<td>a 97</td>
<td>b 98</td>
<td>c 99</td>
<td>d 100</td>
<td>e 101</td>
<td>f 102</td>
</tr>
<tr>
<td>'15x</td>
<td>h 104</td>
<td>i 105</td>
<td>j 106</td>
<td>k 107</td>
<td>l 108</td>
<td>m 109</td>
<td>n 110</td>
</tr>
<tr>
<td>'16x</td>
<td>p 112</td>
<td>q 113</td>
<td>r 114</td>
<td>s 115</td>
<td>t 116</td>
<td>u 117</td>
<td>v 118</td>
</tr>
<tr>
<td>'17x</td>
<td>x 120</td>
<td>y 121</td>
<td>z 122</td>
<td>- 123</td>
<td>- 124</td>
<td>&quot; 125</td>
<td>&quot; 126</td>
</tr>
</tbody>
</table>

### Font Layout 4

The character font `cmtt10`. All tt fonts are set up with the same pattern. The differences from Layout 1 lie in symbols 11–15, 60, 62, 92, 123, 124, and 125. Furthermore, the italic typewriter fonts also have £ in place of $.
Appendix G. TeX Fonts

The term "old" in these encoding names refers to the original encoding as introduced by Donald Knuth, in the same way that the text encoding OT1 indicated the old, or original, text encoding with 128 characters per font. The new encoding schemes contain 256 characters. For text, the encoding is named T1 (Section G.4.3); such extended math fonts do not yet exist.

The extra $\mathcal{AMS}$ fonts in Section 12.4.1 add additional bold face sizes for cmmbi* and cmbsy* as well as some more sizes for cmex*.

Decorative fonts

Three families of decorative or special fonts are available in a single design size and limited attributes. They all exhibit OT1 encoding.

cmfr *Funny Roman* family, consists of two fonts, cmff10 leaning to the left and cmfi10 which leans to the right. Both are series $m$ with shapes $n$ and $\it$ respectively.

cmfb Fibbonacci family, contains one font cmfib8 derived from the Fibonacci series of numbers. It has series $m$ and shape $n$.

cmdunh Dunhill family, with one font cmdunh10 with series $m$ and shape $n$. This font is demonstrated in the sample letterhead on page 364.

Logo fonts

The fonts logo8, logo9, logo10, logosl10, and logobf10 contain only the nine letters A, E, F, M, N, O, P, S, and T, for generating the logos

```
\texttt{METAFONT METAPOST METAPOST METAPOST METAPOST}
```

The LaTeX lazy fonts

As an extension of the cmbsy* fonts for mathematical symbols, LaTeX provides some additional symbols with the font lazy* in design sizes 5–10 pt. It contains the 15 symbols: $\langle \bowtie \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown \blacktriangleleft \blacktriangleleft \blacktriangleright \blacktriangleright \blacktriangledown.
### Font Layout 5

This corresponds to the OML encoding scheme, as does cmmib10. It contains lower case Greek letters in positions 11–39 and math symbols in 40–47, 60–64, and 123–127.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>'00x</td>
<td>⊏ 0</td>
<td>∗ 1</td>
<td>× 2</td>
<td>* 3</td>
<td>÷ 4</td>
<td>◇ 5</td>
<td>± 6</td>
</tr>
<tr>
<td>'01x</td>
<td>⊕ 8</td>
<td>⊗ 9</td>
<td>⊗ 10</td>
<td>⊗ 11</td>
<td>⊗ 12</td>
<td>⊗ 13</td>
<td>⊗ 14</td>
</tr>
<tr>
<td>'02x</td>
<td>⊲ 16</td>
<td>⊳ 17</td>
<td>⊺ 18</td>
<td>⊹ 19</td>
<td>⊺ 20</td>
<td>⊹ 21</td>
<td>⊺ 22</td>
</tr>
<tr>
<td>'03x</td>
<td>~ 24</td>
<td>~ 25</td>
<td>⊂ 26</td>
<td>⊃ 27</td>
<td>≺ 28</td>
<td>≻ 29</td>
<td>≺ 30</td>
</tr>
<tr>
<td>'04x</td>
<td>← 32</td>
<td>→ 33</td>
<td>↑ 34</td>
<td>↓ 35</td>
<td>↔ 36</td>
<td>↘ 37</td>
<td>↗ 38</td>
</tr>
<tr>
<td>'05x</td>
<td>≦ 40</td>
<td>≧ 41</td>
<td>↑ 42</td>
<td>↓ 43</td>
<td>↔ 44</td>
<td>↓ 45</td>
<td>↑ 46</td>
</tr>
<tr>
<td>'06x</td>
<td>⊥ 48</td>
<td>⊥ 49</td>
<td>∈ 50</td>
<td>∈ 51</td>
<td>Δ 52</td>
<td>Δ 53</td>
<td>/ 54</td>
</tr>
<tr>
<td>'07x</td>
<td>∀ 56</td>
<td>∃ 57</td>
<td>∀ 58</td>
<td>∀ 59</td>
<td>⊃ 60</td>
<td>⊃ 61</td>
<td>⊃ 62</td>
</tr>
<tr>
<td>'10x</td>
<td>⊗ 64</td>
<td>⊗ 65</td>
<td>⊗ 66</td>
<td>⊗ 67</td>
<td>⊗ 68</td>
<td>⊗ 69</td>
<td>⊗ 70</td>
</tr>
<tr>
<td>'11x</td>
<td>⊗ 72</td>
<td>⊗ 73</td>
<td>⊗ 74</td>
<td>⊗ 75</td>
<td>⊗ 76</td>
<td>⊗ 77</td>
<td>⊗ 78</td>
</tr>
<tr>
<td>'12x</td>
<td>⊗ 80</td>
<td>⊗ 81</td>
<td>⊗ 82</td>
<td>⊗ 83</td>
<td>⊗ 84</td>
<td>⊗ 85</td>
<td>⊗ 86</td>
</tr>
<tr>
<td>'13x</td>
<td>⊗ 88</td>
<td>⊗ 89</td>
<td>⊗ 90</td>
<td>⊗ 91</td>
<td>⊗ 92</td>
<td>⊗ 93</td>
<td>⊗ 94</td>
</tr>
<tr>
<td>'14x</td>
<td>⊗ 96</td>
<td>⊗ 97</td>
<td>⊗ 98</td>
<td>⊗ 99</td>
<td>⊗ 100</td>
<td>⊗ 101</td>
<td>⊗ 102</td>
</tr>
<tr>
<td>'15x</td>
<td>⊗ 104</td>
<td>⊗ 105</td>
<td>⊗ 106</td>
<td>⊗ 107</td>
<td>⊗ 108</td>
<td>⊗ 109</td>
<td>⊗ 110</td>
</tr>
<tr>
<td>'16x</td>
<td>⊗ 112</td>
<td>⊗ 113</td>
<td>⊗ 114</td>
<td>⊗ 115</td>
<td>⊗ 116</td>
<td>⊗ 117</td>
<td>⊗ 118</td>
</tr>
<tr>
<td>'17x</td>
<td>⊗ 120</td>
<td>⊗ 121</td>
<td>⊗ 122</td>
<td>⊗ 123</td>
<td>⊗ 124</td>
<td>⊗ 125</td>
<td>⊗ 126</td>
</tr>
</tbody>
</table>

### Font Layout 6

The font cmsy10. The cmsy fonts, adhering to the OMS encoding scheme, contain many math symbols as well as the calligraphic letters $\mathcal{A} \ldots \mathcal{Z}$ in positions 65–90. The bold face version cmbsy10 is set up on the same pattern.
Font Layout 7: The font cmex10, containing the mathematical symbols that appear in varying sizes. This is the OMX encoding scheme.
G.3. The METAFONT program

METAFONT is a program for designing and developing character fonts, written by Donald E. Knuth as a companion to his \TeX program (Knuth, 1986c, 1986d, 1986e). It is this program that made the uniform high quality typographic output possible for \TeX at a time when most printers had their own built-in fonts of varying standards.
The printer driver programs have the task of converting the DVI output from \TeX into instruction code for a particular printer. To do this, they need to know how to print each of the symbols used in the document. These symbols are represented as a set of black and white dots (pixels) that is adjusted to the output resolution. Clearly to produce the letter \(A\) in a given size at 600 dpi (dots per inch) requires many more dots than at 150 dpi (16 times). This information is contained in the corresponding .pk files, in compressed format.

However, designing fonts as a set of dots, and that for any number of resolutions, and taking into account idiosyncrasies of individual printers, is an impossible task. Rather, the fonts are defined with .mf files, containing instructions on how to draw the symbols with a pen of a given shape. This is the most general definition, describing the ideal form. METAfont then translates this ideal into the practical realization, the pixels, for any specified resolution. Aspects of the various printers, such as relative pixel size and shape, may also be included, so that the resulting patterns are both resolution and printer dependent. The font metric .tfm files are also produced in this process.

Today it is hardly necessary for most users to know anything more about METAfont other than that it exists. The drivers and previewers are now so constructed that when they notice that the required .pk file is missing, they invoke METAfont themselves to generate it. In this way the collection of pixel files grows as needed and includes only those that are actually used. (Originally huge sets of pixel files had to be supplied to cover every remote possibility.)

Another reason for the decline in the awareness of METAfont is the increasing use of the type 1 fonts for PostScript and PDF output (Chapter 10).

Donald Knuth has announced that he wishes to withdraw from any further development of either of these programs, being prepared merely to correct any definite errors they may contain. To emphasize this decision, he will from now on give version numbers to \TeX that converge to \(\pi\) (3.14159...) and to METAfont that approach the value of e (2.71828...). At present, \TeX is at version 3.14159 and METAfont at 2.718. As a consequence of this decision, any further major developments to these two programs, such as undertaken by user groups, will be under new names, since Knuth has kept the copyright to the existing names.

### G.4 Extended Computer fonts

The Computer Modern fonts were developed in the early days of \TeX when it could only handle 128 characters per font. The modern \TeX program can deal with 256 characters in each font, requiring a new set of standard fonts to exploit this feature.
G.4.1 Limitations and deficiencies of the CM fonts

Most of the diacritical marks (accents) used in European languages written with the Latin alphabet are contained in, or may be generated by, Ti\TeX’s Computer Modern (CM) fonts. A basic set of naked accents is available for combination with other letters, such as the acute accent ‘ with the letter e to make é. Other combinations may be constructed for diacritical marks that are not predefined in Ti\TeX or LATEX.

Fashioning diacritical characters as a combination of letters and special symbols has one great disadvantage for the Ti\TeX processing: words containing such characters cannot take part in the automatic word division since the hyphenation patterns include only pure letters. The accented letters, such as those in German and French and most other languages, must be treated as single characters in the hyphenation patterns, and must appear as single letters in the character set.

In addition to diacritical characters, a number of special letters are employed in some European languages, such as ß, Æ, æ, Œ, œ, Ø, and ø, which are provided in standard Ti\TeX with the CM fonts (Section 2.5.6). However, other special letters, such as Đ, ɲ, Ƿ, þ, and δ, are missing completely and cannot be easily constructed from existing ones.

G.4.2 The Cork proposal

At the 1990 International Ti\TeX Conference in Cork, Ireland, an extension of the Latin alphabet and its assignments within the 256 character positions was proposed and accepted. This extension includes the majority of special and diacritical letters as single characters for many languages written with the Latin alphabet. Hyphenation patterns for such languages may include the special and diacritical letters as single letters for optimal word division by Ti\TeX and LATEX.

Character fonts conforming to the Cork scheme are to bear the identifying letters ec in their names for ‘Extended Computer’ in place of the cm for ‘Computer Modern’.

G.4.3 The realization of EC fonts

The Cork proposal for extending the Ti\TeX fonts to 256 characters was first implemented by Norbert Schwarz, who produced an initial set of META\FONT source files. He also selected the designation dc to emphasize that this was a preliminary realization of the EC fonts. Some work was still needed to fine-tune the design of several symbols.

After issuing versions 1.2 and 1.3 of the DC fonts in 1995 and 1996, Jörg Knappen released the first set of true EC fonts in January, 1997. Font Layout 9 presents his font ecrm1000, the extended version of cmr10. The EC fonts are now considered to be stable in that neither their encoding
| Font Layout 9: The extended font ecrm1000 with T1 encoding. |
nor their metrics (the .tfm files) will be changed in future. Thus their behavior as far as \TeX{} and \LaTeX{} are concerned is finalized. The actual printed characters might be modified slightly in later updates.

The EC font names are of the form \texttt{ecxxn00n}, where \texttt{xx} represents two letters specifying the font characteristics, and \texttt{n00n} the design size in hundredths of points. Thus \texttt{ecrm1000} is the upright Roman font in size 10 points.

The META\TeX{} source files are available for the following extended fonts (without the size specification):

\begin{verbatim}
ecrm  ecrb  eccc  ecci  ecvi  ecss  ecdh  
ecbx  ecti  ecxc  ectx  ecvi  ecsi  
ecbl  ecui  ecsc  ecit  ectx  
ecrb  ecbi  ecoc  ecvt  ecst  ecos  
\end{verbatim}

Comparing these with the root names of the CM fonts from Table G.1 on page 491, one may easily recognize the correspondence. For example, \texttt{ecbx*} is the extended bold font corresponding to the CM font \texttt{cmbx*}.

It is intended that the EC fonts should exist in most design sizes. The present distribution contains almost all the fonts in sizes from 5 to 35.83 pt, that is, with size specifications:

\begin{verbatim}
  0500  0600  0700  0800  0900  1000  1095  
  1200  1440  1728  2074  2488  2986  3583  
\end{verbatim}

In contrast to the CM text fonts which exhibit differences in the symbol assignments among them, as illustrated in Layouts 1–4, the EC fonts all have exactly the same encoding scheme, as presented in Layout 9.

The EC fonts are now part of the standard \LaTeX{} installations.

A parallel set of fonts called \textit{text companion}, or TC, fonts is also provided. These contain special symbols for text that are normally found in the CM math fonts, if at all, such as currency symbols and degree signs. These fonts are still somewhat experimental so that the symbol assignments are not yet stable. The current contents are shown in Layout 10.

\section*{G.4.4 Invoking the EC and TC fonts}

The EC fonts correspond to the NFSS encoding scheme T1 (Section A.1). The simplest way to activate them is to place

\begin{verbatim}
\usepackage[T1]{fontenc}
\end{verbatim}

in the preamble of the document. All this really does is to make T1 the standard encoding by redefining \texttt{\encodingdefault} to be T1, and it loads the file \texttt{t1enc.def} which redefines the accent and special letter commands by means of the encoding commands of Section A.3.7.
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>'00x</td>
<td>←</td>
<td>0</td>
<td>←</td>
<td>1</td>
<td>←</td>
<td>2</td>
<td>←</td>
<td>3</td>
</tr>
<tr>
<td>'01x</td>
<td>←</td>
<td>8</td>
<td>←</td>
<td>9</td>
<td>←</td>
<td>10</td>
<td>←</td>
<td>11</td>
</tr>
<tr>
<td>'02x</td>
<td>←</td>
<td>16</td>
<td>←</td>
<td>17</td>
<td>←</td>
<td>18</td>
<td>←</td>
<td>19</td>
</tr>
<tr>
<td>'03x</td>
<td>←</td>
<td>24</td>
<td>←</td>
<td>25</td>
<td>←</td>
<td>26</td>
<td>←</td>
<td>27</td>
</tr>
<tr>
<td>'04x</td>
<td>b</td>
<td>32</td>
<td>←</td>
<td>33</td>
<td>←</td>
<td>34</td>
<td>←</td>
<td>35</td>
</tr>
<tr>
<td>'05x</td>
<td>←</td>
<td>40</td>
<td>←</td>
<td>41</td>
<td>←</td>
<td>42</td>
<td>←</td>
<td>43</td>
</tr>
<tr>
<td>'06x</td>
<td>←</td>
<td>48</td>
<td>←</td>
<td>49</td>
<td>←</td>
<td>50</td>
<td>←</td>
<td>51</td>
</tr>
<tr>
<td>'07x</td>
<td>←</td>
<td>56</td>
<td>←</td>
<td>57</td>
<td>←</td>
<td>58</td>
<td>←</td>
<td>59</td>
</tr>
<tr>
<td>'10x</td>
<td>←</td>
<td>64</td>
<td>←</td>
<td>65</td>
<td>←</td>
<td>66</td>
<td>←</td>
<td>67</td>
</tr>
<tr>
<td>'11x</td>
<td>←</td>
<td>72</td>
<td>←</td>
<td>73</td>
<td>←</td>
<td>74</td>
<td>←</td>
<td>75</td>
</tr>
<tr>
<td>'12x</td>
<td>←</td>
<td>80</td>
<td>←</td>
<td>81</td>
<td>←</td>
<td>82</td>
<td>←</td>
<td>83</td>
</tr>
<tr>
<td>'13x</td>
<td>←</td>
<td>88</td>
<td>←</td>
<td>89</td>
<td>←</td>
<td>90</td>
<td>←</td>
<td>91</td>
</tr>
<tr>
<td>'14x</td>
<td>←</td>
<td>96</td>
<td>←</td>
<td>97</td>
<td>←</td>
<td>98</td>
<td>←</td>
<td>99</td>
</tr>
<tr>
<td>'15x</td>
<td>←</td>
<td>104</td>
<td>←</td>
<td>105</td>
<td>←</td>
<td>106</td>
<td>←</td>
<td>107</td>
</tr>
<tr>
<td>'16x</td>
<td>←</td>
<td>112</td>
<td>←</td>
<td>113</td>
<td>←</td>
<td>114</td>
<td>←</td>
<td>115</td>
</tr>
<tr>
<td>'17x</td>
<td>←</td>
<td>120</td>
<td>←</td>
<td>121</td>
<td>←</td>
<td>122</td>
<td>←</td>
<td>123</td>
</tr>
<tr>
<td>'20x</td>
<td>←</td>
<td>128</td>
<td>←</td>
<td>129</td>
<td>←</td>
<td>130</td>
<td>←</td>
<td>131</td>
</tr>
<tr>
<td>'21x</td>
<td>←</td>
<td>136</td>
<td>←</td>
<td>137</td>
<td>←</td>
<td>138</td>
<td>←</td>
<td>139</td>
</tr>
<tr>
<td>'22x</td>
<td>←</td>
<td>144</td>
<td>←</td>
<td>145</td>
<td>←</td>
<td>146</td>
<td>←</td>
<td>147</td>
</tr>
<tr>
<td>'23x</td>
<td>←</td>
<td>152</td>
<td>←</td>
<td>153</td>
<td>←</td>
<td>154</td>
<td>←</td>
<td>155</td>
</tr>
<tr>
<td>'24x</td>
<td>←</td>
<td>160</td>
<td>←</td>
<td>161</td>
<td>←</td>
<td>162</td>
<td>←</td>
<td>163</td>
</tr>
<tr>
<td>'25x</td>
<td>←</td>
<td>168</td>
<td>←</td>
<td>169</td>
<td>←</td>
<td>170</td>
<td>←</td>
<td>171</td>
</tr>
<tr>
<td>'26x</td>
<td>←</td>
<td>176</td>
<td>←</td>
<td>177</td>
<td>←</td>
<td>178</td>
<td>←</td>
<td>179</td>
</tr>
<tr>
<td>'27x</td>
<td>←</td>
<td>184</td>
<td>←</td>
<td>185</td>
<td>←</td>
<td>186</td>
<td>←</td>
<td>187</td>
</tr>
<tr>
<td>'30x</td>
<td>←</td>
<td>192</td>
<td>←</td>
<td>193</td>
<td>←</td>
<td>194</td>
<td>←</td>
<td>195</td>
</tr>
<tr>
<td>'31x</td>
<td>←</td>
<td>200</td>
<td>←</td>
<td>201</td>
<td>←</td>
<td>202</td>
<td>←</td>
<td>203</td>
</tr>
<tr>
<td>'32x</td>
<td>←</td>
<td>208</td>
<td>←</td>
<td>209</td>
<td>←</td>
<td>210</td>
<td>←</td>
<td>211</td>
</tr>
<tr>
<td>'33x</td>
<td>←</td>
<td>216</td>
<td>←</td>
<td>217</td>
<td>←</td>
<td>218</td>
<td>←</td>
<td>219</td>
</tr>
<tr>
<td>'34x</td>
<td>←</td>
<td>224</td>
<td>←</td>
<td>225</td>
<td>←</td>
<td>226</td>
<td>←</td>
<td>227</td>
</tr>
<tr>
<td>'35x</td>
<td>←</td>
<td>232</td>
<td>←</td>
<td>233</td>
<td>←</td>
<td>234</td>
<td>←</td>
<td>235</td>
</tr>
<tr>
<td>'36x</td>
<td>←</td>
<td>240</td>
<td>←</td>
<td>241</td>
<td>←</td>
<td>242</td>
<td>←</td>
<td>243</td>
</tr>
<tr>
<td>'37x</td>
<td>←</td>
<td>248</td>
<td>←</td>
<td>249</td>
<td>←</td>
<td>250</td>
<td>←</td>
<td>251</td>
</tr>
</tbody>
</table>

Font Layout 10: The text companion font `tcrm1000` with the TS1 encoding scheme.
To obtain access to the symbols in the TC fonts, one can load the `textcomp` package, which not only redefines several existing symbol commands, it also adds many new ones. For example, \copyright, which is normally defined to be `\textcircled{c}`, is changed to print character 169 from an appropriate TC font. Character 191 is the symbol for the European currency unit, the euro, printed with `\texteuro`; however, better ways of producing it are presented in Section 2.5.8.

### G.4.5 Special character commands

Inspecting Font Layout 9, one notices that the EC fonts contain not only many single characters that are formed out of two CM symbols (like Ä = A + ¨) but also several characters that have no correspondence in the CM font layout at all. The first type is accommodated by internally redefining the action of accent and special character commands. The second set requires new commands that are recognized only when the T1 encoding is active. These are

- the ogonek accent `\k{o}`: 
- special letters `\DH = Đ` `\DJ = Đ` `\NG = Đ` `\TH = Đ`
- `\dh = đ` `\dj = đ` `\ng = ĩ` `\th = đ`
- special symbols `\guillemotleft = «` `\guillemotright = »`
- `\guilsinglleft = ‹` `\guilsinglright = ›`
- `\quotedblbase = “` `\quotesinglbase = ’`
- `\textquotedbl` = “

When issued in OT1 encoding, these commands print an error message.

**Note:** The `\guillemotleft` and `\guillemotright` are not misprints even though the proper word for the French quotations marks is *guillemet*. The PostScript fonts contain these erroneous names for these symbols and this mistake has propagated to such an extent that it can never be removed from all the software that includes it. A *guillemot* is in fact an Arctic bird, not a French quotation mark.

### G.5 PostScript fonts

PostScript fonts, also known as outline or type 1 fonts, are treated exactly the same way as the METAFONT fonts as far as \LaTeX{} is concerned: during the processing, a `.tfm` file is read in for each font specifying the character sizes and other properties. This is all that \LaTeX{} needs to know, for it is the task of the driver program to print the actual character that fills the reserved space.

The driver makes use of the *virtual font* mechanism, which means that what \LaTeX{} sees are actually artificial fonts that do not really exist on their
own. There are .tfm files for these fonts, so the \LaTeX processing proceeds as normal. What the driver then does is to read a .vf file instead of the pixel .pk files. The instructions in the virtual font file tell the driver how to create each character: they may be drawn, taken from other fonts, or distorted. This is how PostScript slanted and small caps fonts are emulated, for such fonts do not exist in the ‘raw’ form. See Section 10.1.4 for example of how this works.

Even the font layout can be redesigned with virtual fonts. The raw PostScript fonts have an encoding scheme that conforms to neither OT1 nor T1, but which is used as a pool of symbols for constructing virtual fonts that do conform to these schemes. In particular, the Computer Modern fonts contain upper case Greek letters in the first 11 slots (Layout 1 on page 492) which are to be found in the PostScript symbol font only. The virtual font ptmr7t conforms to this by taking its characters from both the raw Times-Roman and Symbol fonts.

### G.5.1 Naming scheme for PostScript fonts

In order to be compatible with all possible operating systems, it is necessary to reduce the names of the PostScript font files to a maximum of eight characters. This makes for an extremely abbreviated and cryptic nomenclature.

The most commonly used scheme is that of Karl Berry. Here, the first letter of the name specifies the supplier of the font, for example p for Adobe (stands for PostScript), or h for Bigelow & Holmes, who designed the Lucida fonts used in this book, or m for Monotype, or l for Linotype, and so on.

<table>
<thead>
<tr>
<th>Font Name</th>
<th>Root Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>pagd</td>
<td>AvantGarde-Demi</td>
</tr>
<tr>
<td>pagdo</td>
<td>AvantGarde-DemiOblique</td>
</tr>
<tr>
<td>pagk</td>
<td>AvantGarde-Book</td>
</tr>
<tr>
<td>pagko</td>
<td>AvantGarde-BookOblique</td>
</tr>
<tr>
<td>pbkd</td>
<td>Bookman-Demi</td>
</tr>
<tr>
<td>pbkdi</td>
<td>Bookman-Demilight</td>
</tr>
<tr>
<td>pbk1</td>
<td>Bookman-Light</td>
</tr>
<tr>
<td>pbk1i</td>
<td>Bookman-LightItalic</td>
</tr>
<tr>
<td>pcrb</td>
<td>Courier-Bold</td>
</tr>
<tr>
<td>pcrbo</td>
<td>Courier-BoldOblique</td>
</tr>
<tr>
<td>pcr</td>
<td>Courier</td>
</tr>
<tr>
<td>pcrro</td>
<td>Courier-Oblique</td>
</tr>
<tr>
<td>phvb</td>
<td>Helvetica-Bold</td>
</tr>
<tr>
<td>phvbo</td>
<td>Helvetica-BoldOblique</td>
</tr>
<tr>
<td>phvbrn</td>
<td>Helvetica-Narrow-Bold</td>
</tr>
<tr>
<td>phvbon</td>
<td>Helvetica-Narrow-BoldOblique</td>
</tr>
<tr>
<td>phvr</td>
<td>Helvetica</td>
</tr>
<tr>
<td>phvro</td>
<td>Helvetica-Oblique</td>
</tr>
<tr>
<td>phvrrn</td>
<td>Helvetica-Narrow</td>
</tr>
<tr>
<td>phvron</td>
<td>Helvetica-Narrow-Oblique</td>
</tr>
<tr>
<td>pncb</td>
<td>NewCenturySchlbk-Bold</td>
</tr>
<tr>
<td>pncbi</td>
<td>NewCenturySchlbk-BoldItalic</td>
</tr>
<tr>
<td>pncr</td>
<td>NewCenturySchlbk-Roman</td>
</tr>
<tr>
<td>pncri</td>
<td>NewCenturySchlbk-Italic</td>
</tr>
<tr>
<td>pplb</td>
<td>Palatino-Bold</td>
</tr>
<tr>
<td>plpli</td>
<td>Palatino-BoldItalic</td>
</tr>
<tr>
<td>plpr</td>
<td>Palatino-Roman</td>
</tr>
<tr>
<td>plpri</td>
<td>Palatino-Italic</td>
</tr>
<tr>
<td>psyr</td>
<td>Symbol</td>
</tr>
<tr>
<td>psm</td>
<td>Times-Bold</td>
</tr>
<tr>
<td>psmi</td>
<td>Times-BoldItalic</td>
</tr>
<tr>
<td>ptmr</td>
<td>Times-Roman</td>
</tr>
<tr>
<td>ptmri</td>
<td>Times-Italic</td>
</tr>
<tr>
<td>pzcm</td>
<td>ZapfChancery-MediumItalic</td>
</tr>
<tr>
<td>pzdr</td>
<td>ZapfDingbats</td>
</tr>
</tbody>
</table>
Table G.3: A selection of the encoding suffixes in the Berry nomenclature

<table>
<thead>
<tr>
<th>Suffix</th>
<th>Encoding</th>
<th>NFSS designation</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7t</td>
<td>7-bit TeX text encoding</td>
<td>OT1</td>
<td>492</td>
</tr>
<tr>
<td>7m</td>
<td>TeX math italics encoding</td>
<td>OML</td>
<td>495</td>
</tr>
<tr>
<td>7v</td>
<td>TeX math extension encoding</td>
<td>OMX</td>
<td>496</td>
</tr>
<tr>
<td>7y</td>
<td>TeX math symbols encoding</td>
<td>OMS</td>
<td>495</td>
</tr>
<tr>
<td>8t</td>
<td>8-bit Cork encoding</td>
<td>T1</td>
<td>500</td>
</tr>
<tr>
<td>8a</td>
<td>Adobe standard encoding</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8r</td>
<td>TeXBase1Encoding</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A two-letter typeface code follows the supplier letter, such as tm for Times-Roman. Next come various letters to specify weight, for example r for regular (upright Roman) or b for bold, and variant, like i for italic, o for oblique (slanted). Next come a number plus letter to indicate the encoding scheme, followed by possible width code letters. The Berry names for the 35 standard PostScript fonts that should be loaded in every printer are listed in Table G.2 on the facing page, without the encoding suffixes.

The most important encoding suffixes for our purposes are listed in Table G.3. The 7-bit encodings have already been explained in Section G.2.2 and shown in Layouts 1, 5–7, while the 8t or T1 scheme is to be found in Layout 9 in Section G.4.3.

The other 8-bit schemes listed in Table G.3 are the raw encoding, 8r, and the Adobe standard encoding, 8a, which is the default for most type 1 fonts if no re-encoding is specified. This scheme plays absolutely no role in the NFSS installation for PostScript fonts.

Virtual fonts ptmro8t and ptmro8t are both constructed from the same raw font ptmro8r, differing only in the character assignments. There are also raw fonts that are modifications of the basic fonts. For example, ptmro8t is an oblique Times-Roman virtual font, based on ptmro8r, the raw version. However, there is no such font in the repertoire of Table G.2. This pseudo font is generated by applying a PostScript slanting operation to Times-Roman, as demonstrated in Section 10.1.4.

G.6 Computer Modern as PostScript fonts

The traditional pixel fonts produced by METAFONT have served us well in the days when the emphasis was on printed paper output. Even for PostScript output, the pixel fonts are acceptable for online viewing. However, they become very tedious when included in PDF output for
viewing: they appear very fuzzy and require a long time to draw. For this reason, PDF output, whether produced with pdf\TeX{}, dvipdfm, or converted from PostScript, should use type 1 fonts exclusively. To this end, the Computer Modern fonts themselves have been converted to type 1 fonts.

The first such undertaking was carried out by Basil Malyshev whose set of font files are known as bakoma. However, the set that is more often used today is the result of a joint project of Y&Y Inc. and Blue Sky Research. Originally part of their commercial \TeX{} installations, these fonts are now available free of charge. They are included on the \TeX{}Live CD under texmf/fonts/type1/bluesky/ in several subdirectories as .pfb files. They bear the same names and have the same font metrics as the corresponding pixel fonts (Appendix G) and therefore do not need any additional .tfm files. However, one must also get the mapping file bsr.map from texmf/dvips/bluesky/ and add it to the list of mapping files in config.ps.

The Bluesky fonts are incomplete in that certain less often used fonts are missing in some sizes. One can either include the map file bsr-interpolated.map, which substitutes the missing sizes with the nearest approximation, or employ the bakoma fonts that fill the gaps, with the map file bakomaextra.map. The \TeX{}Live CD does not include the complete bakoma set, but only those fonts that are missing in the Bluesky ensemble.

Together Bluesky and bakoma provide PostScript-encoded versions for all the Computer Modern fonts described in Section G.2, for text, math, symbols, decoration, pictures, along with the additional \AMS{} fonts for additional symbols and Cyrillic. However, these are only for the \OT1 encoding with 128 characters per font.

A more recent effort to apply PostScript to the Extended Computer (EC) fonts with the T1 encoding has been successfully completed by Vladimir Volovich with his cm-super collection. Each of the basic .pfb files contains from 468 to 585 characters, which by means of encoding vectors and mapping files can be used to emulate all the fonts in Section G.4, including the text companion fonts with TS1 encoding, and many other additional encoding schemes. They were generated from the original \METAFONT .mf source files and should therefore produce exactly the same results as the corresponding pixel fonts.

To use them, one must install the cm-super package from the \TeX{}Live CD, which copies the .pfb files from \texttt{\texmf/fonts/type1/cm-super/} and the mapping and encoding files from \texttt{\texmf/dvips/cm-super/}. The metric .tfm files are the same as those for the pixel files and so need not be provided extra.
Command Summary

This appendix contains a brief description of all the \LaTeX{} commands, in alphabetical (ASCII) order, neglecting the leading backslash \textbackslash, along with some \TeX{} commands that have been explained in this book. In the following section, the commands are presented in their logical grouping in a number of tables and figures.

H.1 Brief description of the \LaTeX{} commands

For each command in the following summary, the section and page number is given where it is introduced and described in detail. The numbers are shown in the form: ‘(Section) – Page’: for example (2.5.1) – 13 means ‘Section 2.5.1, page 13’. If these numbers are missing, then the command has not been presented in the book but is only mentioned here.

The following notations may be added to the commands:

[m] those permitted in math mode only;

[a] those belonging to \AMS-L\LaTeX{};

[p] those allowed only in the preamble.

\textbackslash\textbackslash . . . . . . . . . . . . . . . . . . . . . . . . . . . (2.1), (2.7.1) – 19, 29

Normal space between words after a command without arguments or after a period that is not the end of a sentence.

! . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (9.4.2) – 226

Field separation character within the \texttt{\index} command. For example: with \texttt{\index{command!fragile}} one produces an index sub-entry ‘fragile’ under the main entry ‘command’.

!‘ produces i . . . . . . . . . . . . . . . . . . . . . . . . . . . (2.5.6) – 24

\textbackslash! [m] . . . . . . . . . . . . . . . . . . . . . . . . . . . . (5.5.1) – 145

In math mode, a negative space of \(-1/6\) quad: \texttt{xx\!x = xxx}.
Appendix H. Command Summary

" .................................................. (2.5.2), (9.4.2), (14.2) – 23, 227, 311
1. In normal text, this produces the double closing quote ".
2. Literal sign for MakeIndex, in order to print one of the special characters !, @, |, or " . Example: \index{"!} to enter character ! without it being interpreted as a separation character.
3. Delimiter for a text field in Bib\TeX. Example: AUTHOR = "Donald E. Knuth".

\" .................................................. (2.5.7) – 24
Produces an umlaut accent: "{a} = ä.

# .................................................. (8.3.2), (8.4.2) – 187, 198
Argument replacement character in a user-defined command or environment.

### .................................................. (8.5.6) – 204
Replacement character for an internal argument within a nested user-defined command or environment.

\# .................................................. (2.5.4) – 23
Command to produce a hash symbol: \# = #.

$ .................................................. (5.1) – 119
Toggle character for switching between text and in-line math modes. On the first appearance (text to math) it behaves the same as \( or \begin{math}, while the second call (math to text) is as \) or \end{math}.

\$ .................................................. (2.5.4) – 23
Command to produce a dollar sign: \$ = $.

% .................................................. (4.11) – 118
Comment character. The rest of the line of text following % is ignored by the \TeX processing.

\% .................................................. (2.5.4) – 23
Command to produce a percent sign: \% = %.

& .................................................. (4.8.1) – 97
Indicates a new column in \array and \tabular environments.

\& .................................................. (2.5.4) – 23
Command to produce an ampersand symbol: \& = &.

\’ .................................................. (2.5.7), (4.6.4) – 24, 83
1. Command to produce an acute accent: \’ a = á.
2. Within the \texttt{tabbing} environment, a command to jump to the end of the current column.

\begin{itemize}
    \item \texttt{( )} \hspace{1cm} \texttt{(13.1.2), (14.2) – 289, 312} For a picture command in picture environment, specifies a coordinate pair. In \LaTeX, an alternative form for the outermost grouping of the entry type.
    \item \texttt{\textbackslash (} \hspace{1cm} \texttt{(5.1) – 119} Switches from text to in-line math mode to produce formulas within a line of text. It functions the same as \texttt{\begin{math}} and as a \$ sign in text mode.
    \item \texttt{\textbackslash )} \hspace{1cm} \texttt{(5.1) – 119} Switches back from in-line math mode to text mode. It functions the same as \texttt{\end{math}} and as a \$ sign in math mode.
    \item \texttt{\textbackslash +} \hspace{1cm} \texttt{(4.6.3) – 82} Within the \texttt{tabbing} environment, increments the left margin by one tab stop (moves it to the right).
    \item \texttt{\textbackslash ,} \hspace{1cm} \texttt{(2.7.1), (5.5.1) – 29, 145} Small space, the size of 1/6 quad, for use in text and math mode: \texttt{xx\,x} = \texttt{xxx}.
    \item \texttt{-} \hspace{1cm} \texttt{(2.5.3) – 23} As \texttt{-}, produces the hyphen - for compound words and word division, as \texttt{--}, the en dash –, and as ---, the em dash —.
    \item \texttt{\textbackslash -} \hspace{1cm} \texttt{(2.8.1), (4.6.3) – 35, 82} 1. Denotes possible word division. If a word contains at least one \texttt{\textbackslash -} the normal word division rules are suspended for that word and division may occur only at those locations.
        2. Within the \texttt{tabbing} environment, decrements the left margin by one tab stop (moves it to the left).
    \item \texttt{\textbackslash .} \hspace{1cm} \texttt{(2.5.7) – 24} Command to produce a dot accent: \texttt{\textbackslash .} = \texttt{\textdeg}.
    \item \texttt{\textbackslash /} \hspace{1cm} \texttt{(2.5.10) – 26} Command to break up ligatures: \texttt{shelf\textbackslash ful} = shelfful.
    \item \texttt{\textbackslash : [m]} \hspace{1cm} \texttt{(5.5.1) – 145} In math mode, a medium space, the size of 2/9 quad: \texttt{xx\,x} = \texttt{xxx}.
    \item \texttt{\textbackslash ; [m]} \hspace{1cm} \texttt{(5.5.1) – 145} In math mode, a large space, the size of 5/18 quad: \texttt{xx\,x} = \texttt{xxx}.
\end{itemize}
Within the tabbing environment, moves to the left by one tab stop.

1. Command to produce a macron accent: \(\o\) = \(\bar{\o}\).

2. Within the tabbing environment, sets a tab stop at the current position within the line.

Within the tabbing environment, moves right to the next tab stop.

\texttt{?i} produces \(\ddot{i}\).

1. In MakeIndex, separates an entry in an \texttt{index} command into a lexical (for alphabetization) and printing part. Example: \texttt{\index{sum@$\sum$}} means that the entry appears in the index at the location of the word 'sum' but what is printed is the summation sign $\sum$.

2. In \LaTeX, denotes the entry type. Example: \texttt{\@BOOK} indicates that the following literature entries correspond to those of a book.

Extra space at the end of a sentence ending with a capital letter.

After commands or environment calls, specifies an optional argument.

Switches from text mode to displayed math mode for putting a formula on a line by itself. Has the same effect as \texttt{\begin{displaymath}} or $$ in text mode.

Ends the current line (without right justifying it). The optional argument \texttt{[space]} inserts additional vertical spacing of length \texttt{space} before the next line.

The same as \texttt{\backslash} but prevents a page break from occurring between the current and next line.

Switches back from displayed math mode to text mode. Has the same effect as \texttt{\end{displaymath}} or $$ in math mode.
H.1. Brief description of the \LaTeX commands

\[ x^2 = x^2, \quad x^{-2n} = x^{-2n}. \]

\( \hat{\text o} \) produces an accent: \( \hat{\text o} = \hat{\text o} \).  

\( \hat{\text m} \) produces \( \hat{\text m} \).  

\[ x^{-2n} \]

\( \hat{\text n} \) produces \( \hat{\text n} \).  

\( \text m \) produces \( \text m \).

Exponents and superscripts in equations:

\[ x^2 = x^2, \quad x^{-2n} = x^{-2n}. \]

Exponents and superscripts in equations:

\[ x^2 = x^2, \quad x^{-2n} = x^{-2n}. \]

Subscripts in equations:

\[ a_n = a_n, \quad a_{i,j,k} = a_{i,j,k}. \]

Subscripts in equations:

\[ a_n = a_n, \quad a_{i,j,k} = a_{i,j,k}. \]

Command to produce a circumflex accent: \( \text o = \text o \).

Command to produce the underbar sign: \( \text v = \text v \).

Command to produce a circumflex accent: \( \text o = \text o \).

Command to produce a grave accent: \( \text o = \text o \).

Within the \textit{tabbing} environment, pushes the following text up against the right margin of the line.

Within the \textit{tabbing} environment, pushes the following text up against the right margin of the line.

1. After a command or environment call, specifies a mandatory argument.

2. Grouping a section of text to create a nameless environment.

3. In \texttt{BibTeX}, delimiting the name of an entry type, as well as an alternative delimiter for the text field.

\{ produces a left curly brace: \[ \{ = \{ \] produces a left curly brace: \[ \{ = \{ \]

In \texttt{MakeIndex}, the command character within a \texttt{\index} command.

In \texttt{MakeIndex}, the command character within a \texttt{\index} command.

1. After the command \texttt{\newcommand{\ii}[1]{\textit{#1}}} has been defined, \texttt{\index{entry|ii}} produces the page number for ‘entry’ in the index in italic type.

2. The cross-reference command \texttt{\see} from \texttt{makeidx.sty} can be invoked with \texttt{\index{bison|see{buffalo}}} to produce cross-references within the index.

\| produces a right curly brace: \[ \} = \} \]

A normal space between words, but without the possibility that the line will be broken there. Example: \texttt{Prof. ‘Jones ensures that ‘Prof.’ and ‘Jones’ both remain on the same line.}

\| produces a right curly brace: \[ \} = \} \]

\( \text n \) produces \( \text n \).

\( \text n \) produces \( \text n \).

Command to produce a tilde accent: \( \tilde{n} = \tilde{n} \).

Command to produce a tilde accent: \( \tilde{n} = \tilde{n} \).
Appendix H. Command Summary

\(\texttt{a=}\) ........................................... (4.6.4) – 83
Produces a macron accent within tabbing environment: \(\texttt{a=ɔ} = \bar{o}\).

\(\texttt{a'}\) ................................................ (4.6.4) – 83
Produces an acute accent within tabbing environment: \(\texttt{a'ɔ} = \acute{o}\).

\(\texttt{a"}\) ............................................... (4.6.4) – 83
Produces a grave accent within tabbing environment: \(\texttt{a"ɔ} = \grave{o}\).

\(\texttt{AA}\) produces Å ................................ (2.5.6) – 24
\(\texttt{aa}\) produces â ................................ (2.5.6) – 24
\texttt{above displayskip}[m] ......................... (5.5.4) – 149
Vertical space between a \emph{long} displayed equation and the preceding line of text. A new value may be assigned with the \texttt{setlength} command:

\texttt{setlength{above displayskip}{10pt plus2pt minus5pt}}

\texttt{abovedisplayshortskip}[m] ..................... (5.5.4) – 149
Vertical space between a \emph{short} displayed equation and the preceding line of text. A new value may be assigned with the \texttt{setlength} command as in the above example.

\texttt{abstractname} ................................. (D.4.1) – 460
Command containing the heading for the abstract. In English, this is ‘Abstract’ but may be altered for adaptation to other languages.

\texttt{acute{\texttt{x}}}[m] ............................... (5.3.9) – 129
Acute accent over math variable \(x\): \texttt{acute{a} = \acute{a}}

\texttt{Acute{\texttt{x}}}[m][a] .......................... (12.2.2) – 263
With the \texttt{amsmath} package, can be used like \texttt{acute}, but with multiple \texttt{AMS-TEX} math accents the positioning will be correct.

\texttt{addcontentsline{file}{format}{entry}} ........ (3.4.3), (3.4.4) – 59, 60
Manual addition of \texttt{entry} into the list file .toc, .lof, or .lot, according to the value of \texttt{file}, to be formatted as the heading of a sectioning command, as given by \texttt{format}, for example

\texttt{addcontentsline{toc}{section}{References}}

\texttt{address{sender}} ............................... (16.1) – 351
In the \texttt{letter} document class, enters the sender’s address. Multiple lines in \texttt{sender} are separated by \texttt{\\}.

\texttt{addtime{secs}} ................................. (15.1.3) – 327
In the \texttt{slides} class, if the option \texttt{clock} has been selected, a time marker, in minutes, appears at the bottom of the notes. This command adds the specified number of seconds to the marker. See also \texttt{settime}. 
H.1. Brief description of the \LaTeX{} commands

\addtocontents{file}\{entry\} \hspace{1cm} (3.4.3), (3.4.4) - 59, 60
Manual addition of entry into the list file .toc, .lof, or .lot, according to the value of file. Example:
\addtocontents{lof}\{\protect\newpage\}

\addtocounter{counter}\{number\} \hspace{1cm} (8.1.3) - 182
Adds number to the current value of the number stored in counter.

\addtolength{length}\{length\} \hspace{1cm} (8.2) - 184
Adds the quantity length to the current value of the length command \texttt{length\_name}.

\addvspace{\texttt{length}} \hspace{1cm} (8.2) - 185
Inserts vertical spacing of amount length between paragraphs at the point where the command is given. If other vertical spacing exists, the total will not exceed length.

\AE \ produces \texttt{Æ} \hspace{1cm} (2.5.6) - 24
\ae \ produces \texttt{æ} \hspace{1cm} (2.5.6) - 24
\aleph \texttt{[m]} \ produces \texttt{ℵ} \hspace{1cm} (5.3.6) - 127
\allowdisplaybreaks\{num\} \texttt{[p][a]} \hspace{1cm} (12.2.7) - 278
With the \texttt{amsmath} package, allows automatic page breaks to occur within multiline math formulas. If \texttt{num} is present, it takes a value of 0-4 to increase the ease with which page breaks occur. Without this command, a manual page break can be made at the end of any formula line with \texttt{displaybreak}.

\Alph{counter} \hspace{1cm} (8.1.4) - 183
Prints the current value of counter as a capital letter.

\alph{counter} \hspace{1cm} (8.1.4) - 183
Prints the current value of counter as a lower case letter.

\alpha \texttt{[m]} \ produces \texttt{α} \hspace{1cm} (5.3.1) - 125
\alslname \hspace{1cm} (D.4.1) - 460
Command for use in the \texttt{makeidx} package. It prints the text for a command \texttt{\seealso}. In English, this is ‘see also’ but may be altered for adaptation to other languages.

\amalg \texttt{[m]} \ produces \texttt{⨿} \hspace{1cm} (5.3.3) - 125
\and \hspace{1cm} (3.3.1) - 53
Used to separate author names within the \texttt{author} command for generating a title page with \texttt{maketitle}.

\angle \texttt{[m]} \ produces \texttt{∠} \hspace{1cm} (5.3.6) - 127
\appendixname \hspace{1cm} (D.4.1) - 460
Command containing the heading for the appendix. In English, this is ‘Appendix’ but may be altered for adaptation to other languages.
\approx \text{ produces } \approx \quad (5.3.4) - 126
\arabic{\text{counter}} \quad (8.1.4) - 183

Prints the current value of \text{counter} as an Arabic number.

\arccos \text{ produces } \arccos \quad (5.3.8) - 128

Command to produce the function name 'arccos' in equations.

\arcsin \text{ produces } \arcsin \quad (5.3.8) - 128

Command to produce the function name 'arcsin' in equations.

\arctan \text{ produces } \arctan \quad (5.3.8) - 128

Command to produce the function name 'arctan' in equations.

\arg \text{ produces } \arg \quad (5.3.8) - 128

Command to produce the function name 'arg' in equations.

\arraycolsep \quad (4.8.2) - 98

Half the width of the intercolumn spacing in the array environment.
Value is assigned with the \LaTeX\ command \texttt{\setlength}:
\begin{verbatim}
\setlength{\arraycolsep}{3mm}
\end{verbatim}

\arrayrulewidth \quad (4.8.2) - 98

The thickness of vertical and horizontal lines in the array and table environments. Its value is assigned to a length with \texttt{\setlength}:
\begin{verbatim}
\setlength{\arrayrulewidth}{0.5mm}
\end{verbatim}

\arraystretch \quad (4.8.2) - 98

Factor to change the spacing between lines in a table, normal value being 1. Spacing is multiplied by this factor, which is set to a new value with \texttt{\renewcommand{\arraystretch}{factor}}.

\ast \text{ produces } \ast \quad (5.3.3) - 125
\asymp \text{ produces } \asymp \quad (5.3.4) - 126

\AtBeginDocument{\texttt{\textbackslash code} }[p] \quad (D.2.4) - 444

Stores the \texttt{\textbackslash code} to be inserted into the processing stream when \texttt{\begin{document}} is executed. Commands that are only allowed in the preamble may be part of \texttt{\textbackslash code}. A package might include coding in this way to ensure that it is not overwritten by another package.

\AtEndDocument{\texttt{\textbackslash code} }[p] \quad (D.2.4) - 444

Stores the \texttt{\textbackslash code} to be inserted into the processing stream when \texttt{\end{document}} is executed. A package might include coding in this way to have additional features printed automatically at the end of the document.
\AtEndOfClass{code} \[p\] (D.2.4) – 444
Stores the code to be inserted into the processing stream when the current class file has finished being read. May only be given in a class file, or in another file that is read by a class file. May be used by a local configuration file to overwrite defaults in the class file itself.

\AtEndOfPackage{code} \[p\] (D.2.4) – 444
Stores the code to be inserted into the processing stream when the current package file has finished being read. May only be given in a package file, or in another file that is read by a package file. May be used by a local configuration file to overwrite defaults in the package file itself.

\author{name} (3.3.1) – 53
Enters the author name(s) for a title page produced by the \maketitle command.

\b{x} (2.5.7) – 24
Command to produce an underbar accent: \b{o} = \bar{o}.

\backmatter (3.3.5) – 57
In the book class, introduces the material that comes at the end (bibliography, index) by turning off the chapter numbering of the \chapter command.

\backslash produces \ (5.3.6) – 127
\bar{x} \[m\] (5.3.9) – 129
Macron accent over the math variable x: \bar{a} = \bar{a}.

\Bar{x} \[m]\[a] (12.2.2) – 263
With the amsmath package, can be used like \bar, but with multiple AMS-LATEX math accents the positioning will be correct.

\baselineskip (3.2.4) – 46
Interline spacing within a paragraph. Every font has its own internal line spacing. A new value (a rubber length) may be assigned with \setlength:
\setlength{\baselineskip}{12pt plus2pt minus1pt}

\baselinestretch (3.2.4), (4.1.2) – 46, 63
A factor with the normal value of 1 by which the internal length \baselineskip is multiplied to produce the actual interline spacing. May be changed with:
\renewcommand{\baselinestretch}{factor}
The new value takes effect after the next change in font size!
\begin{envrnmnt} . . . . . . . . . . . . . . . . . . . . . (2.2) – 19
Start of an environment with the name \textit{envrnmnt}. This command
must be coupled with \texttt{\end{envrnmnt}} to terminate the environment.
The environment name in both these commands must be identical.

\begin{abstract} . . . . . . . . . . . . . . . . . . . . . (3.3.2) – 55
Start of the environment \texttt{abstract} to produce an abstract. With
document class \texttt{article}, font size \texttt{small} and the quotation
environment are selected. With \texttt{report}, the abstract appears on a
separate page with normal font size and line width. In both cases,
the heading \texttt{Abstract} is centered above the text.

\begin{align} . . . . . . . . . . . . . . . . . . . . . (12.2.6) – 274
With the \texttt{amsmath} package, switches to displayed math mode to pro-
duce a set of aligned equations. Line are terminated by \texttt{\\} commands.
The lines are split into columns aligned on the first, third, fifth … \&
characters. Each line receives an equation number unless the \texttt{*}-form
of the environment has been selected.

\begin{alignat}{num} . . . . . . . . . . . . . (12.2.6) – 275
Is the same as the \texttt{align} environment except that no spacing is
inserted automatically between the column pairs. The argument \texttt{num}
is the number of column pairs = \((1 + n_\&)/2\) where \(n_\&\) is the number
of \& signs in one row. Explicit spacing may be placed between column
pairs, especially if the left part of that pair is otherwise empty.

\begin{aligned} . . . . . . . . . . . . . (12.2.6) – 276
With the \texttt{amsmath} package, is like the \texttt{align} environment but is
used as an element within math mode. The optional argument \texttt{pos}
determines the vertical positioning relative to neighboring elements:
t or b for top or bottom, no argument for centering.

\begin{appendix} . . . . . . . . . . . . . . . . . . . . . (3.3.4) – 57
Start of the environment \texttt{appendix} to produce an appendix. The
main section counter is reset to zero and its numbering appears as
capital letters.

\begin{array}{pos} . . . . . . . . (4.8.1), (5.4.3) – 95, 134
Start of the environment \texttt{array} to produce matrices and arrays in
math mode. The column definition \texttt{col} contains a formatting charac-
ter for each column. Thus \texttt{\begin{array}{lcr}} produces an array
with three columns: one \textit{left justified}, one \textit{centered}, and one \textit{right
justified}. The optional parameter \texttt{pos} determines how the array is
aligned vertically with text outside it on the same line: \texttt{t} with the top
line, \texttt{b} with the bottom line, while the default is with the center. See
also \texttt{\begin{tabular}}

\begin{center} . . . . . . . . . . . . . . . . . . . . . (4.2.1) – 67
Start of the environment \texttt{center}. Each line of text terminated by \texttt{\\}
appears centered. See also \texttt{\centering}.
H.1. Brief description of the \LaTeX\ commands 517

\begin{command\_name} . . . . . . . . . . . . . . . . . (2.2) – 20
Most declaration commands, such as the font styles and sizes, can be
used as environment names. For example, \begin\{small\} switches
to font size \texttt{\small} until the countercommand \end\{small\} is given.

\begin{alltt} . . . . . . . . . . . . . . . . . (4.9.1) – 111
When the alltt package has been loaded, this environment prints
out original text in typewriter typeface, maintaining line breaks, special
symbols, and so on, except for \{\} which function as usual.
This allows commands to be executed within the typewriter text.

\begin\{bmatrix\}[m][a] . . . . . . . . . . . . . . . (12.2.4) – 266
Like the matrix environment, but enclosed in square brackets [].

\begin\{Bmatrix\}[m][a] . . . . . . . . . . . . . . . (12.2.4) – 266
Like the matrix environment, but enclosed in curly braces {}.

\begin\{cases\}[m][a] . . . . . . . . . . . . . . . (12.2.6) – 277
With the amsmath package, writes math expressions on several lines,
terminated by \textbackslash{}, in left-justified columns, separated by &,
with a curly brace enclosing all lines at the left, the whole being centered
vertically.

\begin\{description\} . . . . . . . . . . . . . . . . . (4.3.3) – 70
Start of the environment description to produce an indented list
with labels. The label text is the argument label in the command
\item[label].

\begin\{displaymath\} . . . . . . . . . . . . . . . . . (5.1) – 120
Switches from text to displayed math mode for producing a formula
on a line by itself. Functions the same as \[.\]

\begin\{document\} . . . . . . . . . . . . . . . . . . . (1.5.2) – 12
Start of the outermost environment of a text document. This command
terminates the preamble. It is obligatory for every \LaTeX\ docu-
ment, as is its counterpart \end\{document\} for ending the docu-
ment.

\begin\{enumerate\} . . . . . . . . . . . . . . . . . (4.3.2) – 70
Start of the environment enumerate to produce a numbered, in-
dented listing. The style of numbering depends on the depth of
nesting; at the first level, it consists of a running Arabic number that
is incremented with each call to \item.

\begin\{eqnarray\}[m] . . . . . . . . . . . . . . . . . (5.4.7) – 138
Switches from text to displayed math mode to produce a set of equa-
tions or a multiline formula in the form of a three-column table \{rc\}.
The individual lines of the formula are ended with the command \textbackslash{}
the fields within a line are separated by \& characters. Each line is
given a sequential equation number unless the command \nonumber
appears within it.
\begin{eqnarray*} [m] \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (5.4.7) - 138
Is the same as the eqnarray environment except that no equation numbers are printed.
\end{eqnarray*}

\begin{equation} [m] \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (5.1) - 120
Switches from text to displayed math mode to produce a formula on a line by itself, including an automatic sequential equation number.
\end{equation}

\begin{falign} [a] \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (12.2.6) - 275
With the amsmath package, is the same as the align environment except that spacing is inserted between the column pairs to fill up the entire line.
\end{falign}

\begin{figure}\[loc\] \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (7.1) - 169
Float environment for entering text for a figure. The optional argument \textit{loc} can be any combination of the letters h, t, b, and p to determine the various positioning possibilities. Default is \texttt{tbp}. The character \texttt{!} may additionally be given to ignore all float spacing and number restrictions set by the float style parameters.
\end{figure}

\begin{figure*}\[loc\] \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (7.1) - 169
The same as the figure environment except that the figure is inserted over the width of two columns when the option \texttt{twocolumn} or the command \texttt{\twocolumn} has been selected. The standard form \texttt{figure} will only fill the width of one column.
\end{figure*}

\begin{filecontents}{file name} \[p\] \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (D.2.9) - 448
An environment that may be given only before \texttt{\documentclass}, it writes its lines literally to a file of the specified name, if that file does not already exist. It also adds comment lines stating its source. In this way, non-standard files may be included in the main document file for shipment to other installations. If a file with the stated name already exists, it is not overwritten, but a warning message is issued.
\end{filecontents}

\begin{filecontents*}{file name} \[p\] \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (D.2.9) - 448
Is the same as the filecontents environment, except that no comment lines are written to the file. The file will contain exactly the contents of the environment, and nothing more.
\end{filecontents*}

\begin{flushleft} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (4.2.2) - 67
Start of the flushleft environment in which each line of text is left justified, that is, it begins flush with the left margin but is not expanded to match the right edge. The equivalent declaration is \texttt{\raggedright}.
\end{flushleft}

\begin{flushright} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (4.2.2) - 67
Start of the flushright environment in which each line of text is right justified, that is, the right-hand side is flush with the right margin, but the line is not expanded to start exactly at the left edge. The equivalent declaration is \texttt{\raggedleft}.
H.1. Brief description of the \LaTeX\ commands

\begin{gather}
. . . . . . . . . . . . . . . . . (12.2.6) – 273
\end{gather}

With the amsmath package, switches to displayed math mode to produce several lines of equations, all centered with no alignment. Lines are terminated by \\
commands. Each line receives an equation number, unless the *-form of the environment has been selected.

\begin{gathered}
. . . . . . . . . . . . . . . . . (12.2.6) – 276
\end{gathered}

With the amsmath package, is like the gather environment but is used as an element within math mode. The optional argument \texttt{pos} determines the vertical positioning relative to neighboring elements: \texttt{t} or \texttt{b} for top or bottom, no argument for centering.

\begin{itemize}
. . . . . . . . . . . . . . . . . . . . (4.3.1) – 69
\end{itemize}

Start of the \texttt{itemize} environment for producing labeled, indented listings. The type of label depends on the depth of nesting; at the first level it is a • generated by each \texttt{item} command.

\begin{longtable}
. . . . . . . . . . . . . . . . . . (4.8.4) – 108
\end{longtable}

With the \texttt{longtable} package, begins a table with the same syntax as \texttt{tabular} but which will continue to other pages as necessary. The head and footlines on the continued pages can be specified, terminated with \texttt{\endhead}, \texttt{\endfirsthead}, \texttt{\endfoot}, \texttt{\endlastfoot}.

\begin{letter}{\texttt{recipient}}
. . . . . . . . . . . . . . (16.1) – 352
\end{letter}

Start of a letter with the document class \texttt{letter}. Name and address of the \texttt{recipient} are given within the second pair of brackets; lines of text within this argument are ended with the command \texttt{\.}

\begin{list}{\texttt{standard\_label}}{\texttt{list\_decl}}
. . . . . . . . . . . . . . . . (4.4) – 74
\end{list}

Start of a generalized list environment. The label is defined by \texttt{standard\_label}, which is generated by each \texttt{item} command. The desired list declarations are contained in \texttt{list\_decl} (see page 75).

\begin{lrbox}{\texttt{boxname}}
. . . . . . . . . . . . . . . . . (4.7.1) – 87
\end{lrbox}

Functions in a way similar to the command \texttt{sbox}, except that it is the text of the environment that is stored in the LR box named \texttt{boxname}, which has previously been created with \texttt{newsavebox{\texttt{boxname}}}. The contents of the box may be printed as often as desired with \texttt{usebox{\texttt{boxname}}}.

\begin{math}
. . . . . . . . . . . . . . . . . . . . . . . . . (5.1) – 119
\end{math}

Switches from text to in-line math mode to produce formulas within a line of text. This environment has the same effect as \texttt{\c{\c{\c{\c{\$}}} or $} in text mode.

\begin{matrix}
. . . . . . . . . . . . . . . . . (12.2.4) – 266
\end{matrix}

With the amsmath package, is the same as the \texttt{array} environment except that the column specifier argument may be omitted, without
which up to 10 centered columns may be entered. This maximum may be changed with the counter \texttt{MaxMatrixCols}. The environments \texttt{pmatrix}, \texttt{bmatrix}, \texttt{Bmatrix}, \texttt{vmatrix}, and \texttt{Vmatrix} function the same as \texttt{matrix} but are enclosed in braces \{, \}, | | and \| \|, respectively.

\begin{minipage} \[ pos \] \[ height \] \[ inner_pos \] \{ width \} . . . . . . . . . . . . . . . . (4.7.3), (4.7.5) - 88, 90

Environment to format text within a 'minipage' of width \texttt{width}. Its vertical positioning with respect to the surrounding text is determined by the optional argument \texttt{pos}: \texttt{t} for alignment with its top line, \texttt{b} with its bottom line, and centered with no argument. The other two optional arguments are: \texttt{height} to give the total height, and \texttt{inner_pos} to specify how the text is to be positioned inside it. Possible values are \texttt{t} for top, \texttt{b} for bottom, \texttt{c} for centered, and \texttt{s} to be stretched out to fill the whole vertical space. The default is the value of the external positioning \texttt{pos} option. The \texttt{height} argument may contain the parameters \texttt{\height}, \texttt{\depth}, \texttt{\width}, and \texttt{\totalheight}.

\begin{multicols} \{ num_cols \} \[ header \] \[ pre_space \] . . . . . . . . . . . . . . . . . . . . . . . . . (3.2.8), (B.5.4) - 51, 395

This environment is provided by the \texttt{multicol} package in the tools collection (Section B.5.4). It switches to printing the text in \texttt{num_cols} columns, with \texttt{header} printed in one column across the top. A new page is inserted only if the remaining space on the current page is less than \texttt{\premulticols} or the optional argument \texttt{pre_space}. A new page is inserted at the end if the remaining space is less than \texttt{\postmulticols}. The columns on the last page are balanced. Column separation and rule are set by the lengths \texttt{\columnsep} and \texttt{\columnseprule}.

With the starred version \texttt{multicols*}, the columns of text are not balanced on the last page of the environment.

\begin{multline} \[ a \] . . . . . . . . . . . . . . . . (12.2.6) - 271

With the \texttt{amsmath} package, switches to displayed math mode to produce a single equation over several lines. Line breaks are forced by \texttt{\} commands. The first line is to the far left, the last to the right, all others centered. With \texttt{\shoveleft\{form\}} and \texttt{\shoveright\{form\}}, single lines consisting of \texttt{form} may be pushed to the left or right. The single equation number appears at the right of the last line, or at the left of the first line, depending on class options \texttt{reqno} (default) and \texttt{leqno}, respectively. With the \texttt{*-form} of the environment, the equation number is suppressed.

\begin{note} . . . . . . . . . . . . . . . . . . . . . . . . . (15.1.2) - 326

In \texttt{slides} class, the environment for producing a note for the current slide. Notes are numbered with the current slide number followed by a hyphen and running number, for example 8-1, 8-2, etc.
H.1. Brief description of the \LaTeX\ commands

\begin{overlay} \hfill \end{overlay} (15.1.2) - 325

In \texttt{slides} class, the environment for producing an overlay for the current slide. Overlays are numbered with the current slide number followed by a lower case letter, for example 3-a, 3-b, etc. See also \begin{slide}.

\begin{picture}(x\,\text{dimen},y\,\text{dimen}) \hfill \end{picture} (13.1.2) - 288

Environment to generate a picture with the width \texttt{x\,\text{dimen}} and height \texttt{y\,\text{dimen}}, where the unit of length has previously been specified by the declaration \texttt{\unitlength}.

\begin{picture}(x\,\text{dimen},y\,\text{dimen})(x\,\text{offset},y\,\text{offset}) (13.1.6) - 301

Most general form of the call to the \texttt{picture} environment. The picture is displaced to the left by \texttt{x\,\text{offset}} and downwards by \texttt{y\,\text{offset}}.

\begin{pmatrix} \hfill \end{pmatrix} (12.2.4) - 266

Like the \texttt{matrix} environment, but enclosed in round parentheses ().

\begin{quotation} \hfill \end{quotation} (4.2.3) - 67

Start of the \texttt{quotation} environment in which text is indented on both sides relative to the normal page margins. Paragraphs within the environment are marked with an additional indentation of the first line.

\begin{quote} \hfill \end{quote} (4.2.3) - 67

The same as the \texttt{quotation} environment except that the first line of a paragraph is not indented but instead additional line spacing comes between paragraphs.

\begin{slide} \hfill \end{slide} (15.1.2), (15.2.1) - 325, 331

In \texttt{slides} and \texttt{seminar} classes, the main environment for producing a slide.

\begin{sloppypar} \hfill \end{sloppypar} (2.8.3) - 36

Inside this environment word spacings are allowed to stretch more generously than usual so that paragraphs are broken up into lines with fewer word divisions. See also \texttt{sloppy}. The countercommand is \texttt{\fussy}.

\begin{split} \hfill \end{split} (12.2.6) - 272

With the \texttt{amsmath} package, is used within a math environment such as \texttt{equation} to write a formula over several lines. Line breaks are forced with \texttt{\\} commands and the lines are horizontally aligned on the \& alignment marker. Any equation number is generated by the outer environment. It is either centered with the class option \texttt{centertags} (default) or with \texttt{tbtags} it appears at the right of the last line, or at the left of the first line, depending on class options \texttt{reqno} (default) and \texttt{leqno}, respectively.
Appendix H. Command Summary

\begin{subarray}{pos}{1st line\\..\last line} [m][a] \ (12.2.2) - 261

With the amsmath package, sets multiline text for superscripts and subscripts, like $\substack$, but the parameter \texttt{pos} may take on values \texttt{c} or \texttt{l} for centered or left-justified lines.

\begin{subequations} [a] \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ (12.2.7) - 277

With the amsmath package, numbers equations within it with a fixed main number and sequence of lower case letters attached, as 7a, 7b, 7c, ....

\begin{tabbing} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ (4.6.1) - 81

Start of the \texttt{tabbing} environment in which special tabbing commands become operational: \texttt{\=} sets a tab stop, \texttt{\>} jumps to the next stop, \texttt{\<} goes back a stop, \texttt{\\\} terminates and starts a new line, \texttt{\+} sets the left margin one tab stop further, \texttt{\-} moves the left margin back one stop.

\begin{table} \ [loc] \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ (4.8.5), (7.1) - 109, 169

Float environment for entering text for a table. The optional argument \texttt{loc} can be any combination of the letters \texttt{h}, \texttt{t}, \texttt{b}, and \texttt{p} to determine the various positioning possibilities. Default is \texttt{tbp}. The character \texttt{!} may additionally be given to ignore all float spacing and number restrictions set by the float style parameters.

\begin{table*} \ [loc] \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ (7.1) - 169

The same as the \texttt{table} environment except that the table is inserted over the width of two columns when the option \texttt{twocolumn} or the command \texttt{\twocolumn} has been selected. The standard form \texttt{table} will only fill the width of one column.

\begin{tabular} \ [pos]{cols} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ (4.8.1) - 95

Start of the \texttt{tabular} environment for producing tables. The argument \texttt{cols} contains a formatting character for each column in the table: \texttt{c} for centered text, \texttt{l} for left, \texttt{r} for right justification, or \texttt{p\{}wd\texttt{}} for a column of width \texttt{wd} in which the text may extend over several lines.

When the entry \texttt{@\{}text\texttt{}} appears between any two of the above column formatting characters, \texttt{text} is inserted in every row between those two columns. Where the character \texttt{|} appears, a vertical line is drawn in every row.

The optional argument \texttt{pos} specifies how the table is to be vertically aligned with the surrounding text: with no argument, it is centered, otherwise with \texttt{t} the top line, with \texttt{b} the bottom line is aligned with the external baseline.

The text entries of the individual columns are separated by \texttt{&} and the rows are terminated by \texttt{\\\}.
\begin{tabular*}{\textwidth}{\pos\cols}. . . . . . . . . . (4.8.1) – 95
The same as \begin{tabular} except that the total width of the table is given by the argument width. This may only be achieved successfully if there is rubber spacing between the columns. This may be added with @[\extracolsep\fill] somewhere within the cols format definition.
\end{tabular*}

\begin{verbatim}
\end{verbatim}

\begin{verbatim*}
The same as the \verbatim environment except that blanks are printed as \_ to make them visible.
\end{verbatim*}

\begin{verse}. . . . . . . . . . . . . . . . . . . . . (4.2.4) – 68
Environment for setting rhymes, poems, verses, etc. Stanzas are separated by blank lines, individual lines by the \\ command.
\end{verse}
\begin{vmatrix} [m] [a] \end{vmatrix} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldot
\bibname  
Command containing the heading for the bibliography in book and report document classes. In English, this is 'Bibliography' but may be altered for adaptation to other languages.

\bigbr_symbol [m]  
A bracket symbol larger than normal, but smaller than \Big. Example: \big(.

\Bigbr_symbol [m]  
A bracket symbol larger than \big, but smaller than \bigr. Example: \Big[.

\bigcap [m] produces ∩  
\bigcirc [m] produces ◯  
\bigcup [m] produces ∪  
\bigr_symbol [m]  
A bracket symbol larger than \Big, but smaller than \Bigr. Example: \bigr[.

\Bigr_symbol [m]  
The largest bracket symbol. Example: \Bigrangle.

\bigodot [m] produces ⊗  
\bigoplus [m] produces ⊕  
\bigotimes [m] produces ⊗  
\bigtriangledown [m] produces ▽  
\bigtriangleup [m] produces △  
\bigskip  
Inserts large vertical spacing of the amount \bigskipamount. See also \medskip and \smallskip.

\bigskipamount
Standard value for the amount of vertical spacing that is inserted with the command \bigskip. May be changed with the \setlength command:
\setlength{\bigskipamount}{5ex plus1.5ex minus2ex}

\bigsqcup [m] produces †  
\biguplus [m] produces ⊔  
\bigvee [m] produces ∨  
\bigwedge [m] produces ∧  
\binom{over}{under} [m][a]  
With the amsmath package, prints a binomial expression:
\[[\binom{n}{k}]\] yields \(\binom{n}{k}\).
Appendix H. Command Summary

\texttt{\textbackslash bmod} \ [m] \quad \ldots \ldots \ldots \ldots \quad (5.3.8), (12.2.5) - 129, 268
Command to produce the function name ‘mod’ in the form
\[ a \bmod b = a \mod b \]

\texttt{\textbackslash boldmath} \quad \ldots \ldots \ldots \ldots \quad (5.4.9) - 143
Switches to bold face for math mode. This command must be given in text mode, however, before going into math mode. To set only part of a formula in bold face, use \texttt{\textbackslash mbox}\{\textbackslash boldmath\$\ldots\$\} to return temporarily to text mode.

\texttt{\textbackslash boldsymbol\{symbol\}} \ [m][a] \quad \ldots \ldots \ldots \ldots \quad (12.2.1) - 258
When one of the packages amsmath or amsbsy has been loaded, this command prints \textit{symbol} in bold face. Unlike \texttt{\textbackslash mathbf}, it also affects math symbols and lower case Greek letters.

\texttt{\textbackslash bot} \ [m] \ produces \bot \quad \ldots \ldots \ldots \ldots \quad (5.3.6) - 127
\texttt{\textbackslash botfigrule} \quad \ldots \ldots \ldots \ldots \quad (7.3) - 173
A command that is executed before a float at the bottom of a page. It is normally defined to do nothing, but may be redefined to add a rule between the float and the main text. It must not add any net vertical spacing.
\begin{verbatim}
\renewcommand{\botfigrule}{\vspace*{-.4pt}\rule{\columnwidth}{.4pt}}
\end{verbatim}

\texttt{\textbackslash bottomfraction} \quad \ldots \ldots \ldots \ldots \quad (7.3) - 172
Maximum fraction of a page that may be taken up by floats at the bottom. May be set to a new value with:
\begin{verbatim}
\renewcommand{\bottomfraction}{\texttt{decimal,fraction}}
\end{verbatim}

\texttt{\textbackslash bottomnumber} \quad \ldots \ldots \ldots \ldots \quad (7.3) - 171
Maximum number of floats that may appear at the bottom of a page. Set to a new number with \texttt{\setcounter{bottomnumber}\{num\}}.

\texttt{\textbackslash bowtie} \ [m] \ produces \bowtie \quad \ldots \ldots \ldots \ldots \quad (5.3.4) - 126
\texttt{\textbackslash Box} \ [m] \ produces \Box \quad \ldots \ldots \ldots \ldots \quad (5.3.3) - 125
\texttt{\textbackslash boxed\{formula\}} \ [m][a] \quad \ldots \ldots \ldots \ldots \quad (12.2.5) - 270
With the amsmath package, sets the mathematical \textit{formula} in a box.

\texttt{\textbackslash breve\{x\}} \ [m] \quad \ldots \ldots \ldots \ldots \quad (5.3.9) - 129
Breve accent over the math variable \textit{x}: \texttt{\textbackslash breve\{a\}} = \breve{a}.

\texttt{\textbackslash Breve\{x\}} \ [m][a] \quad \ldots \ldots \ldots \ldots \quad (12.2.2) - 263
With the amsmath package, can be used like \texttt{\textbackslash breve}, but with multiple \texttt{\textbackslash AMS-\LaTeX} math accents the positioning will be correct.

\texttt{\textbackslash bullet} \ [m] \ produces \bullet \quad \ldots \ldots \ldots \ldots \quad (5.3.3) - 125
\texttt{\textbackslash c\{x\}} \quad \ldots \ldots \ldots \ldots \quad (2.5.7) - 24
Produces a cedilla under \textit{x}: \texttt{\textbackslash c\{C\}} = \c{C}.
H.1. Brief description of the \LaTeX commands

\cap [m] produces $\cap$ ........................................ (5.3.3) - 125
\caption[short_form]{caption_text} .................................... (7.4) - 173

Produces a numbered title or caption with the text \texttt{caption_text} within
the float environments \texttt{figure} or \texttt{table}. The \texttt{short_form} is the
abbreviated text appearing in the list of figures or tables, which is
the same as the \texttt{caption_text} if it is omitted.

\captionslanguage ............................................. (D.4.1), (11.1) - 459, 255

A command used in several language adaptations to redefine the
headings of special sections such as ‘Chapter’ and ‘Contents’. It
occurs in packages like \texttt{german} as well as in the \texttt{babel} system. This
command is normally part of the definition of the \texttt{selectlanguage}
command.

\cc{list} ...................................................... (16.1) - 353

Command within document class \texttt{letter} to generate ‘cc:’, copies,
followed by a list of names \texttt{list} at the end of the letter.

\ccname ...................................................... (D.4.1) - 460

Command in the \texttt{letter} document class containing the word to be
printed by the \texttt{cc} command. In English, this is ‘cc’ but may be
altered for adaptation to other languages.

\cdot [m] produces $\cdot$ ........................................ (5.3.3) - 125
\cdots [m] produces $\cdots$ ...................................... (5.2.6) - 123
\centering ...................................................... (4.2.1) - 67

Declaration to switch to centered lines of text, each input line being
terminated by \texttt{\textbackslash}. See also \texttt{\begin{center}}.

\centerline{text} ............................................ (4.2.1) - 67

An additional \LaTeX command that sets \texttt{text} centered on a horizontal
line by itself.

\cfrac\texttt{[pos]}\texttt{\{}\texttt{over}\texttt{\}}\texttt{\{}\texttt{under}\texttt{\}} [m][a] ............... (12.2.3) - 265

With the \texttt{amsmath} package, produces a continued fraction. The op-
tional argument \texttt{pos} may be \texttt{l} or \texttt{r} to have the numerator left or right
justified on the horizontal rule, otherwise it is centered.

\chapter[short_title]{title} .............................. (3.3.3) - 55

Starts a new chapter on a new page, with an automatic sequential
chapter number and \texttt{title} as header. If the optional \texttt{short_title} is given,
it appears in place of \texttt{title} in the table of contents and in the running
head at the top of the pages.

\chapter*[\texttt{title}] .................................... (3.3.3) - 55

Starts a new chapter on a new page, with \texttt{title} as header, but without a
chapter number. The entry does not appear in the table of contents.
\chapter{Command Summary}

\chaptername \hfill (D.4.1) \hfill 460
Command containing the chapter heading. In English, this is ‘Chapter’ but may be altered for adaptation to other languages.

\check{x} \hfill (5.3.9) \hfill 129
Háček accent over the math variable $x$: \check{$a$} = $\check{a}$.

\Check{x} \hfill (12.2.2) \hfill 263
With the amsmath package, can be used like \check, but with multiple AMS-\LaTeX math accents the positioning will be correct.

\CheckCommand{\comname} \hfill (D.2.5) \hfill 445
Tests that the current definition of \comname is as expected. If not, an error message is issued. This is used to ensure that important commands have not been altered by other packages.

\chi \hfill (5.3.1) \hfill 125
\circ \hfill (5.3.3) \hfill 125
\circle{diameter} \hfill (13.1.4) \hfill 295
\circle*[diameter] \hfill (13.1.4) \hfill 295
Picture element command to produce a circle with diameter diameter in the picture environment. To be used as an argument in a \put or \multiput command.

\cite\hfill (9.3.3), (14.1) \hfill 219, 310
\citet\hfill (9.3.4) \hfill 221
\citep\hfill (9.3.4) \hfill 221
Literature citation using the identifier key to produce a reference label in the text. The optional note text is included with the label.

\ClassError{class name} \hfill (D.2.7) \hfill 446
Writes an error message error text to the monitor and transcript file, labeled with the class name, and halts processing, waiting for a user response as for a \LaTeX error. If \texttt{H\langle return\rangle} is typed, the help text is printed. Both error text and help may contain \MessageBreak for a new line, \space for a forced space, and \protect before commands that are to have their names printed literally and not interpreted.
H.1. Brief description of the \LaTeX{} commands

\ClassInfo{\textit{class\_name}}{\textit{info\_text}} [p] ...................... (D.2.7) – 447

Is like \ClassWarningNoLine except that the text \textit{info\_text} is only written to the transcript file, and not to the monitor.

\ClassWarning{\textit{class\_name}}{\textit{warning\_text}} [p] .................. (D.2.7) – 446

Writes \textit{warning\_text} to the monitor and transcript file, labeled with the class name and the current line number of the input file. Processing continues. The \textit{warning\_text} is formatted in the same way as that for \ClassError.

\ClassWarningNoLine{\textit{class\_name}}{\textit{warning\_text}} [p] .... (D.2.7) – 446

Is like \ClassWarning except that the current line number of the input file is not printed.

\cleardoublepage .................................................. (2.7.4) – 34

Ends the current page and outputs all unprocessed floats on to one or more float pages. The next page will be a right-hand one, with an odd page number.

\clearpage ......................................................... (2.7.4) – 33

Ends the current page and outputs all unprocessed floats on to one or more float pages.

\cline{\textit{n-m}} .................................................. (4.8.1) – 97

In tabular environment, produces a horizontal rule from the beginning of column \textit{n} to the end of column \textit{m}. Example: \cline{2-5}.

\closing{\textit{regards}} ............................................ (16.1) – 353

End of the text in the \texttt{letter} environment; \textit{regards} stands for the desired terminating text.

\clubsuit \texttt{[m]} produces ♣ .................................... (5.3.6) – 127

\texttt{color \textit{col\_spec}} ........................................ (6.2) – 167

A command made available with the \texttt{color} package. It is a declaration that switches the color in which the text is printed to that specified. It remains in effect until the end of the current environment or until countermanded by another \texttt{color} command. The \textit{col\_spec} is either the name of a color defined (or predefined) by \texttt{definecolor} or of the form \texttt{[\textit{model}]{\textit{specs}}}, where the arguments have the same meaning as they do for \texttt{definecolor}. Examples:

\texttt{\color{rgb}{0.5,0.5,0} \color{magenta}}

\texttt{\colorbox{\textit{col\_spec}}{\textit{text}}} .......................... (6.2) – 167

A command made available with the \texttt{color} package. The \textit{text} is set in an LR box with the specified color as the background color. The \textit{col\_spec} is the same as for \texttt{color}.
\columnsep \text{(3.1.1) - 40}
Declaration for the amount of intercolumn spacing in two-column page formatting. May be changed with the \setlength command:
\begin{verbatim}
\setlength{\columnsep}{1pt}
\end{verbatim}

\columnseprule \text{(3.1.1) - 40}
Declaration for the thickness of the vertical rule separating the columns in two-column page formatting. Value is set with the \setlength command:
\begin{verbatim}
\setlength{\columnseprule}{1pt}
\end{verbatim}

\cong \text{[m]} \text{ produces } \equiv \text{ (5.3.4) - 126}
\text{\textbackslash contentsline}\{sec\_type\}\{numberline\{sec\_num\}\title\_text\}{\page}
Command that appears in the .toc file for every entry in the table of contents, which is read when the \tableofcontents command is given. Such commands may be altered or added to the .toc file by means of the text editor. The entry sec\_type stands for the sectioning level, such as section, while sec\_num is its number (for example, 2.3) and page is the page number where the entry appears.

\copyright \text{ produces } © \text{ (2.5.5) - 23}
\cos \text{[m]} \text{ (5.3.8) - 128}
Command to produce the function name 'cos' in formulas.

\cosh \text{[m]} \text{ (5.3.8) - 128}
Command to produce the function name 'cosh' in formulas.

\cot \text{[m]} \text{ (5.3.8) - 128}
Command to produce the function name 'cot' in formulas.

\coth \text{[m]} \text{ (5.3.8) - 128}
Command to produce the function name 'coth' in formulas.

\csc \text{[m]} \text{ (5.3.8) - 128}
Command to produce the function name 'csc' in formulas.

\cup \text{[m]} \text{ produces } \cup \text{ (5.3.3) - 125}
\text{\CurrentOption} \text{[p]} \text{ (D.2.3) - 443}
A command that may only be used in the definition of options, especially for default options. It contains the name of the option being processed.
H.1. Brief description of the \LaTeX\ commands

\{x\} .......................................................... (2.5.7) – 24

Produces a ‘dot under’ accent: \{o\} = o.

\dag produces † .................................................. (2.5.5) – 23
\dagger [m] produces † ........................................... (5.3.3) – 125
\dashbox{dash}(x\dimen,y\dimen)[pos]{text} ........ (13.1.4) – 291

Picture element command to produce a dashed frame with width x\dimen and height y\dimen, using a dash length of dash in the picture environment. Without the optional pos, the contents text are centered within the frame, otherwise they are positioned at the left (l), right (r), top (t), or bottom (b), or a combination thereof, such as lt. This command is used as an argument in a \put or \multiput command.

\dashv [m] produces ⊣ ........................................... (5.3.6) – 127
\date\{date\text\} ............................................. (3.3.1), (16.1) – 52, 353

1. The command \maketitle normally prints the current date on the title page. The declaration \date will replace the date with whatever text is given in date\text.

2. Prints the text date\text instead of the automatic current date in a letter.

\datelanguage .................................................. (11.1) – 255

A command used in several language adaptations to redefine the \today command according to the requirements of language. It occurs in german package as well as in the babel system. It may also be used for ‘dialects’, such a \dateUSenglish and \dateenglish. This command is normally part of the definition of the \selectlanguage command.

\dblfigrule ...................................................... (7.3) – 173

A command that is executed after a two-column float at the top of a page. It is normally defined to do nothing, but may be redefined to add a rule between the float and the main text. It must not add any net vertical spacing.

\renewcommand{\dblfigrule}{\vspace*{-0.4pt}
\rule{\textwidth}{0.4pt}}

\dblfloatpagefraction ...................................... (7.3) – 172

For two-column page formatting, the fraction of a float page that must be filled with floats before a new page is called. A new value is assigned with

\renewcommand{\dblfloatpagefraction}{decimal\frac}

\dbinom\{over\}\{under\} [m][a] .......................... (12.2.3) – 264

With the amsmath package, produces a binomial as \binom does, but in \displaystyle size.
\texttt{dblfloatsep} \hspace{1cm} \text{(7.3) - 172}

For two-column page formatting, the vertical spacing between floats that extend over both columns. A new value is set with the \texttt{setlength} command:

\begin{verbatim}
setlength{\dblfloatsep}{12pt plus 2pt minus 4pt}
\end{verbatim}

\texttt{dbltextfloatsep} \hspace{1cm} \text{(7.3) - 172}

For two-column page formatting, the vertical spacing between floats extending over both columns at the top of the page and the following text. A new value is set with the \texttt{setlength} command.

\texttt{dbltopfraction} \hspace{1cm} \text{(7.3) - 172}

For two-column page formatting, the maximum fraction of a page that may be occupied at the top by floats extending over both columns. A new value is assigned with

\begin{verbatim}
\renewcommand{\dbltopfraction}{decimal.fracl}
\end{verbatim}

\texttt{dbltopnumber} \hspace{1cm} \text{(7.3) - 172}

For two-column page formatting, the maximum number of floats that may appear at the top of a page extending over both columns. A new value is assigned with

\begin{verbatim}
\setcounter{dbltopnumber}{num}
\end{verbatim}

\texttt{ddag} produces \textasteriskcentered\hspace{1cm} \text{(2.5.5) - 23}

\texttt{ddagger [m]} produces \textasteriskcentered\hspace{1cm} \text{(5.3.3) - 125}

\texttt{dddot{\textit{x}} [m][a]} \hspace{1cm} \text{(12.2.2) - 263}

With the amsmath package, a triple dot accent in math formulas:

\begin{verbatim}
\dddot{a}
\end{verbatim}

\texttt{dddot{\textit{x}} [m][a]} \hspace{1cm} \text{(12.2.2) - 263}

With the amsmath package, a four-dot accent in math formulas:

\begin{verbatim}
\ddddot{a}
\end{verbatim}

\texttt{ddot{\textit{x}} [m]} \hspace{1cm} \text{(5.3.9) - 129}

A double dot accent in mathematical formulas: \texttt{\ddot{a}}.

\texttt{Ddot{\textit{x}} [m][a]} \hspace{1cm} \text{(12.2.2) - 263}

With the amsmath package, can be used like \texttt{\ddot{a}}, but with multiple \texttt{\textit{AMS-La}}\textsc{teX} math accents the positioning will be correct.

\texttt{ddots [m]} produces \hspace{1cm} \text{(5.2.6) - 123}

\texttt{\DeclareErrorFont{\textit{code}}{\textit{fam}}{\textit{ser}}{\textit{sfp}}{\textit{sz}} [p]} \hspace{1cm} \text{(A.3.6) - 376}

If no valid font can be found, even after substituting the attributes as given by \texttt{\DeclareFontSubstitution}, the font declared with this command is selected as a last resort.
H.1. Brief description of the \LaTeX\ commands

\DeclareFixedFont{\cmd}{\code}{\fam}{\ser}{\shp}{\sz} [p]

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . (A.3.2) – 373

Defines \cmd to be a font declaration that selects a font with the
fixed attributes given in the definition.

\DeclareFontEncoding{\code}{\text.set}{\math.set} [p] . (A.3.6) – 376

Declares \code to be the name of a new encoding scheme; \text.set is a
set of commands that is to be executed when switching to text mode,
\math.set a set for math mode.

\DeclareFontEncodingDefaults{\text.set}{\math.set} [p] (A.3.6) – 376

Declares the sets of commands to be executed by all encodings when
switching to text and math modes; the additional commands specific
to each encoding are executed afterwards.

\DeclareFontFamily{\code}{\fam}{\opt} [p] . . . . . . (A.3.6) – 376

Declares \fam to be a new font family with the encoding \code; the
encoding must previously have been declared. The commands in \opt
are executed every time this family is selected.

\DeclareFontShape{\code}{\fam}{\ser}{\shp}{\def}{\opt} [p]

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . (A.3.6) – 377

Associates external font names with the given font attribute values;
the actual definition \def relates font sizes to font names, as explained
on page 377. The commands in \opt are executed every time this shape
is selected.

\DeclareFontSubstitution{\code}{\fam}{\ser}{\shp} [p] (A.3.6) – 376

Declares the font attributes that are to be substituted in case there
is no valid font corresponding to the set of attributes selected. The
order of substitution is shape, series, and family; the encoding is
never substituted.

\DeclareGraphicsExtensions{\ext.list} [p] . . . . . . . (6.1.7) – 165

Establishes the list of default extensions for graphics files that can be
imported with the \includegraphics command and the graphics
or graphicx packages; \ext.list is a comma-separated list of file ex-
tensions, such as .eps,.ps.

\DeclareGraphicsRule{\ext}{\type}{\bb}{\cmd} [p] . . . (6.1.7) – 165

Associates a graphics file extension \ext with a graphics file type and a
file extension (\bb) where the bounding box information is to be read,
and an operating command (\cmd) that is to be executed on the file
to make it available for importation. For example

\DeclareGraphicsRule{.eps.gz}{eps} {\eps.bb}{'gunzip -c #1}

Here the command must be prefixed with ‘ and #1 represents the
name of the file to be processed.
\DeclareMathAccent{\cmd}{\type}{\symfnt}{\pos} [p] . (A.3.4) – 375
Declares \cmd to be a math accent command, printed with the character in position \pos of the symbol font with the internal name \symfnt. The \type is either \mathord or \mathalpha; in the latter case the symbol changes with math alphabet.

\DeclareMathAlphabet{\cmd}{\code}{\fam}{\ser}{\shp} [p] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (A.3.3) – 373
Defines \cmd to be a math alphabet for setting letters in math mode; the font selected in all math versions is that with the specified font attributes. If a different font is to be invoked for certain math versions, they are defined individually with \SetMathAlphabet afterwards. However, if the shape attribute is left empty, the alphabet command is indeed created, but remains undefined in all versions, requiring an explicit \SetMathAlphabet declaration for each one.

\DeclareMathDelimiter{\cmd}{\type}{\symfnt1}{\pos1} {\symfnt2}{\pos2} [p] . . . . . . . . . . . . . . (A.3.4) – 375
Declares \cmd to be a math delimiter in two sizes: the smaller variant is the character in position \pos1 of the symbol font with the internal name \symfnt1, while the larger is from position \pos2 of font \symfnt2.

\DeclareMathOperator{\cmd}{\name} [p][a] . . . . (12.2.5) – 268
With the amsopn or amsmath packages, defines \cmd to be a math mode command to print the text \name as a function name in an upright font with appropriate spacing. With the *-form, the raising and lowering operators \^ and _ produce limits, above or below the name.

\DeclareMathRadical{\cmd}{\symfnt1}{\pos1} {\symfnt2}{\pos2} [p] . . . . . . . . . . . . . . (A.3.4) – 375
Declares \cmd to be a math radical symbol: the smaller variant is the character in position \pos1 of the symbol font with the internal name \symfnt1, while the larger is from position \pos2 of font \symfnt2.

\DeclareMathSizes{\text}{\math_text}{\script}{\sscript} [p] (A.3.4) – 375
Sets the point sizes for the three math styles \textstyle, \scriptstyle, and \sscriptstyle when the text is being printed in point size \text. The unit pt is not included.

\DeclareMathSymbol{\symbol}{\type}{\symfnt}{\pos} [p] (A.3.4) – 375
Declares \symbol to be a command that prints the character in position \pos with the symbol font with the internal name \symfnt. This same symbol is printed in all math alphabets, but may be different for other math versions if the symbol font has been \Set... to have different attributes in that math version.
H.1. Brief description of the \LaTeX{} commands

\DeclareMathVersion{\text{ver}}[p] \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots (A.3.3) - 374

Declares \text{ver} to be a new math version for math alphabets and symbol fonts. Initially, the version will use those fonts defined by \texttt{\Declare...} commands, which may be redefined for this math version with appropriate \texttt{\Set...} commands.

\DeclareOldFontCommand{\text{cmd}}{\text{text specs}}{\math{math specs}}[p] \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots (A.3.2) - 373

Defines \text{cmd} to be a font declaration that invokes \text{text specs} in text mode, and \text{math specs} in math mode. The new command behaves in math mode as a 2.09 font declaration, although \text{math specs} must be a math font command without its argument. It is meant to define commands to be compatible with \LaTeX{} 2.09 and should generally be avoided.

\DeclareOption{\text{option}}{\text{code}}[p] \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots (D.2.3) - 442

In a class or package file, this command defines the set of commands \text{(code)} that is to be associated with the given \text{option}. These commands are executed when \texttt{\ExecuteOptions} or \texttt{\ProcessOptions} is called. After the latter, all definitions are erased, to save memory. The \text{code} is internally stored in a command named \texttt{\ds@option}.

\DeclareOption*{\text{code}}[p] \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots (D.2.3) - 442

In a class or package file, this command defines the default set of commands that is associated with every undefined option. Special commands that may be used within \text{code} are \texttt{\CurrentOption} (the name of the option) and \texttt{\OptionNotUsed}. Example:

\texttt{\DeclareOption*{\InputIfFileExists}{\CurrentOption.sty}{}{\OptionNotUsed}}

\DeclareRobustCommand{\text{cmd}}[\text{narg}][\text{opt}]{\text{def}} ... (D.2.5) - 445

Defines or redefines the command \text{\texttt{\cmd}} in the same way as \texttt{\newcommand} except that the result is robust: it may be used in the argument of another command without a \texttt{\protect} command before it.

\DeclareRobustCommand*{\text{cmd}}[\text{narg}][\text{opt}]{\text{def}} ... (D.2.6) - 445

The same as \texttt{\DeclareRobustCommand} except that the arguments to \texttt{\cmd} must be ‘short’, not containing any new paragraphs.

\DeclareSymbolFont{\text{sym_fnt}}{\text{code}}{\text{fam}}{\text{ser}}{\text{shp}}[p] \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots (A.3.4) - 374

Declares the math font with the given attributes to be a symbol font that may be addressed by other commands with the name \text{sym_fnt}. The symbol font applies to all math versions unless redefined with a \texttt{\SetSymbolFont} command.
Appendix H. Command Summary

\DeclareSymbolFontAlphabet{\cmd}{sym_fnt} [p] . . (A.3.4) – 375
Defines \cmd to be a math alphabet that uses the font declared with a \DeclareSymbolFont command to have the internal name sym_fnt. This is preferred over \DeclareMathAlphabet if there is a defined symbol font with the necessary attributes for the math alphabet.

\DeclareTextAccent{\cmd}{code}{pos} [p] . . . . . . . (A.3.7) – 379
Defines \cmd to be an accent command when encoding code is active; it uses the symbol in font position pos as the accent.

\DeclareTextAccentDefault{\cmd}{code} [p] . . . . . . . (A.3.7) – 380
Declares the default encoding that is to be taken if the accent command \cmd is called in an encoding for which it is not explicitly defined.

\DeclareTextCommand{\cmd}{code} [narg][opt]{def} [p]
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . (A.3.7) – 379
Defines \cmd in the same way as \newcommand except the definition is only valid when encoding code is active.

\DeclareTextCommandDefault{\cmd}[narg][opt]{def} [p]
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . (A.3.7) – 380
Creates a default definition for the command \cmd for all encodings for which it is not explicitly defined.

\DeclareTextComposite{\cmd}{code}{let}{pos} [p] . (A.3.7) – 379
Defines \cmd followed by the letter let to be the single character in font position pos when encoding code is active. It thus defines the action of accent commands in T1 encoding when the accented letter exists as a separate symbol. For example:
\DeclareTextComposite{'}{T1}{e}{233}
The command must already have been defined for the encoding with either \DeclareTextAccent or \DeclareTextCommand (with one argument).

\DeclareTextCompositeCommand{\cmd}{code}{let}{def} [p]
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . (A.3.7) – 379
Is the same as \DeclareTextComposite except that any definition may be assigned to the command/letter combination, and not just a single symbol.

\DeclareTextFontCommand{\cmd}{font specs} [p] . . (A.3.2) – 373
Defines \cmd to be a text font command that sets its argument with the font_spec specifications. Example:
\DeclareTextFontCommand{\textbf}{\textbf}{\bfseries}

\DeclareTextSymbol{\cmd}{code}{pos} [p] . . . . . (A.3.7) – 379
Defines \cmd to print the symbol in font position pos when encoding code is active.
H.1. Brief description of the \LaTeX\ commands 537

\DeclareTextSymbolDefault{\cmd}{code}{[p]} (A.3.7) – 380

Declares the default encoding that is to be taken if the symbol command \cmd is called in an encoding for which it is not explicitly defined.

\definecolor{\name}{\model}{\specs} (6.2) – 167

A command made available with the color package. It associates the name of a color (\name) with the specifications \specs according to a certain \model. Possible values for \model are rgb (red, green, blue), cmyk (cyan, magenta, yellow, black), gray, and named. In each case, \specs is a comma-separated list of numbers between 0 and 1 specifying the strength of the relevant component. In the case of the named model, \specs is an internal name for the color that is known by the driver program. Examples:

\definecolor{litegrn}{cmyk}{0.25,0,0.75,0}
\definecolor{brown}{named}{RawSienna}

A number of colors are predefined in all color drivers: red, green, blue, yellow, cyan, magenta, black, and white.

\deg [m] (5.3.8) – 128

Command to produce the function name 'deg' in formulas.

\DeleteShortVerb{\c} (4.9.1, B.5.3) – 111, 393

When the standard package shortvrb has been loaded, this command counteracts the effects of a previous \MakeShortVerb{\c}, allowing the character c to have its original meaning once more.

\Delta [m] produces $\Delta$ (5.3.1) – 125
\delta [m] produces $\delta$ (5.3.1) – 125
\depth (4.7.1) – 86

A length parameter equal to the depth of a box (baseline to bottom); it may only be used in the width specification of \makebox, \framebox, or \savebox, or in the height specification of a \parbox or minipage environment.

\framebox[20\depth]{text}

\det [m] (5.3.8) – 128

Command to produce the function name 'det' in formulas. Can be combined with a lower limit by means of the subscript command.

\dfrac{\numerator}{\denominator} [m][a] (12.2.3) – 264

With the amsmath package, produces a fraction as \frac does, but in displaystyle size.

\DH (G.4.5) – 503

When T1 encoding is active, prints the character D.
\dh . . . . . . . . . . . . . . . . . . . . . . . . . . (G.4.5) – 503

When T1 encoding is active, prints the character δ.

\Diamond [m] produces ◊ . . . . . . . . . . . . . . . . . . . (5.3.3) – 125
\diamond [m] produces ◇ . . . . . . . . . . . . . . . . . . . (5.3.3) – 125
\diamondsuit [m] produces ♦ . . . . . . . . . . . . . . . . . . (5.3.6) – 127
\dim [m] . . . . . . . . . . . . . . . . . . . . . . . . . . (5.3.8) – 128

Command to produce the function name ‘dim’ in formulas.

\discretionary{before}{after}{without} . . . . . . . (2.8.1) – 35

Hyphenation suggestion within a word. The word may be divided such that before is at the end of one line, and after at the start of the next line. If no division occurs, without is printed.

\displaybreak[num] [m][a] . . . . . . . . . . . . . (12.2.7) – 278

With the amsmath package, allows a manual page break in a multiline math formula when given just before \; if num is present, it takes on values of 0–4 with increasing encouragement for a break, where 4 is the same as no value, a forced page break. Automatic page breaks are impossible in multiline formulas unless \allowdisplaybreaks has been issued in the preamble.

\displaystyle [m] . . . . . . . . . . . . . . . . . . . . . . (5.5.2) – 146

Switches to font size \displaystyle as the active font within a math formula.

\div [m] produces ÷ . . . . . . . . . . . . . . . . . . . . . . . . . (5.3.3) – 125
\DJ . . . . . . . . . . . . . . . . . . . . . . . . . . (G.4.5) – 503

When T1 encoding is active, prints the character D.

\dj . . . . . . . . . . . . . . . . . . . . . . . . . . (G.4.5) – 503

When T1 encoding is active, prints the character d.

\documentclass[options]{class}[version] [p] . . . . . . . (3.1) – 37

Normally the first command in a \LaTeX{} document, determining the overall characteristics. Standard values for class are:

- article, report, book, letter, and slides

of which only one may be selected. In addition, various options may be chosen, their names separated by commas. Possibilities are:

- 10pt, 11pt, 12pt,
- letterpaper, legalpaper, executivepaper,
- a4paper, a5paper, b5paper, landscape,
- onecolumn, twocolumn,
- oneside, twoside,
- notitlepage, titlepage,
- leqno, fleqn, openbib,
- draft, final
These and any additional options are all global, meaning that they also apply to any packages specified with a following `\usepackage` command.

The optional `version` is a date, given in the form `yyyy/mm/dd`, as for example `1994/08/01`. If the date of the class file read in is earlier than this, a warning message is printed.

`\doublebox{text}` ........................................ (4.7.9) – 94

With the `fancybox` package, is a variant of `\fbox`, drawing a doubled box around `text`; the thicknesses and separation of the lines depend on the length `\fboxrule`.

`\dot{x} [m]` ........................................... (5.3.9) – 129

A dot accent in mathematical formulas: `\dot{a} = \dot{a}`.

`\Dot{x} [m][a]` ........................................ (12.2.2) – 263

With the `amsmath` package, can be used like `\dot`, but with multiple `\A\M\S-\L\T\E\X` math accents the positioning will be correct.

`\doteq [m]` produces `\doteq` ................................ (5.3.4) – 126

`\dotfill` ................................................. (2.7.1) – 30

Fills up the space in a line with a dotted leader: `\dotfill`.

`\dots` produces `\dots` ................................... (5.2.6) – 123

With the `amsmath` package, places continuation dots in math mode automatically at a height determined by the following symbol.

`\dotsb [m][a]` produces `\dotsb` ................................ (12.2.2) – 264

`\dotsc [m][a]` produces `\dotsc` ................................... (12.2.2) – 264

`\dotsi [m][a]` produces `\dotsi` ................................... (12.2.2) – 264

`\dotsm [m][a]` produces `\dotsm` ................................... (12.2.2) – 264

`\doublerulesep` ........................................... (4.8.2) – 98

The distance between double rules inside the `tabular` and `array` environments. New value assigned with `\setlength` outside of the environment.

`\Downarrow [m]` produces `\Downarrow` .................. (5.3.5) – 127

`\downarrow [m]` produces `\downarrow` .................... (5.3.5) – 127

`\ell [m]` produces `\ell` ...................................... (5.3.6) – 127

`\em` .......................................................... (4.1.1) – 62

This declaration switches to an emphatic font, one that has the current family and series, but with a different shape attribute. It normally toggles between an upright and an italic shape.
\textbf{Appendix H. Command Summary}

\texttt{\textbackslash emph\{text\}} \hspace{1cm} (4.1.1) - 62
This command sets its argument in an emphatic font, one that has the current family and series, but with a different shape attribute. It normally toggles between an upright and an italic shape.

\texttt{\textbackslash emptyset\{m\}} produces $\emptyset$ \hspace{1cm} (5.3.6) - 127

\texttt{\textbackslash encl\{enclosures\}} \hspace{1cm} (16.1) - 353
Command in the document class \texttt{letter} to add the word ‘encl:’ with the list \texttt{enclosures} at the end of a letter. For International \LaTeX, the actual word printed is contained in the command \texttt{\enclname}.

\texttt{\enclname} \hspace{1cm} (D.4.1) - 460
Command in the \texttt{letter} document class containing the word to be printed by the \texttt{\encl} command. In English, this is ‘encl’ but may be altered for adaptation to other languages.

\texttt{\textbackslash end\{environment\}} \hspace{1cm} (2.2) - 19
Command to terminate an environment started with a command \texttt{\begin\{environment\}}.

\texttt{\textbackslash endfirsthead} \hspace{1cm} (4.8.4) - 108
Within the \texttt{longtable} environment of the \texttt{longtable} package, this command ends those specifications that are to be added to the top of the table (the head) on the first page only.

\texttt{\textbackslash endfoot} \hspace{1cm} (4.8.4) - 108
Within the \texttt{longtable} environment of the \texttt{longtable} package, this command ends those specifications that are to be added at the bottom of the table (the foot) before it is continued to other pages.

\texttt{\textbackslash endhead} \hspace{1cm} (4.8.4) - 108
Within the \texttt{longtable} environment of the \texttt{longtable} package, this command ends those specifications that are to be added to the top of the table (the head) when it is continued to other pages.

\texttt{\textbackslash endlastfoot} \hspace{1cm} (4.8.4) - 108
Within the \texttt{longtable} environment of the \texttt{longtable} package, this command ends those specifications that are to be added at the bottom of the table (the foot) on the last page.

\texttt{\enlargethispage\{size\}} \hspace{1cm} (2.7.4) - 34
The \texttt{\textbackslash texttheight} parameter is temporarily increased by the length \texttt{size}, in order to improve a bad page break. On the following pages, \texttt{\textbackslash texttheight} will have its normal value once more.

\texttt{\enlargethispage*} \hspace{1cm} (2.7.4) - 34
Is the same as \texttt{\enlargethispage} except that any additional spacing between the lines is removed as necessary to maximize the amount of text on the page.
H.1. Brief description of the \LaTeX\ commands

\texttt{\textbackslash ensuremath\{math cmds\}} \hfill (8.3.1) - 186

Sets \textit{math cmds} in math mode; may be called in both text and math modes. Its main use is to define new commands that require math mode but can be called from either mode.

\texttt{\textbackslash epsilon\ [m]} produces $\epsilon$ \hfill (5.3.1) - 125

\texttt{\textbackslash eqref\{marker\} \[a\]} \hfill (12.2.7) - 278

With the amsmath package, is a variation on the \texttt{\textbackslash ref} command, and prints the equation number defined with \texttt{\textbackslash label\{marker\}} in parentheses, as (5.6).

\texttt{\textbackslash equiv\ [m]} produces $\equiv$ \hfill (5.3.4) - 126

\texttt{\textbackslash eta\ [m]} produces $\eta$ \hfill (5.3.1) - 125

\texttt{\textbackslash euro} \hfill (2.5.8) - 25

With the eurosym package, prints the euro symbol $\epsilon$ from the eurosym METAFONT fonts. With the eurosans package, it prints the symbol from the Adobe (PostScript) euro fonts. In both cases, the symbol is sans serif but changes to bold face or slanted to match the current font.

\texttt{\textbackslash EUR} \hfill (2.5.8) - 25

With the europs package, prints the euro symbol $\epsilon$ from the Adobe (PostScript) euro fonts, such that it matches the font family and other attributes. \texttt{\textbackslash EURofc} prints the invariable symbol $\epsilon$.

\texttt{\textbackslash evensidemargin} \hfill (3.2.5) - 48

Sets the left margin for the even-numbered pages. It is effective in the document class \texttt{book} and, when the option \texttt{twoside} has been selected, in the other classes. A new value is assigned with the \texttt{\setlength} command:

\texttt{\setlength\{\evensidemargin\}\{2.5cm\}}

\texttt{\textbackslash ExecuteOptions\{option\_list\} \[p\]} \hfill (D.2.3) - 443

In a class or package file, this command executes all the option definitions in \texttt{option\_list}. This is normally invoked just prior to \texttt{\ProcessOptions} to establish certain options as default.

\texttt{\textbackslash exists\ [m]} produces $\exists$ \hfill (5.3.6) - 127

\texttt{\textbackslash exp\ [m]} \hfill (5.3.8) - 128

Command to produce the function name 'exp' in formulas.

\texttt{\textbackslash extracolsep\{extra\_width\}} \hfill (4.8.1) - 96

Tabular command for setting extra spacing between all the following columns in a table. This command is inserted as an @-expression into the column definition field of the \texttt{tabular} environment:

\texttt{\begin\{tabular\}\{lr@{\extracolsep\{2.5mm\}}\}lcr\}}
Appendix H. Command Summary

\fbox{text} produces a frame around text . . . . . . . . . . . . . . (4.7.1) - 86
\fancypage{cmds1}{cmds2} . . . . . . . . . . . . . . . . . . . (4.7.9) - 95

With the fancybox package, places a framed box around the contents of all subsequent pages; cmds1 exclude the head and footlines, cmds2 includes them. The arguments set box parameters like \fboxrule but must end with a box command like \shadowbox. Usually one set of cmds is left blank.

\fboxrule . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (4.7.8) - 93
The line thickness for the frames drawn by \fbox and \framebox commands. A new value is assigned with the \setlength command:
\setlength{\fboxrule}{1pt}

\fboxsep . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (4.7.8) - 93
The distance between the frame and text in the \fbox and \framebox commands. A new value is assigned with the \setlength command:
\setlength{\fboxsep}{1mm}

\fcolorbox col_spec1 col_spec2{text} . . . . . . . . . . . (6.2) - 167
A command made available with the color package. Like \colorbox, the text is set in an LR box with the col_spec2 as the background color, but with a frame of color col_spec1 around it. The col_specs are either both defined names or employ the same model. Examples:
\fcolorbox[rgb]{1,0,0}{0,1,0}{Text}
\fcolorbox{red}{green}{Text}

\figurename . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (D.4.1) - 459
Command containing the name for a figure caption. In English, this is 'Figure' but may be altered for adaptation to other languages.

\fill . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (2.4.2) - 22
A rubber length with a natural size of zero that can stretch to any size necessary to fill up the horizontal or vertical space available.

\flat [m] produces b . . . . . . . . . . . . . . . . . . . . . . (5.3.6) - 127
\floatpagefraction . . . . . . . . . . . . . . . . . . . . . . (7.3) - 172
The fraction of a float page that must be filled with floats before a new page is called. A new value is assigned with
\renewcommand{\floatpagefraction}{\text{decimal\_frac}}

\floatsep . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (7.3) - 172
The vertical spacing between floats that appear at the top or bottom of a page. A new value is set with the \setlength command:
\setlength{\floatsep}{12pt plus 2pt minus 4pt}

\flushbottom . . . . . . . . . . . . . . . . . . . . . . . . . . . . (3.2.5) - 48
A declaration that puts vertical spacing between paragraphs so that the last line on every page is at the same position. Standard for the book document class and for the twoside option.
H.1. Brief description of the \LaTeX{} commands

\texttt{\fnsymbol{counter}} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (8.1.4) – 183

Prints the current value of the given \emph{counter} as a ‘footnote symbol’:
\[ * \dagger \ddagger \§ \| \ast \dagger \ddagger \]

\texttt{\fontencoding{enc}} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (A.1) – 368

This command selects the font encoding scheme. Possible values of \emph{enc} are \texttt{OT1} for the standard and \texttt{T1} for the Cork encodings. Other values are also possible.

\texttt{\fontfamily{fam}} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (A.1) – 368

This command selects the ‘family’ of fonts. Possible values of \emph{fam} for standard \LaTeX{} with the Computer Modern fonts are \texttt{cmr}, \texttt{cmss}, \texttt{cmtt}, and \texttt{cmfi}.

\texttt{\fontseries{ser}} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (A.1) – 368

This command selects the ‘series’ of fonts within a ‘family’. Possible values of \emph{ser} for standard \LaTeX{} are \texttt{m} (medium) and \texttt{bx} (bold extended).

\texttt{\fontshape{form}} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (A.1) – 368

This command selects the ‘shape’ of fonts. Possible values of \emph{form} are \texttt{n} (normal), \texttt{i}t (italic), \texttt{s}l (slanted), \texttt{s}c (small caps), and \texttt{u} (‘unslanted’ italic).

\texttt{\fontsize{sz}{line_sp}} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (A.1) – 368

This command selects the font size. The argument \emph{sz} specifies the size of the characters in points (without the dimension \texttt{pt}) and \emph{line_sp} determines the value of the interline spacing (\baselineskip), with an explicit dimension. Example: \texttt{\fontsize{12}{14pt}}.

\texttt{\footnote[\emph{num}]{\emph{footnote}\_text}} \ldots \ldots (4.10.1), (4.10.2) – 112, 114

Produces a footnote containing the text \emph{footnote}\_text. The optional argument \emph{num} will be used as the footnote number in place of the next number in the automatic sequence.

\texttt{\footnotemark[\emph{num}]} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (4.10.3) – 114

Produces a footnote marker in the current text. The optional argument \emph{num} will be used as the footnote number in place of the next number in the automatic sequence. May be used within structures where \texttt{\footnote} is not normally permitted, such as LR boxes, tables, math formulas.

\texttt{\footnoterule} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (4.10.6) – 118

This is an internal command to produce the horizontal rule between the regular text on a page and the footnote text at the bottom. May be changed with, e.g.,

\begin{verbatim}
\renewcommand{\footnoterule}
{\rule{width}{hght}\vspace{-hght}}
\end{verbatim}
Appendix H. Command Summary

\footnotesep \hspace{1cm} (4.10.6) - 117

The vertical spacing between two footnotes. A new value is assigned with the setlength command:
\setlength{\footnotesep}{6.5pt}

\footnotesize \hspace{1cm} (4.1.2) - 62

Switches to the font size \footnotesize, which is smaller than \small but larger than \scriptsize.

\footnotetext[num]{footnote_text} \hspace{1cm} (4.10.3) - 114

Produces a footnote with the text \textit{footnote_text} but without generating a marker in the current text. The marker that is used for the footnote itself at the bottom of the page derives from the current value of the counter footnote, which remains unchanged, or from the value of the optional argument \textit{num}. This command may be used together with the \footnotemark command to insert footnotes into structures where they are otherwise not allowed, such as LR boxes, tables, and math formulas. The \footnotetext command must be given outside of that structure.

\footskip \hspace{1cm} (3.2.5) - 48

The distance from the bottom edge of the text body to the lower edge of the footline. A value is assigned with the \setlength command:
\setlength{\footskip}{25pt}

\forall \hspace{1cm} (5.3.6) - 127

Changes \textit{m} produces \( \forall \).

\foreignlanguage{language}{text} \hspace{1cm} (11.1) - 254

In the babel system, sets a short text in the selected language.

\frac{numerator}{denominator} \hspace{1cm} (5.2.3) - 122

Math command for generating fractions.

\frame{text} \hspace{1cm} (13.1.4) - 298

Produces a frame without any intervening spacing around \textit{m}. Mainly used as a picture element in a \put or \multiput command within the picture environment.

\framebox[width][pos]{text} \hspace{1cm} (4.7.1) - 86

Produces a frame of width \textit{width} around \textit{text}. By default, the text is centered within the frame, but may be left or right justified by giving the optional argument \textit{pos} as \texttt{l} or \texttt{r}. It may also have the value \texttt{s}, to stretch the text to the given width.

\framebox(x_dimen,y_dimen)[pos]{text} \hspace{1cm} (13.1.4) - 291

Picture element command to produce a frame of width \textit{x_dimen} and height \textit{y_dimen} within the picture environment. Without the optional argument \textit{pos}, the text is centered vertically and horizontally.
The text may be left or right justified, and/or aligned at the top or bottom, by setting `pos` to a combination of the letters `l`, `r`, `t`, and `b`, such as `tr` for top, right; `pos` may also contain `s` to stretch the text to the full width. The command is to be used as the argument of a \put or \multiput command.

\frenchspacing ...................................................(2.7.1) - 29

After this command has been given, no additional horizontal spacing is inserted at the end of a sentence. The countercommand is \nonfrenchspacing.

\frontmatter ...................................................................(3.3.5) - 57

In the book class, introduces the material that comes at the beginning (preface, table of contents) by turning off the chapter numbering of the \chapter command and switching to Roman numbers for the pagination.

\frown [m] produces \(\frown\) ...........................................(5.3.4) - 126
\fussy ...........................................................................(2.8.3) - 36

Countercommand of \sloppy that allows larger interword spacings than normal. After \fussy has been given, the normal spacings apply once more.

\Gamma [m] produces \(\Gamma\) .............................................(5.3.1) - 125
\gamma [m] produces \(\gamma\) .............................................(5.3.1) - 125
\gcd [m] .........................................................................(5.3.8) - 128

Command to produce the function name 'gcd' in formulas. A lower limit may be given as a subscript.

\ge [m] produces \(\ge\) .....................................................(5.3.4) - 126
\genfrac{left}{right}{thkns}{mathsz}{over}{under} [m][a]
...................................................................................(12.2.3) - 265

With the amsmath package, produces a generalized fraction with delimiters `left` and `right`, line thickness `thkns`, math font size `mathsz` 0-3, and with `over` on top of `under`. If `mathsz` is empty, the sizing is automatic.

\geq [m] produces \(\geq\) ..................................................(5.3.4) - 126
\gets [m] produces \(\gets\) ..................................................(5.3.5) - 127
\gg [m] produces \(\gg\) ....................................................(5.3.4) - 126
\glossary{glossary.entry} ..............................................(9.4.4) - 230

Write a \glossaryentry command to the .glo file if \makeglossary has been issued in the preamble; else it does nothing.

\glossaryentry{glossary.entry}{page_number} ...........(9.4.4) - 230

The form in which the entry is written to the .glo file by the \glossary command.
A command added with the graphpap package for use in the picture environment. It plots a labeled grid system with the lower left corner at \((x, y)\), \(lx\) wide and \(ly\) high. Grid lines are drawn every \(num\) units, with the fifth ones thicker. If \(num\) is not specified, it is assumed to be 10. All arguments must be integers, not decimal fractions.

A grave accent over the math variable \(x\): \(\text{\textbackslash grave}\{x\}\) = \(\grave{a}\).

With the amsmath package, can be used like \(\text{\textbackslash grave}\), but with multiple AMS-L\(\LaTeX\) math accents the positioning will be correct.

When \(T1\) encoding is active, prints the symbol «.

When \(T1\) encoding is active, prints the symbol ».

When \(T1\) encoding is active, prints the symbol ‹.

When \(T1\) encoding is active, prints the symbol ›.

Hungarian double acute accent: \(\text{\textbackslash H}\{x\}\) = ˝o.

Circumflex over the math variable \(x\): \(\text{\textbackslash hat}\{x\}\) = ˆa.

With the amsmath package, can be used like \(\text{\textbackslash hat}\), but with multiple AMS-L\(\LaTeX\) math accents the positioning will be correct.

\(\text{\textbackslash hbar}\) produces \(\hbar\).

The height of the head at the top of each page. A new value is assigned with the \(\text{\textbackslash setlength}\) command:
\[
\text{\textbackslash setlength}\{\text{\textbackslash headheight}\}\{25\text{pt}\}
\]

Vertical spacing between the lower edge of the page head and the top of the main text. A new value is assigned with the \(\text{\textbackslash setlength}\) command:
\[
\text{\textbackslash setlength}\{\text{\textbackslash headsep}\}\{0.25\text{in}\}\]
H.1. Brief description of the \LaTeX\ commands 547

\headtoname ........................................... (D.4.1) - 460
Command in the letter document class containing the text that precedes the recipient's name in the headline after the first page. In English, this is 'To' but may be altered for adaptation to other languages.

\heartsuit [m] produces ♥ ................................. (5.3.6) - 127
\height .................................................. (4.7.1) - 86
A length parameter equal to the natural height of a box (distance from the baseline to the top); it may only be used in the \texttt{width} specification of \texttt{makebox}, \texttt{framebox}, or \texttt{savebox}, or in the \texttt{height} specification of a \texttt{parbox} or a \texttt{minipage} environment.
   \framebox[6\height]{text}

\hfill ..................................................... (2.7.1) - 30
A horizontal rubber spacing with a natural length of zero that can be stretched to any value. Used to fill up a horizontal line with blank spacing. This command is an abbreviation for \texttt{hspace{\fill}}.

\hline ..................................................... (4.8.1) - 97
Produces a horizontal line in the \texttt{array} and \texttt{tabular} environments over the width of the entire table.

\hoffset .................................................. 602, 603
Horizontal offset of the output page from the printer border set by the printer driver. This printer border is normally 1 inch from the left edge of the paper. The standard value of \hoffset is 0 pt so that the left reference margin of the page is identical with the printer margin. A new value is assigned with the \texttt{setlength} command:
   \setlength{\hoffset}{-1in}

\hom [m] ................................................... (5.3.8) - 128
Command to produce the function name 'hom' in formulas.

\hookleftarrow [m] produces ← ................................. (5.3.5) - 127
\hookrightarrow [m] produces ⟵ ................................. (5.3.5) - 127
\hrulefill .................................................. (2.7.1) - 30
Fills up the space in a line with a rule: \phantom{alphabet} = \hrulefill.

\hspace{\textit{width}} ........................................ (2.7.1) - 29
Produces horizontal spacing of length \textit{width}. It is ignored if it occurs at the beginning or end of a line.

\hspace*[\textit{width}] ........................................ (2.7.1) - 29
Produces horizontal spacing of length \textit{width} even at the beginning or end of a line.
Appendix H. Command Summary

\huge \hspace{2cm} \text{(4.1.2) – 62}

Switches to the font size \huge, which is smaller than \Huge but larger than \LARGE.

\Huge \hspace{2cm} \text{(4.1.2) – 62}

Switches to the largest font size available \Huge, which is larger than \huge.

\hyperlink{name}{link} \hspace{2cm} \text{(10.2.4) – 247}

With the \hyperref package, makes the contents of \textit{link} to be a link to the target with the name \textit{name}, established with the \hyperlink\command command. The \textit{link} may be text, a symbol, or a graphics loaded with \includegraphics. The link is framed with a colored box, unless the option colorlinks has been set to \textit{true}, in which case the \textit{link} is set in a text color determined by the option linkcolor.

\hypertarget{name}{text} \hspace{2cm} \text{(10.2.4) – 247}

With the \hyperref package, prints the \textit{text} argument as normal, but makes it a target for internal links, named \textit{name}. The \textit{text} may be empty.

\hypersetup{key = value, \ldots} \hspace{2cm} \text{(10.2.4) – 241}

With the \hyperref package, allows parameters (\textit{key}) to be assigned values (\textit{value}); the parameters may also be set as options in the \usepackage loading command, by with \hypersetup\command, these values may be changed within the document. The list of possible keys and values is given on pages 242–246.

\hyphenation{hyphenation list} \hspace{2cm} \text{(2.8.2) – 35}

Sets up a list of hyphenation exceptions. The \textit{hyphenation list} consists of a collection of words containing hyphens at the places where word division may occur: hy-phen-a-tion per-mit-ted.

\i \hspace{2cm} \text{(2.5.7) – 24}
\idotsint \hspace{2cm} \text{(12.2.2) – 260}
\iff \hspace{2cm} \text{(5.3.5) – 127}
\IfFileExists{file name}{true}{false} \hspace{2cm} \text{(D.2.8) – 447}

Tests if the file \textit{file name} can be found in the places where \LaTeX looks for files; if so, the code \textit{true} is executed; otherwise, \textit{false}. This is like \texttt{InputIfFileExists} except the file is not input.

\iflanguage{language}{yes text}{no text} \hspace{2cm} \text{(11.1) – 254}

In the multilingual \texttt{babel} system, tests if \textit{language} is the currently selected language and, if so, executes \textit{yes text}, otherwise \textit{no text}. 
H.1. Brief description of the LATEX commands

549

. . . . . (8.3.5) – 193

\[ifthen]ifthenelse{test}{then text}{else text}

A conditional command available when the standard package ifthen
has been loaded. If the logical statement test evaluates to htruei
then then text is inserted, otherwise else text. The logical statement may be relational (two numbers with one of < = > between them), an even–odd test (\isodd{number}), a comparison
of two texts (\equal{text1}{text2}), a comparison of two lengths
(\lengthtest{length1 op length2}, op is one of < = >), or a test
of a boolean switch (\boolean{switch}). Switches are created with
\newboolean{switch} and set with \setboolean{switch}{value},
where value is true or false. Logical statements may be combined
with logical operators \and, \or, and \not, and grouped with \( and
\).

\Im [m] produces = . . . . .
\imath [m] produces ı . . .
\in [m] produces ∈ . . . . .
\include{file} . . . . . . .

.
.
.
.

.
.
.
.

.
.
.
.

.
.
.
.

.
.
.
.

.
.
.
.

.
.
.
.

.
.
.
.

.
.
.
.

.
.
.
.

.
.
.
.

.
.
.
.

.
.
.
.

.
.
.
.

(5.3.6)
(5.3.6)
(5.3.4)
(9.1.2)

–
–
–
–

127
127
126
209

Inserts the contents of the file with the root name file and extension
.tex into the current text at the point where the command appears.
A new page is always started! Together with \includeonly, this
command allows portions of the document to be processed as though
the rest of the text were present.

\includegraphics[llx,lly][urx,ury]{file name} . . . . . (6.1.2) – 155
A command made available with the graphics package that imports
external graphics stored in the file file name. The coordinates of the
bounding box are given by llx, lly (lower left corner) and by urx, ury
(upper right corner). It is the contents of this bounding box that
are used for further manipulation: scaling, rotating. It is also the
(manipulated) bounding box that is used to reserve space in the text;
any graphics that extend beyond the limits of this box will also be
printed, but overlapping any other material that may be beside it.
The bounding box coordinates may have units attached to them; the
default units are big points bp (72 per inch).
If the bounding box coordinates are omitted, the information is
obtained in some other manner, depending on the type of graphics
file. For an encapsulated PostScript file, this information is taken
from the graphics file itself.
If llx and lly are not specified, they are assumed to be 0. That is, if
only one set of optional coordinates are given, they refer to the upper
right corner.

\includegraphics*[llx,lly][urx,ury]{file name} . . . . (6.1.2) – 155
The same as \includegraphics except that any graphics extending
beyond the bounding box are not included: the figure is clipped.


With the \texttt{graphicx} package, this command has a different syntax in which the scaling, rotating, clipping are effected through key=value pairs, like width=7cm, angle=90, scale=.5.

Only those files whose names are in \texttt{file list}, separated by commas, will be read in by the \texttt{\include} commands. The \texttt{\include} commands for other file names are ignored. Nevertheless, all the auxiliary files are read in so that the page and section numbers will be correct, as are the cross-reference markers.

The first line of the next paragraph is to be indented.

Writes a \texttt{\indexentry} command to the .idx file if the \texttt{\makeindex} command has been issued in the preamble; otherwise it does nothing. The MaketIndex program (Section 9.4.3) can process this file if the entries are in the forms

\begin{verbatim}
\texttt{\index{main entry}}
\texttt{\index{main entry!sub entry}}
\texttt{\index{main entry!sub entry!sub sub entry}}
\end{verbatim}

making up a theindex environment with the entries alphabetically ordered and organized with \texttt{\item}, \texttt{\subitem}, and \texttt{\subsubitem} commands.

The form in which the entry is written to the .idx file by the \texttt{\index} command.

Command containing the heading for the index. In English, this is ‘Index’ but may be altered for adaptation to other languages.

Command within theindex environment to produce a blank line.

Command to produce the function name ‘inf’ in formulas. A lower limit may be set as a subscript.

When the package amsmath is loaded, this command inserts text as a left-justified line between lines of an equation, without affecting their horizontal alignment.
H.1. Brief description of the \LaTeX commands

\input{file} ........................................ (9.1.1) - 207
Inserts the contents of the file with the root name \texttt{file} and extension .tex into the current text at the point where the command appears. The file that is read in may also contain further \input commands.

\InputIfFileExists{file name}{true}{false} .......... (D.2.8) - 447
Tests if the file \texttt{file name} can be found in the places where \LaTeX looks for files; if so, the code \texttt{true} is executed and the file is input; otherwise, \texttt{false} is executed.

\int [m] produces \(\int\) ........................................ (5.2.5) - 123
\iint [m][a] produces \(\iint\) ............................... (12.2.2) - 260
\iiint [m][a] produces \(\iiint\) .......................... (12.2.2) - 260
\iiiint [m][a] produces \(\iiiint\) ....................... (12.2.2) - 260
\intextsep ............................................. (7.3) - 172
The vertical spacing between floats in the middle of a page and the surrounding text. A new value is assigned with the \setlength command:
\setlength{\intextsep}{10pt plus2pt minus3pt}

\invisible ............................................. (15.1.2) - 326
In \texttt{slides} class, a declaration that makes the following text be printed in ‘invisible ink’, that is, it takes up as much space as though it were there. It remains in effect until the end of the environment, or end of the curly braces, in which it was issued, or until \visible is given. It is used for making overlays.

\iota [m] produces \(\iota\) ..................................... (5.3.1) - 125
\itdefault ........................................... (A.3.1) - 372
This command defines the shape attribute that is selected with the \itshape command. It may be redefined with \renewcommand:
\renewcommand{\itdefault}{\it}

\item[\texttt{label}] ................................... (4.3), (4.4.1) - 69, 74
Produces a label and the start of an item text in a list environment. Without the optional argument, the label is generated according to the type of environment, for example numbers for the \texttt{enumerate} environment. The optional argument inserts \texttt{label} in place of this standard item label.

\item ............................................... (9.4.1) - 225
Produces a main entry in the \texttt{index} environment.

\itemindent ........................................... (4.4.2) - 77
The amount by which the label and the text of the first line after each \item is indented in a list environment. The standard value is 0 pt, but a new value may be assigned with the \setlength command:
\setlength{\itemindent}{1em}
Appendix H. Command Summary

\itemsep ............................... (4.4.2) - 75
The amount of vertical spacing in addition to \parskip that is inserted between the \item texts in a list environment. A new value may be assigned with the \setlength command:
\setlength{\itemsep}{2pt plus1pt minus1pt}

\itshape ................................. (4.1.3), (A.2) - 64, 371
This declaration switches to a font in the current family and series, but with the italic shape attribute.

\j produces j ............................ (2.5.7) - 24
\jmath [m] produces \( j \) ........................ (5.3.6) - 127
\Join [m] produces \( \Join \) ........................ (5.3.6) - 127
\jot ...................................... (5.5.4) - 149
The amount of vertical spacing between the formula lines of an eqnarray or eqnarray* environment. Standard value is 3 pt. A new value may be assigned with the \setlength command:
\setlength{\jot}{4.5pt}

\k{\( x \)} .................................. (G.4.5) - 503
When T1 encoding is active, prints the ogonek accent \( \k{A} \) = \( \acute{\text{A}} \).

\kappa [m] produces \( \kappa \) .......................... (5.3.1) - 125
\ker [m] ................................. (5.3.8) - 128
Command to produce the function name 'ker' in formulas.

\kill ..................................... (4.6.2) - 82
Removes the preceding sample line in a tabbing environment that was given only to set the tabs and not to be printed at this point.

\L produces Ł ............................ (2.5.6) - 24
\l produces ł .............................. (2.5.6) - 24
\label{\text{marker}} .......................... (9.2.1) - 213
Sets a marker in the text at this position with the name \text{marker}. It may be referred to either earlier or later in the document with the command \ref{\text{marker}} to output the counter that was then current, such as the section, equation, or figure number, or with the command \pageref{\text{marker}} to print the page number where the marker was set.

\labelenum n ............................ (4.3.5) - 73
A set of commands to produce the standard labels for the nesting levels of the enumerate environments, where \text{n} is one of i, ii, iii, or iv. For example,
\renewcommand{\labelenumii}{\text{arabic}{\text{enumii}.})}
changes the standard labels of the second-level enumerate environment to be 1.), 2.), etc.
\labelitem \labelitemi \labelitemii \labelitemiii \labelitemiv

A set of commands to produce the standard labels for the nesting levels of the itemize environments, where \textit{n} is one of \textit{i}, ii, iii, or iv. For example,
\begin{verbatim}
\renewcommand{\labelitemi}{$\Rightarrow$}
\end{verbatim}
changes the standard labels of the outermost itemize environment to $\Rightarrow$.

\labelsep

In a list environment, the distance between the label box and the list text. A new value is assigned with the \texttt{\setlength} command:
\begin{verbatim}
\setlength{\labelsep}{5pt}
\end{verbatim}

\labelwidth

In a list environment, the width of the box reserved for the label. A new value is assigned with the \texttt{\setlength} command:
\begin{verbatim}
\setlength{\labelwidth}{2.2cm}
\end{verbatim}

\texttt{\LaTeX} \texttt{\Lambda} \texttt{\lambda} \texttt{\langle} \texttt{\language}

In \TeX versions 3.0 and later, the set of hyphenation patterns number \texttt{num} is made active. The patterns must be previously loaded into the format file by an \texttt{initex} run in which \texttt{\language\{num\}} was given before those patterns were read in.

\large

Switches to the font size \texttt{\large}, which is smaller than \texttt{\Large} but larger than \texttt{\normalsize}.

\Large

Switches to the font size \texttt{\Large}, which is smaller than \texttt{\LARGE} but larger than \texttt{\large}.

\LARGE

Switches to the font size \texttt{\LARGE}, which is smaller than \texttt{\huge} but larger than \texttt{\Large}.
Appendix H. Command Summary

\leftlbrack [m] . . . . . . . . . . . . . (5.4.1) - 131
Adjusts the size of the bracket symbol \lbrack to fit the height of the formula between the \left ... \right pair. For example, \leftlbrack. If there is to be no matching bracket, the \left and \right commands must still be given to specify the part of the formula to be sized, but the missing bracket is given as a period (for example, \right.).

\Leftarrow [m] produces ⇐ . . . . . . . . (5.3.5) - 127
\leftarrow [m] produces ← . . . . . . . . (5.3.5) - 127
\leftharpoondown [m] produces ⇐ . . . . . . . . (5.3.5) - 127
\leftharpoonup [m] produces ↩ . . . . . . . . . . . (5.3.5) - 127
\lefteqn [m] . . . . . . . . . . . . . . . . . . . . (5.4.7) - 140
A command inside the eqnarray environment that outputs its argument as though it had zero width, thus having no effect on the column widths. It is used mainly for the first row of a multi-row formula.

\leftmargin . . . . . . . . . . . . . . . . . . . . . . . (4.4.2) - 76
In a list environment, the amount by which the left edge of the text is indented relative to the surrounding text. A new value is assigned with the setlength command. For nested list environments, different values for the indentation can be specified by adding i ... vi to the declaration name, such as
\setlength{\leftmarginiii}{0.5cm}
See also . . . . . . . . . . . . . . . . . . . . . . . (4.3.4) - 71.

\Lefttrightarrow [m] produces ⇔ . . . . . . . . (5.3.5) - 127
\lefttrightarrow [m] produces ← . . . . . . . . (5.3.5) - 127
\leftroot{shift} [m][a] . . . . . . . . . . . . (12.2.5) - 269
With the amsmath package, used in the index to a \sqrt command to shift it slightly to the left. The shift is a number specifying how many units to move it. Example:
\sqrt[\leftroot{-1}\uproot{3}\beta]{k}
\leq [m] produces ≤ . . . . . . . . . . . . . . . . . . . . (5.3.4) - 126
\lfloor [m] produces \lfloor . . . . . . . . . . . (5.4.1) - 132
\lg [m] . . . . . . . . . . . . . . . . . . . . . . . . . (5.3.8) - 128
Command to produce the function name 'lg' in formulas.
\lhd [m] produces < . . . . . . . . . . . . . . . . . . . (5.3.3) - 125
\lim [m] . . . . . . . . . . . . . . . . . . . . . . . . (5.3.8) - 128
Command to produce the function name 'lim' in formulas. A lower limit may be set as a subscript.
\liminf [m] . . . . . . . . . . . . . . . . . . . . . . . . (5.3.8) - 128
Command to produce the function name 'lim inf' in formulas. A lower limit may be set as a subscript.
H.1. Brief description of the \LaTeX\ commands

\verb|\limits[m]| \hspace{1cm} (5.2.5), (5.3.7) - 123, 128

Places the upper and lower limits above and below the appropriate symbols where these would normally go just after them.

\verb|\limsup[m]| \hspace{1cm} (5.3.8) - 128

Command to produce the function name ‘\textit{lim sup}’ in formulas. A lower limit may be set as a subscript.

\verb|\line(\Delta x,\Delta y){length}| \hspace{1cm} (13.1.4) - 293

A picture element command within a \textit{picture} environment for drawing horizontal and vertical lines of any length as well as slanted lines at a limited number of angles. For horizontal and vertical lines, the \textit{length} argument is the actual length in units of \texttt{\unitlength}. For slanted lines, \textit{length} is the length of the projection on to the x-axis (horizontal displacement). The slope is determined by the \texttt{(\Delta x,\Delta y)} arguments, which take on integral values such that \(-6 \leq \Delta x \leq 6\) and \(-6 \leq \Delta y \leq 6\). This command is the argument of a \texttt{\put} or \texttt{\multiput} command.

\verb|\linebreak[n]| \hspace{1cm} (2.7.2) - 31

A recommendation to break the line of text at this point such that it fills the horizontal space available (left and right justified). The urgency of the recommendation is given by the integral number \texttt{n} between 0 and 4, with the higher numbers meaning a stronger recommendation. A value of 4 is the same as the command without the optional argument and means an obligatory line break.

\verb|\linethickness{thickness}| \hspace{1cm} (13.1.4) - 300

Sets the thickness of the horizontal and vertical lines in the \textit{picture} environment. The argument \textit{thickness} is a length specification with units, for example, \texttt{1.2mm}.

\verb|\linewidth| \hspace{1cm} (3.2.5) - 47

A length that is set to the current text line width, whether in one column of a two-column page, in a minipage or parbox. This must never be changed, but is used when that width is needed, say to set the width of an included graphics with

\begin{verbatim}
\includegraphics[width=0.8\linewidth]{..}
\end{verbatim}

\verb|\listfigurename| \hspace{1cm} (D.4.1) - 459

Command containing the heading for the list of figures. In English, this is ‘List of Figures’ but may be altered for adaptation to other languages.

\verb|\listfiles[p]| \hspace{1cm} (9.1.1), (D.2.9) - 208, 447

When given in the preamble, causes a list of all files read in during the processing to be printed to the monitor and to the transcript file at the end of the run. The list includes version number, date, and any additional information entered with one of the \texttt{\Provides...} commands.
Appendix H. Command Summary

\listoffigures \hspace{1em} (3.4.4) - 59

Produces a list of figures with the entries from all the \caption commands in figure environments.

\listoftables \hspace{1em} (3.4.4) - 59

Produces a list of tables with the entries from all the \caption commands in table environments.

\listparindent \hspace{1em} (4.4.2) - 77

Depth of indentation for the first line of a paragraph inside a list environment. A new value may be assigned with the \setlength command:

\setlength{\listparindent}{1em}

\listtablename \hspace{1em} (D.4.1) - 459

Command containing the heading for the list of tables. In English, this is 'List of Tables' but may be altered for adaptation to other languages.

$\ll [m]$ produces \ll \hspace{1em} (5.3.4) - 126

$\ln [m]$ \hspace{1em} (5.3.8) - 128

Command to produce the function name 'ln' in formulas.

\LoadClass[options]{class}[version] [p] \hspace{1em} (D.2.2) - 442

This command may only be invoked within a class file to load another class file. It may only be called once within any class file. The file loaded must have the extension .cls. Any options specified in the \documentclass command are not passed over as global options.

The optional version is a date, given in the form yyyy/mm/dd, as for example 1994/08/01. If the date of the class file is earlier than this, a warning message is printed.

\LoadClassWithOptions{class}[version] [p] \hspace{1em} (D.2.2) - 442

Like \LoadClass except all the currently specified options are automatically passed to class.

\location{number} \hspace{1em} (16.1) - 352

In the letter document class, enters the sender's room number. In the standard \LaTeX letter class, number is only output if \address has not been called. It is intended to be used in company letterheads.

$\log [m]$ \hspace{1em} (5.3.8) - 128

Command to produce the function name 'log' in formulas.
H.1. Brief description of the \LaTeX{} commands 557

\Longleftarrow [m] produces ⇐ .................. (5.3.5) - 127
\longleftarrow [m] produces ← .................. (5.3.5) - 127
\Longleftrightarrow [m] produces ⇐⇒ ............... (5.3.5) - 127
\longleftrightarrow [m] produces ←→ ............... (5.3.5) - 127
\longmapsto [m] produces ↦−→ .................... (5.3.5) - 127
\Longrightarrow [m] produces =⇒ ................. (5.3.5) - 127
\longrightarrow [m] produces −→ ................... (5.3.5) - 127

\langle produces ‘, identical to the ‘ symbol.
\vert [m][a] produces | (left delimiter) ............ (12.2.5) - 270
\Vert [m][a] produces || (left delimiter) .......... (12.2.5) - 270

\mainmatter ...................................... (3.3.5) - 57

In the book class, introduces the main body of text after the front matter, by resetting the page numbering to 1 with Arabic numbers, and by reactivating the chapter numbering with the \chapter command. It undoes the effects of \frontmatter.

\makebox[width][pos]{text} ....................... (4.7.1) - 86

Produces a box of width width containing text centered horizontally, unless pos is given to specify that it is to be left (l) or right (r) justified. It may also have the value s, to stretch the text to width.

\makebox[\(x_{dimen},y_{dimen}\)][pos]{text} ...... (13.1.4) - 291

Picture element command to produce a box of width \(x_{dimen}\) and height \(y_{dimen}\) within the picture environment. Without the optional argument pos, the text is centered vertically and horizontally. The text may be left or right justified, and/or aligned at the top or bottom, by setting pos to a combination of the letters l, r, t, and b, such as tr for top, right; pos may also contain s to stretch the text to the full width. The command is used as the argument of a \put or \multiput command.

\makeglossary [p] .................................. (9.4.4) - 230
Command to activate the \glossary commands in the text.

\makeindex [p] ..................................... (9.4.2) - 226
Command to activate the \index commands in the text.

\makelabel{text} .................................. (4.4.1) - 75
An internal command that is called by the \item command to produce the actual label text within a list environment.

\makelabels ...................................... (16.1) - 355

Produces address labels in the letter document class using the entries from the \begin{letter} environment.

\MakeLowercase{text_cmd} ........................ (D.3.2) - 455
Converts text_cmd (text and commands) to lower case.
\MakeShortVerb{\c} ........................ (4.9.1, B.5.3) - 111, 393

When the standard package shortvrb has been loaded, this command makes the character \texttt{c} a shorthand form for \texttt{\verb{c}}: everything that appears between two occurrences of \texttt{c} is printed literally, in typewriter type. With \DeleteShortVerb{\c}, the character is restored to its normal meaning. Example: \MakeShortVerb{\|}

\maketitle ............................... (3.3.1) - 54

Produces a title page using entries in the \texttt{\author} and \texttt{\title} commands, and optionally those in the \texttt{\date} and \texttt{\thanks} commands.

\MakeUppercase{\text{cmd}} .......................... (D.3.2) - 455

Converts \text{cmd} (text and commands) to upper case.

\mapsto [m] produces \rightarrow .................. (5.3.5) - 127

\marginpar[\text{left text}]{\text{right text}} .......................... (4.10.5) - 117

Produces a marginal note at the right of the text containing \text{right text}. With two-sided formatting, the marginal note goes into the left margin on the even pages, in which case the optional \text{left text} will be written instead. For two-column text, the marginal notes always go into the 'outer' margin, and again \text{left text} will be used for the left margin.

\marginparpush .................................. (4.10.6) - 118

The minimum vertical separation between two marginal notes. A new value may be assigned with the \texttt{\setlength} command.

\marginparsep .................................. (4.10.6) - 118

The spacing between the edge of the text and a marginal note. A new value may be assigned with the \texttt{\setlength} command.

\marginparwidth .................................. (4.10.6) - 118

The width of the box reserved for marginal notes. A new value may be assigned with the \texttt{\setlength} command.

\markboth{\text{left head}}{\text{right head}} .................. (3.2.1) - 43

Sets the text entries for the left and right page headlines in two-sided formatting when the page style \texttt{myheadings} has been selected, or when the automatic entries of page style \texttt{headings} are to be changed.

\markright{\text{head}} .......................... (3.2.1) - 43

Sets the text entry for the page headline when the page style \texttt{myheadings} has been selected, or when the automatic entry of page style \texttt{headings} is to be manually changed. In two-sided formatting, only the right headline is set with this command.

\mathbf{\text{text}} [m] .......................... (5.4.2), (A.3.3) - 133, 373

This command sets \text{text} in a bold font (\texttt{\bfseries}) within math mode. Spaces are ignored as usual.
H.1. Brief description of the \LaTeX\ commands

\texttt{\textbackslash mathcal\{text\} [m]} .................. (5.3.2), (5.4.2), (A.3.3) - 125, 133, 373
This command sets text in calligraphic letters within math mode:
$\texttt{\mathcal\{ABC\}}$ = \textit{ABC}.

\texttt{\textbackslash mathindent} ................................................ (3.1.1) - 40
The indentation of displayed formulas from the left margin when the option \texttt{fleqn} has been selected. A new value may be assigned with the \texttt{\setlength\{mathindent\}}{25pt}

\texttt{\textbackslash mathit\{text\} [m]} .................... (5.4.2), (A.3.3) - 133, 373
This command sets text in a text italic font (\texttt{\itshape}) within math mode. It differs from \texttt{\mathnormal} in that the spacing between the letters is as in regular text. Compare \textit{mathit} and \textit{mathnormal}.

\texttt{\textbackslash mathnormal\{text\} [m]} ............ (5.3.1), (5.4.2), (A.3.3) - 125, 133, 373
This command sets text in the normal (italic) math font within math mode. In this font, capital Greek letters are also set in italics:
$\texttt{\Gamma\mathnormal\{\Gamma\}}$ = \textit{$\Gamma$}.

\texttt{\textbackslash mathring\{x\} [m]} ..................... (5.3.9) - 129
A ring accent in mathematical formulas: \texttt{\mathring\{a\}} = â.

\texttt{\textbackslash mathrm\{text\} [m]} ............... (5.4.2), (A.3.3) - 133, 373
This command sets text in a Roman font (\texttt{\rmfamily}) within math mode. Spaces are ignored as usual.

\texttt{\textbackslash mathsf\{text\} [m]} ................. (5.4.2), (A.3.3) - 133, 373
This command sets text in a sans serif font (\texttt{sffamily}) within math mode. Spaces are ignored as usual.

\texttt{\textbackslash mathtt\{text\} [m]} ................. (5.4.2), (A.3.3) - 133, 373
This command sets text in a typewriter font (\texttt{ttfamily}) within math mode. Spaces are ignored as usual.

\texttt{\textbackslash mathversion\{ver\}} ....................... (A.3.3) - 373
Selects the current math version. Possible values are \texttt{bold} and \texttt{normal}; new values may be created with the \texttt{\DeclareMathVersion} command.

\texttt{\textbackslash max [m]} ............................................ (5.3.8) - 128
Command to produce the function name 'max' in formulas. A lower limit may be set as a subscript.

\texttt{\textbackslash mbox\{text\}} produces an LR box around text ........ (4.7.1) - 86
\texttt{\textbackslash mddefault} ........................................... (A.3.1) - 372
This command defines the series attribute that is selected with the \texttt{\mdseries} command. It may be redefined with \texttt{\renewcommand}:
\texttt{\renewcommand\{\mddefault\}[m]}

\mdseries ........................................ (4.1.3), (A.2) - 64, 371
   This declaration switches to a font in the current family and shape, but with the medium series attribute.

\medskip ........................................ (2.7.3) - 32
   Inserts large vertical spacing of the amount \medskipamount. See also \bigskip and \smallskip.

\medskipamount
   Standard value for the amount of vertical spacing that is inserted with the command \medskip. May be changed with the \setlength command:
   \setlength{\medskipamount}{3ex plus1ex minus1ex}

\medspace [m][a] ................................. (12.2.5) - 269
   With the amsmath package, this is an alias for \:, a medium space in a math formula.

\MessageBreak .................................... (D.2.7) - 446
   Forces a new line in the texts of error, warning, and information messages. These are the only places where it may be invoked, otherwise it does nothing.

\mho [m] produces ℧ ................................ (5.3.6) - 127
\mid [m] produces | ................................ (5.3.4) - 126
\min [m] .......................................... (5.3.8) - 128
   Command to produce the function name 'min' in formulas. A lower limit may be set as a subscript.

\mod{} [m][a] ................................. (12.2.5) - 268
   With the amsopn or amsmath packages, command to produce the function name 'mod' in formulas in the form:
   \[ y \mod{a+b} = y \mod a + b \]

\models [m] produces ⊨ ........................... (5.3.4) - 126
\mp [m] produces ⊖ ............................... (5.3.3) - 125
\mspace{mu} [m][a] ................................ (12.2.5) - 269
   With the amsmath package, inserts spacing in math formulas; mu is a math space in units of mu (=1/18 em): \mspace{-4mu}.

\mu [m] produces µ ................................ (5.3.1) - 125
\multicolumn{n}{col}{text} .......................... (4.8.1) - 97
   Merges the next n columns in the array and tabular environments, formatting the text entry text according to the single column definition col, which may be 1, c, r as well as |.
\texttt{\textbackslash{}multiput}(x,y)(\Delta x,\Delta y){n}\{\texttt{pic\_elem}\} \hspace{1em} (13.1.3) - 289

Multiple positioning command in the \texttt{picture} environment. The object \texttt{pic\_elem} is placed \(n\) times, at \((x,y),(x+\Delta x,y+\Delta y),\ldots(x+(n-1)\Delta x,y+(n-1)\Delta y)\).

\texttt{\textbackslash{}multilinegap}[m][a] \hspace{1em} (12.2.6) - 271

A length that determines the left and right margins of formulas produced with the \texttt{AMS-L\LaTeX} \texttt{multline} environment; initial value is 10 pt but may be reset by the user.

\texttt{\textbackslash{}nabla}[m] \hspace{0.5em} produces \(\nabla\) \hspace{1em} (5.3.6) - 127

In the \texttt{letter} document class, enters the sender's name.

\texttt{\textbackslash{}natural}[m] \hspace{0.5em} produces \(\natural\) \hspace{1em} (5.3.6) - 127

\texttt{\textbackslash{}nearrow}[m] \hspace{0.5em} produces \(\nearrow\) \hspace{1em} (5.3.5) - 127

\texttt{\textbackslash{}NeedsTeXFormat}\{\texttt{format}\}[\texttt{version}][p] \hspace{1em} (D.2.1) - 440

Declares the \TeX{} format that is necessary for processing the file. This should be the first statement in the file. At the moment, the only legitimate value for \texttt{format} is \texttt{LaTeX2e}. The \texttt{version}, if included, must be given as a date in the form \texttt{yyyy/mm/dd}, specifying the earliest possible release date of the format that is consistent with all the features employed in the file. Example:

\texttt{\textbackslash{}NeedsTeXFormat\{LaTeX2e\}[1994/06/01]}

\texttt{\textbackslash{}neg}[m] \hspace{0.5em} produces \(\neg\) \hspace{1em} (5.3.6) - 127

\texttt{\textbackslash{}negmedspace}[m][a] \hspace{1em} (12.2.5) - 269

With the \texttt{amsmath} package, this inserts a negative medium space in a math formula.

\texttt{\textbackslash{}negthickspace}[m][a] \hspace{1em} (12.2.5) - 269

With the \texttt{amsmath} package, this inserts a negative thick space in a math formula.

\texttt{\textbackslash{}negthinspace}[m][a] \hspace{1em} (12.2.5) - 269

With the \texttt{amsmath} package, this is an alias for \texttt{\!}, a negative thin space in a math formula.

\texttt{\textbackslash{}neq}[m] \hspace{0.5em} produces \(\neq\) \hspace{1em} (5.3.4) - 127

\texttt{\textbackslash{}newboolean}\{\texttt{switch}\} \hspace{1em} (8.3.5) - 194

Requires the standard \LaTeX{} package \texttt{ifthen}. Creates a new boolean switch. The value of the switch is set with \texttt{\setboolean\{switch\}\{value\}}, where \texttt{value} is \texttt{true} or \texttt{false}. Its value is tested with \texttt{\boolean\{switch\}}, which may be used as a logical statement in the \texttt{test} part of \texttt{\ifthenelse} and \texttt{\whiledo}. 


\newcommand{\textit{com\_name}}{[\textit{narg}][\textit{opt}]{\textit{def}}} \ldots \ldots \ldots (8.3) - 185

Defines a user command with the name \textit{com\_name} to be \textit{def}. The first optional argument \textit{narg} \leq 9 specifies how many variable arguments the command is to have, which appear in the \textit{def} as the replacement characters #1 to #\textit{narg}. If the second optional argument is present, the first argument of the new command is optional, and takes on the value \textit{opt} if it is not explicitly given.

\newcommand*{\textit{com\_name}}{[\textit{narg}][\textit{opt}]{\textit{def}}} \ldots \ldots (D.2.6) - 445

The same as \newcommand except that the arguments to \textit{com\_name} must be 'short', not containing any new paragraphs.

\newcounter{\textit{counter\_name}}{[\textit{in\_counter}]} \ldots \ldots (8.1.2) - 182

Establishes a new counter with the name \textit{counter\_name}. The optional argument \textit{in\_counter} is the name of an existing counter which, when incremented, resets the new counter to zero; that is, the new counter is a sub-counter of \textit{in\_counter}.

\newenvironment{\textit{env\_name}}{[\textit{narg}][\textit{opt}]{\textit{beg\_def}}}{{\textit{end\_def}}} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (8.4) - 195

Defines a user environment with the name \textit{env\_name} which has the \textit{\begin} definition \textit{beg\_def} and the \textit{\end} definition \textit{end\_def}. The optional argument \textit{narg} \leq 9 specifies how many variable arguments the environment is to have, which appear in the \textit{beg\_def} as the replacement characters #1 to #\textit{narg}. If the second optional argument is present, the first argument of the \textit{\begin} command is optional, and takes on the value \textit{opt} if it is not explicitly given.

\newenvironment*{\textit{env\_name}}{[\textit{narg}][\textit{opt}]{\textit{beg\_def}}}{{\textit{end\_def}}} \ldots \ldots (D.2.6) - 445

The same as \newenvironment except that the arguments to \textit{\begin{env\_name}} must be 'short', not containing any new paragraphs.

\newfont{\textit{\texttt{font\_cmd}}}{{\textit{\texttt{font\_name}} \textit{scaled} \textit{size}}} \ldots \ldots (4.1.5) - 66

Establishes the relation between the font file name \textit{file\_name} magnified by the scaling factor \textit{size} and a font selection command \textit{\texttt{font\_cmd}}. After \textit{\texttt{font\_cmd}} has been called, \texttt{\baselineskip}, the interline spacing, still has its previous value.

\newlength{\textit{\texttt{length\_cmd}}} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (8.2) - 185

Creates a new length command with the name \textit{\texttt{length\_cmd}} and initializes it to 0 pt. New values may be assigned as for all length commands with the \texttt{\setlength} command:

\texttt{\setlength{\textit{\texttt{length\_cmd}}}\textit{\texttt{length}}}

The quantity \textit{\texttt{length}} must have units (cm, pt, etc.) and may be a rubber length.
H.1. Brief description of the \LaTeX{} commands

\texttt{\textbackslash newline} ................................................. (2.7.2) - 31
Terminates and starts a line of text \textit{without} right justifying it.

\texttt{\textbackslash newpage} .............................................. (2.7.4) - 33
Terminates and starts a new page, leaving the rest of the page blank.

\texttt{\textbackslash newsavebox\{\textbackslash boxname\}} ............... (4.7.1) - 87
Creates a storage box with the name \textbackslash boxname in which LR boxes may be saved with the \textbackslash savebox command.

\texttt{\textbackslash newtheorem\{type\}[num_like]\{title\}}
\texttt{\textbackslash newtheorem\{type\}\{title\}[in.ctr]} ................. (4.5) - 80
Defines a new theorem-like environment named type which when called prints a theorem declaration with the name title in \textbf{bold face}, followed by an automatic sequential number, and the actual text of the environment in \textit{italic}. The optional argument num\_like is the name of another theorem structure which is to share the same numbering counter. The other optional argument in\_ctr is the name of a sectioning counter, such as chapter, which is to reset the theorem counter every time it is incremented. That is, the theorem counter is a sub-counter of in\_ctr. Only one of the optional arguments may be given.

\texttt{\textbackslash NG} .................................................... (G.4.5) - 503
When T1 encoding is active, prints the character Ê.

\texttt{\textbackslash ng} ................................................... (G.4.5) - 503
When T1 encoding is active, prints the character ñ.

\texttt{\textbackslash ni [m] produces ð} ....................................... (5.3.4) - 126
\texttt{\textbackslash nocite\{key\}} ........................................ (14.1) - 310
The entry in the literature database with the keyword key will be included in the bibliography without any citation (reference) in the text. With \texttt{\textbackslash nocite\{\*\}}, all entries in all databases will be included.

\texttt{\textbackslash nofiles [p]} ............................................. (B.6) - 396
Issued in the preamble, this command suppresses the output of the auxiliary files .aux, .glo, .idx, .lof, .lot, and .toc.

\texttt{\textbackslash noindent} .............................................. (3.2.4) - 46
The first line of the next paragraph will \textit{not} be indented.

\texttt{\textbackslash nolimits [m]} ....................................... (5.3.7) - 128
Places the upper and lower limits after the appropriate symbols where these would normally go just above or below them.
Appendix H. Command Summary

\texttt{\\nolinebreak}[n] \hspace{1cm} (2.7.2) – 32
A recommendation \textit{not} to break the line of text at this point. The urgency of the recommendation is given by the integral number \textit{n} between 0 and 4, with the higher numbers meaning a stronger recommendation. A value of 4 is the same as the command without the optional argument and means absolutely no line break here.

\texttt{\nnonfrenchspacing} \hspace{1cm} (2.7.1) – 29
Countermands \texttt{\frenchspacing}, switching back to the standard formatting in which extra word spacing is inserted at the end of a sentence.

\texttt{\nonumber [m]} \hspace{1cm} (5.4.7) – 139
The formula line in an \texttt{eqnarray} environment in which this command appears will \textit{not} contain an equation number.

\texttt{\nopagebreak}[n] \hspace{1cm} (2.7.4) – 33
A recommendation \textit{not} to break the page at this point. The urgency of the recommendation is given by the integral number \textit{n} between 0 and 4, with the higher numbers meaning a stronger recommendation. A value of 4 is the same as the command without the optional argument and means absolutely no page break here.

\texttt{\normalcolor} \hspace{1cm} (6.2) – 167
A command that normally does nothing. However, if the \texttt{color} package has been loaded, it resets the color for text to be the color that was in effect at the end of the preamble, normally black. A \texttt{\color} command in the preamble can alter this 'standard' color. This command is called by many internal \LaTeX macros, to reset the text color when printing headlines and headings. Other packages should also use it so that they are consistent with the \texttt{color} package.

\texttt{\normalfont} \hspace{1cm} (4.1.3), (A.2) – 65, 371
This declaration switches to the font with the default family, shape, and series attributes.

\texttt{\normalmarginpar} \hspace{1cm} (4.10.5) – 117
Countermands \texttt{\reversemarginpar}, switching back to the standard placement of marginal notes in the 'outer' margin.

\texttt{\normalsize} \hspace{1cm} (4.1.2) – 62
Switches to the font size \texttt{\normalsize}, the size selected by the option in the \texttt{\documentclass} or \texttt{\documentstyle} commands. It is smaller than \texttt{\large} but larger than \texttt{\small}.

\texttt{\not [m]} \hspace{1cm} (5.3.4) – 127
Changes the following comparison symbol into its negative counterpart:
\[
\not\cong = \neq
\]
H.1. Brief description of the \LaTeX\ commands

\notag{mark} \[m]\[a] \quad \ldots \ldots \ldots \ldots \ldots \quad (12.2.6) - 271

Within one of the \texttt{AMSL}\LaTeX\ alignment environments, suppresses the automatic equation number.

\notin [m] \text{produces } \notin \quad \ldots \ldots \ldots \ldots \ldots \quad (5.3.4) - 127
\nu [m] \text{produces } \nu \quad \ldots \ldots \ldots \ldots \ldots \quad (5.3.1) - 125
\numberwithin{ctr}{in ctr} \[a] \quad \ldots \ldots \ldots \ldots \ldots \quad (12.2.7) - 277

With the \texttt{amsmath} package, redefines the counter \texttt{ctr} to be a subcounter of \texttt{in ctr}, meaning it is reset every time \texttt{in ctr} is incremented. The value of \texttt{in ctr} is printed with that of \texttt{ctr}. This is normally used to make equations in an article to be numbered within sections:

\numberwithin{equation}{section}

\nwarrow [m] \text{produces } \nwarrow \quad \ldots \ldots \ldots \ldots \ldots \quad (5.3.5) - 127
\O \text{produces } \Ø \quad \ldots \ldots \ldots \ldots \ldots \quad (2.5.6) - 24
\o \text{produces } \ø \quad \ldots \ldots \ldots \ldots \ldots \quad (2.5.6) - 24
\oddsidemargin \quad \ldots \ldots \ldots \ldots \ldots \quad (3.2.5) - 48

Sets the left margin for the odd-numbered pages in document class \texttt{book} or when the option \texttt{twoside} has been selected for other classes. In all other cases, it sets the left margin for all pages. A new value is assigned with the \texttt{\setlength} command:

\setlength{\evensidemargin}{1.5cm}

\odot [m] \text{produces } \odot \quad \ldots \ldots \ldots \ldots \ldots \quad (5.3.3) - 125
\OE \text{produces } \Œ \quad \ldots \ldots \ldots \ldots \ldots \quad (2.5.6) - 24
\oe \text{produces } \œ \quad \ldots \ldots \ldots \ldots \ldots \quad (2.5.6) - 24
\oint [m] \text{produces } \oint \quad \ldots \ldots \ldots \ldots \ldots \quad (5.3.7) - 128
\Omega [m] \text{produces } \Ω \quad \ldots \ldots \ldots \ldots \ldots \quad (5.3.1) - 125
\omega [m] \text{produces } \ω \quad \ldots \ldots \ldots \ldots \ldots \quad (5.3.1) - 125
\ominus [m] \text{produces } \⊖ \quad \ldots \ldots \ldots \ldots \ldots \quad (5.3.3) - 125
\onecolumn \quad \ldots \ldots \ldots \ldots \ldots \quad (3.2.7) - 51

Starts a new page and switches from two-column to one-column page formatting.

\onlynotes{\texttt{note_nums}} \quad \ldots \ldots \ldots \ldots \ldots \quad (15.1.3) - 327

In \texttt{slides} class, a command to be issued in the preamble to generate only those notes whose numbers appear in \texttt{note_nums}. The command behaves the same as \texttt{\onlyslides} for slides.

\onlyslides{\texttt{slide_nums}} \quad \ldots \ldots \ldots \ldots \ldots \quad (15.1.3) - 327

In \texttt{slides} class, a command to be issued in the preamble to generate only those slides whose numbers appear in \texttt{slide_nums}. The numbers are separated by commas, and may include a range with a hyphen: \texttt{\onlyslides{4,10-13,23}}.
\opening{dear} ................................. (16.1) - 353

In the \texttt{letter} environment of the \texttt{letter} class, this sets the form of the salutation at the start of the letter text; for example,
\opening{Dear George,}.

\oplus[m] produces $\oplus$ .......................... (5.3.3) - 125

\OptionNotUsed[p] ............................... (D.2.3) - 443

A command that may only be used in the definition of options, especially default options. It declares the \CurrentOption to be unprocessed. This is used if the processing of a default option should fail, say because some file is missing. $\LaTeX$ is then informed that this requested option is still outstanding.

\oslash[m] produces $\oslash$ ........................ (5.3.3) - 125

\otimes[m] produces $\otimes$ ........................ (5.3.3) - 125

\oval(x.dimen,y.dimen)[part] .................... (13.1.4) - 296

Picture element command to produce an oval with width $x.dimen$ and height $y.dimen$ in the \texttt{picture} environment. The optional \texttt{part} argument may take on values of $t$, $b$, $l$, and $r$ to draw only the top, bottom, left, or right halves of the oval. A combination of these values may be given to draw only a quarter of the oval, such as $t1$ or $l1$ for the top left part. To be used as an argument in a \texttt{put} or \texttt{multiput} command.

\ovalbox{text} ..................................... (4.7.9) - 94

With the \texttt{fancybox} package, is a variant of \texttt{fbox}, drawing a framed box with round corners around \texttt{text}; the thickness of the lines is given by \texttt{thinline}.

\ovalbox{text} ..................................... (4.7.9) - 94

With the \texttt{fancybox} package, is the same as \texttt{ovalbox} but the thickness of the lines is given by \texttt{thickline}.

\overbrace{sub.form} [m] ............................ (5.4.4) - 136

Produces a horizontal curly brace over the math formula \textit{sub.form}. Any following superscript will be placed centered above the horizontal brace.

\overbrace{a+b} = \overbrace{a + b} \quad \overbrace{x+y+z}^{\xi\eta\zeta} = \overbrace{x + y + z}

\overleftarrow{expr} [m][a] ........................ (12.2.2) - 262

With the \texttt{amsmath} package, places a long leftwards pointing arrow over the mathematical expression \textit{expr}.

\overrightarrow{expr} [m][a] ........................ (12.2.2) - 262

With the \texttt{amsmath} package, places a long double arrow over the mathematical expression \textit{expr}.
\overline{\text{sub form}} \quad \text{(5.4.4) - 136}

Produces a horizontal bar over the math formula \text{sub form}:

\overline{a-b} = \bar{a} - \bar{b}

\overrightarrow{\text{expr}} \quad \text{(12.2.2) - 262}

With the amsmath package, places a long rightwards pointing arrow over the mathematical expression \text{expr}.

\overset{\text{char}}{\text{\symbol{}}} \quad \text{(12.2.2) - 262}

With the amsmath package, places \text{char} over the math symbol \text{\symbol{}} in superscript size.

\P \quad \text{(2.5.5) - 23}

\PackageError{\text{pkg.name}}{\text{error text}}{\text{help}}{\text{[p]}} \quad \text{(D.2.7) - 446}

Writes an error message \text{error text} to the monitor and transcript file, labeled with the package name, and halts processing, waiting for a user response as for a \LaTeX error. If \text{H\langle return\rangle} is typed, the \text{help} text is printed. Both \text{error text} and \text{help} may contain \MessageBreak for a new line, \space for a forced space, and \protect before commands that are to have their names printed literally and not interpreted.

\PackageInfo{\text{pkg.name}}{\text{info text}}{\text{[p]}} \quad \text{(D.2.7) - 447}

Is like \PackageWarningNoLine except that the text \text{info text} is only written to the transcript file, and not to the monitor.

\PackageWarning{\text{pkg.name}}{\text{warn text}}{\text{[p]}} \quad \text{(D.2.7) - 446}

Writes \text{warn text} to the monitor and transcript file, labeled with the package name and the current line number of the input file. Processing continues. The \text{warn text} is formatted in the same way as that for \PackageError.

\PackageWarningNoLine{\text{pkg.name}}{\text{warn text}}{\text{[p]}} \quad \text{(D.2.7) - 446}

Is like \PackageWarning except that the current line number of the input file is not printed.

\pagebreak[n] \quad \text{(2.7.4) - 33}

A recommendation to break the page at this point. The urgency of the recommendation is given by the integral number \text{n} between 0 and 4, with the higher numbers meaning a stronger recommendation. A value of 4 is the same as the command without the optional argument and means an obligatory page break.

\pagecolor{col_spec} \quad \text{(6.2) - 167}

A command made available with the color package. Sets the background color starting with the current page. All following pages have the same background color until \text{\pagecolor} is called once more. The \text{col_spec} is the same as for \text{\color}.
Appendix H. Command Summary

\texttt{\textbackslash pagename} ........................................ (D.4.1) - 460
Command in the \texttt{letter} document class containing the text for page numbers after the first page. In English, this is 'Page' but may be altered for adaptation to other languages.

\texttt{\textbackslash pagenumbering\{style\}} .......................... (3.2.3) - 45
Determines the style of the page numbering and resets the page counter to 1. Possible values for \texttt{style} are: \texttt{arabic}, \texttt{roman}, \texttt{Roman}, \texttt{alph}, and \texttt{Alph}.

\texttt{\textbackslash pageref\{marker\}} ............................... (9.2.1) - 213
Prints the number of the page where \texttt{marker} has been set by a \texttt{\textbackslash label\{marker\}} command.

\texttt{\textbackslash pagestyle\{style\} [p]} .............................. (3.2) - 42
Determines the page style, that is, the contents of the head and footlines on every page. Possible values for \texttt{style} are: \texttt{plain}, \texttt{empty}, \texttt{headings}, and \texttt{myheadings}.

\texttt{\textbackslash path\{directory\}} ................................. (4.9.2) - 112
With the \texttt{url} package, prints \texttt{directory} literally, in typewriter font, with line breaks after non-letters, without hyphens. It functions much the same as the \texttt{url} command, but is \textit{logically} distinct since it is encoding something different.

\texttt{\textbackslash paperheight} .................................... (3.2.5) - 48
The total height of the page as specified by the page size option in the \texttt{\documentclass} command line. With the default \texttt{lettersize} option, this is 11 in; with \texttt{a4paper}, it is 29.7 cm. The additional option \texttt{landscape} interchanges the values of \texttt{\textbackslash paperwidth} and \texttt{\textbackslash paperheight}.

\texttt{\textbackslash paperwidth} ..................................... (3.2.5) - 48
The total width of the page as specified by the page size option in the \texttt{\documentclass} command line. With the default \texttt{lettersize} option, this is 8.5 in; with \texttt{a4paper}, it is 21 cm. The additional option \texttt{landscape} interchanges the values of \texttt{\textbackslash paperwidth} and \texttt{\textbackslash paperheight}.

\texttt{\textbackslash par} ................................................. (2.5.1) - 23
Ends the current paragraph and begins a new one. This command is equivalent to a blank line.

\texttt{\textbackslash paragraph\{short title\}\{title\}} .................. (3.3.3) - 55
The second last command in the sectioning hierarchy, coming between \texttt{\textbackslash subsubsection} and \texttt{\textbackslash subparagraph}. It formats \texttt{title} with the current sub-subsection number and an automatic sequential paragraph number. If the optional \texttt{short title} is given, it appears in place of \texttt{title} in the table of contents.
\paragraph*{title} . . . . . . . . . . . . . . . . (3.3.3) – 55
The same as \paragraph but without a number or an entry in the table of contents.

\parallel [m] produces \parallel . . . . . . . . . . . . . . . . (5.3.4) – 126
\parbox[pos][height][inner.pos]{width}{text} (4.7.3), (4.7.5) – 88, 90
Produces a vertical box of width width in which text is set in lines that are left and right justified to this width. The vertical positioning with respect to the surrounding text is determined by the optional argument pos: t for alignment with its top line, b with its bottom line, and centered with no argument. The two additional optional arguments: height to give the total height, and inner.pos to specify how the text is to be positioned inside it. Possible values are t for top, b for bottom, c for centered, and s to be stretched out to fill the whole vertical space. The default is the value of the external positioning pos option. The height argument may contain the parameters \height, \depth, \width, and \totalheight.

\parindent . . . . . . . . . . . . . . . . . . . . . . . (3.2.4) – 46
The amount of indentation for the first line of a paragraph. A new value may be assigned with the \setlength command:
\setlength{\parindent}{1.5em}

\parsep . . . . . . . . . . . . . . . . . . . . . . . (4.4.2) – 76
The vertical spacing between paragraphs within a list environment. A new value may be assigned with the \setlength command:
\setlength{\parsep}{2pt plus1pt minus1pt}

\parskip . . . . . . . . . . . . . . . . . . . . . . . (3.2.4) – 46
The vertical spacing between paragraphs. A new value may be assigned with the \setlength command:
\setlength{\parskip}{3pt plus1pt minus2pt}

\part[short title]{title} . . . . . . . . . . . . . . . . (3.3.3) – 55
The highest command in the sectioning hierarchy. It begins a new 'Part' with an automatic sequential part number and the heading title. The following sectioning numbers are not influenced by the part number. If the optional short title is given, it appears in place of title in the table of contents.

\part*{title} . . . . . . . . . . . . . . . . . . . . . (3.3.3) – 55
The same as \part but without a number or an entry in the table of contents.

\partial [m] produces \partial . . . . . . . . . . . . . . . . (5.3.6) – 127
\partname . . . . . . . . . . . . . . . . . . . . . . (D.4.1) – 460
Command containing the part heading. In English, this is 'Part' but may be altered for adaptation to other languages.
\partopsep
The additional vertical spacing at the beginning and/or end of a listing when a blank line precedes or follows the environment commands. A new value may be assigned with the \setlength command:
\begin{verbatim}
\setlength{\partopsep}{2pt plus1pt minus1pt}
\end{verbatim}
\PassOptionsToClass{options}{class} [p]
Assigns the options in the list options to the specified class file, which is later loaded with \LoadClass. This command must be called from a class file, or from another file input by a class file. It may be used in the definition of options, or in a configuration file to activate options.
\PassOptionsToPackage{options}{package} [p]
Assigns the options in the list options to the specified package file, which is later loaded with \RequirePackage. This command may be called from a class or package file. It may be used in the definition of options, or in a configuration file to activate certain options.
\perp [m] produces \( \perp \) \( (5.3.4) - 126\)
\Phi [m] produces \( \Phi \) \( (5.3.1) - 125\)
\phi [m] produces \( \phi \) \( (5.3.1) - 125\)
\Pi [m] produces \( \Pi \) \( (5.3.1) - 125\)
\pi [m] produces \( \pi \) \( (5.3.1) - 125\)
\pm [m] produces \( \pm \) \( (5.3.4) - 126\)
\pmb{symbol} [m][a]
When one of the packages amsmath or amsbsy has been loaded, this command prints symbol in simulated bold face. This is done by printing it several times slightly displaced.
\pmod{arg} [m]
Command to produce the function name ‘mod’ in formulas in the form:
\begin{equation}
y \pmod{a+b} = y \pmod{a+b}
\end{equation}
\pod{arg} [m][a]
With the amsopn or amsmath packages, command to produce the function name ‘mod’ in formulas in the form:
\begin{equation}
y \pod{a+b} = y \pmod{a+b}
\end{equation}
\poptabs
Restores the last set of tabular stops in the tabbing environment that has been saved with \pushtabs.
\pounds produces \( \pounds \) \( (2.5.5) - 23\)
\Pr [m]
Command to produce the function name ‘Pr’ in formulas. A lower limit may be set as a subscript.
H.1. Brief description of the \LaTeX commands

\texttt{\textbackslash prec [m]} produces $<$ \hfill (5.3.4) - 126
\texttt{\textbackslash preceq [m]} produces $\preceq$ \hfill (5.3.4) - 126
\texttt{\textbackslash prime [m]} produces ' (identical to the ' symbol) \hfill (5.3.6) - 127
\texttt{\textbackslash printindex} \hfill (9.4.3) - 228

A command defined in the \texttt{makeidx.sty} file that generates the \texttt{index} environment after the program MakeIndex has processed the \texttt{.idx} file.

\texttt{\textbackslash ProcessOptions [p]} \hfill (D.2.3) - 443

In a class or package file, this command processes the requested options by executing the \texttt{\ds@} commands for each one \textit{in the order in which they were defined}. The \texttt{\ds@} commands are then erased.

\texttt{\textbackslash ProcessOptions^[p]} \hfill (D.2.3) - 443

This is the same as \texttt{\textbackslash ProcessOptions} except that the \texttt{\ds@} commands are executed \textit{in the order in which the options were requested}.

\texttt{\textbackslash prod [m]} produces $\prod$ \hfill (5.3.7) - 128
\texttt{\textbackslash propto [m]} produces $\propto$ \hfill (5.3.4) - 126
\texttt{\textbackslash protect} \hfill (D.2.5) - 445

Fragile commands may be used in moving arguments when they are preceded by the \texttt{\protect} command. Example:

\texttt{\section{The \protect\pounds{} Sign}}.

\texttt{\textbackslash providecommand[\texttt{\com_name}[narg][opt]{def}} \hfill (8.3.1) - 187

The same as \texttt{\newcommand} except that if a command with the name \texttt{\com_name} already exists, the new definition is ignored.

\texttt{\textbackslash providecommand*{\texttt{\com_name}[narg][opt]{def}} \hfill (D.2.6) - 445

The same as \texttt{\providecommand} except that the arguments to \texttt{\com_name} must be 'short', not containing any new paragraphs.

\texttt{\textbackslash ProvidesClass{class}[version] [p]} \hfill (D.2.1) - 441

At the beginning of a class file, this statement declares the name of the class and its version, to be checked against the name and version in the \texttt{\documentclass} or \texttt{\LoadClass} command that input it. The \texttt{version} specification, if present, consists of three parts: date, version number, and additional information. Example:

\texttt{\ProvidesClass{thesis}[1995/01/25 v3.8 U of Saigon]}

\texttt{\textbackslash ProvidesFile{file_name}[version]} \hfill (D.2.1) - 442

At the beginning of a general file, this statement declares its name and version. No checking is done when the file is read in with \texttt{\input}, but the information is printed out if \texttt{\listfiles} has been activated. This command is not limited to the preamble as the other \texttt{\Provides..} commands are.
Appendix H. Command Summary

\ProvidesPackage{class}[version] [p] ................... (D.2.1) - 441

At the beginning of a package file, this statement declares the name of the package and its version, to be checked against the name and version in the \usepackage or \RequirePackage command that input it. The version specification, if present, consists of three parts: date, version number, and additional information. Example:

\ProvidesPackage{notes}[1995/02/13 1.2 G. Smith]

\ProvideTextCommand{\cmd}{code}{narg}[opt]{def} [p]

......................................................... (A.3.7) - 379

Defines \cmd in the same way as \providecommand except the definition is only valid when encoding code is active. If the command \cmd is already defined for that encoding, it is not redefined.

\ProvideTextCommandDefault{\cmd}{code}{narg}[opt]{def} [p]

......................................................... (A.3.7) - 380

Is the same as \DeclareTextCommandDefault except that if a default encoding definition already exists for \cmd, then no redefinition occurs and the previous definition is retained.

\ps text .............................................. (16.1) - 353

 Adds a postscript to a letter in the letter document class.

\Psi [m] produces Ψ .................................. (5.3.1) - 125
\psi [m] produces ψ .................................. (5.3.1) - 125

\pushtabs ............................................. (4.6.4) - 83

Saves the current set of tabulator stops in the tabbing environment. It may be recalled with the \poptabs command.

\put(x,y)\{pic.elem\} ................................. (13.1.3) - 289

The positioning command within a picture environment. The picture element pic.elem is placed with its reference point at the location (x,y).

\qbezier[num](x_1,y_1)(x_2,y_2)(x_3,y_3) ............. (13.1.4) - 299

This command can be given within the picture environment to draw a quadratic Bézier curve from point (x_1,y_1) to (x_3,y_3), using (x_2,y_2) as the extra Bézier point. The curve is drawn as num + 1 dots if the optional argument num is given, otherwise num is calculated automatically to produce a solid line. It is the same as \bezier except that num is optional.

\quad .................................................. (2.7.1) - 30

Inserts horizontal spacing of size 1 em.

\qquad .................................................. (2.7.1) - 30

Inserts horizontal spacing of size 2 em.


H.1. Brief description of the \LaTeX \ commands \hspace{1cm} 573

\texttt{\textbackslash quotedblbase} \hspace{1cm} (G.4.5) – 503

When T1 encoding is active, prints the symbol \textquotedbl{."

\texttt{\textbackslash quotesinglbase} \hspace{1cm} (G.4.5) – 503

When T1 encoding is active, prints the symbol \quotesingle{."

\texttt{\textbackslash r\{x\}} \hspace{1cm} (2.5.7) – 24

Produces a circle accent: \texttt{\textbackslash r\{o\} = \textperiodcentered.

\texttt{\textbackslash raggedbottom} \hspace{1cm} (3.2.5) – 48

The standard page formatting for article, report, and letter
document classes when the twoside option has \textit{not} been selected.
The spacing between paragraphs is fixed so that the last line will vary
from page to page. The opposite command is \texttt{\textbackslash flushbottom}.

\texttt{\textbackslash raggedleft} \hspace{1cm} (4.2.2) – 67

After this declaration, the lines of text will only be right justified and
the left margin will be uneven. The individual lines are terminated
by \texttt{\textbackslash}. See also \texttt{\textbackslash begin\{flushright\}}.

\texttt{\textbackslash raggedright} \hspace{1cm} (4.2.2) – 67

After this declaration, the lines of text will only be left justified and
the right margin will be uneven. The individual lines are terminated
by \texttt{\textbackslash}. See also \texttt{\textbackslash begin\{flushleft\}}.

\texttt{\textbackslash raisebox\{lift\}\[height\]\[depth\]\{text\}} \hspace{1cm} (4.7.2) – 87

An LR box containing \texttt{text} is raised an amount \texttt{lift} above the current
baseline. If \texttt{lift} is negative, the box is lowered. The optional arguments state that it is to be treated as though it extended by \texttt{height}
above and by \texttt{depth} below the baseline regardless of its true extents.

\texttt{\textbackslash raisetag\{len\} [m]\[a]} \hspace{1cm} (12.2.6) – 271

Within one of the \texttt{\textbackslash AMS-L\LaTeX} alignment environments, raises the
equation number or marker by \texttt{len} above its normal position.

\texttt{\textbackslash range [m] produces )} \hspace{1cm} (5.4.1) – 132
\texttt{\textbackslash rceil [m] produces ]} \hspace{1cm} (5.4.1) – 132
\texttt{\textbackslash Re [m] produces \textbackslash{\textbackslashceiling{\textbackslashtext}}\textbackslash}} \hspace{1cm} (5.3.6) – 127
\texttt{\textbackslash ref\{marker\}} \hspace{1cm} (9.2.1) – 213

Prints the number of the section, equation, figure, or table where
\texttt{marker} has been set by a \texttt{\textbackslash label\{marker\}} command.

\texttt{\textbackslash reflectbox\{text\}} \hspace{1cm} (6.1.2) – 156

A command made available with the graphics package that reflects
the contents \texttt{text} as an LR box such that left and right are reversed.
\refname  ........................................... (D.4.1) - 460
Command containing the heading for the bibliography in article
document class. In English, this is 'References' but may be altered
for adaptation to other languages.

\refstepcounter{counter}  .................. (8.1.3) - 182
Increases the value of the number stored in counter by one, the same
as \stepcounter, but also makes the specified counter the relevant
one for the \label-\ref cross-referencing commands.

\renewcommand{com_name}[narg][opt]{def}  ........ (8.3) - 185
The same as \newcommand except that the command \com_name
must already exist, otherwise an error message is printed.

\renewcommand*[com_name][narg][opt]{def}  ........ (D.2.6) - 445
The same as \renewcommand except that the arguments to
\com_name must be 'short', not containing any new paragraphs.

\renewenvironment{env}[narg][opt]{beg}{end}  .... (8.4) - 195
The same as \newenvironment except that the environment \env
must already exist, otherwise an error message is printed.

\renewenvironment*[{env}[narg][opt]{beg}{end}  . . (D.2.6) - 445
The same as \renewenvironment except that the arguments to
\begin{env} must be 'short', not containing any new paragraphs.

\RequirePackage[options]{packages}[version] [p] . . (D.2.2) - 442
This command is the equivalent of \usepackage within a class or
package file. It loads one or more package files with the extension
.sty. More than one package may be specified in \packages, the names
being separated by commas. Any \options listed will be applied to all
packages. Furthermore, any options listed in the \documentclass
command will also be applied to the package files.

The optional \version is a date, given in the form \texttt{yyyy/mm/dd}, as for
example \texttt{1994/08/01}. If the date of the package file is earlier than
this, a warning message is printed.

\RequirePackageWithOptions{package}[version] [p]  . (D.2.2) - 442
Like \RequirePackage except all the currently specified options are
automatically passed to \package.

\resizebox{h_length}{v_length}{text}  ............ (6.1.2) - 156
A command made available with the \texttt{graphics} package that scales
the contents \texttt{text} as an LR box such that the horizontal size becomes
\texttt{h_length} and the vertical size \texttt{v_length}. If either size is given as \texttt{!}, the
one scale factor is applied to both dimensions.
H.1. Brief description of the \LaTeX{} commands

\\resizebox*{\hsize}{\vsize}{\text}
(6.1.2) – 156
The same as \resizebox except that the vertical size \vsize refers to the total height plus depth of the LR box.

\reversemarginpar
(4.10.5) – 117
Changes the placement of marginal notes from the standard (right or 'outer' margin) to the opposite side. Can be countermanded with \normalmarginpar.

\rfloor[m] produces \[.
(5.4.1) – 132
Adjusts the size of the bracket symbol \rbrack to fit the height of the formula between the \left ... \right pair. For example, \right. If there is to be no matching bracket, the \left and \right commands must still be given to specify the part of the formula to be sized, but the missing bracket is given as a period (for example, \left.).

\Rightarrow[m] produces ⇒.
(5.3.5) – 127
\rightarrow[m] produces →.
(5.3.5) – 127
\rightharpoonup[m] produces ⇀.
(5.3.5) – 127
\rightharpoondown[m] produces ⇁.
(5.3.5) – 127
\rightleftharpoons[m] produces ⇋.
(5.3.5) – 127
\rightmargin
(4.4.2) – 76
In a list environment, the amount by which the right edge of the text is indented relative to the right side of the surrounding text. Standard value is 0 pt. A new value is assigned with the \setlength command:
\setlength{\rightmargin}{0.5cm}

\rmdefault
(A.3.1) – 372
This command defines the family attribute that is selected with the \rmfamily command. It may be redefined with \renewcommand:
\renewcommand{\rmdefault}{ptm}

\rmfamily
(4.1.3), (A.2) – 64, 371
This declaration switches to a font in the current series and shape, but with the Roman family attribute.

\Roman{\text}
(8.1.4) – 183
Prints the current value of the \text as an upper case Roman numeral.

\roman{\text}
(8.1.4) – 183
Prints the current value of the \text as a lower case Roman numeral.
\rotatebox{angle}{text} ........................................... (6.1.2) – 156
A command made available with the graphics package that rotates the contents text as an LR box through the angle expressed in degrees. The rotation is counterclockwise about the left-hand end of the baseline of the box.

\rq produces ', identical to the ' symbol.

\rule[lift]{width}{height} ................................. (4.7.6) – 91
Produces a black rectangle of width width and height height, raised above the baseline by an amount lift, if this optional argument is given. A value of '0 pt' for either the width or height creates an invisible horizontal or vertical strut that may be used to make spacing.

\rvert [m][a] produces | (right delimiter) ............ (12.2.5) – 270
\rVert [m][a] produces || (right delimiter) ............ (12.2.5) – 270
\S produces § ............................................. (2.5.5) – 23
\SS produces SS, the upper case version of \ss, § .......... (2.5.6) – 24
\savebox{\boxname}[width][pos]{text} ................. (4.7.1) – 87
Functions the same as the \makebox command except that the box contents are not output but saved under the name \boxname, which has been previously defined with \newsavebox. The box may be set any place in the text as often as desired with the command \usebox{\boxname}.

\savebox{\boxname}(x\ dim,y\ dim)[pos]{sub_pic}  (13.1.4) – 300
In the picture environment, a sub-picture sub_pic may be stored as a box of width x\ dim and height y\ dim under the name \boxname, which has been previously defined with \newsavebox. The pos argument functions as it does for the picture \makebox. The box may be set any place in the picture environment with the command \usebox{\boxname}.

\sb produces a subscript, identical to the _ symbol.
\sbox{\boxname}{text} ........................................... (4.7.1) – 87
Stores text in an LR box named \boxname that has previously been created with \newsavebox{\boxname}. The contents of the box may be printed as often as desired with \usebox{\boxname}.

\scalebox{h_scale}[v_scale]{text} ....................... (6.1.2) – 156
A command made available with the graphics package that scales the contents text as an LR box with the horizontal factor h_scale and optionally with the (different) vertical factor v_scale. If v_scale is missing, it is the same as h_scale.

\scdefault .................................................... (A.3.1) – 372
This command defines the shape attribute that is selected with the \scshape command. It may be redefined with \renewcommand:
\renewcommand{\scdefault}{sc}
H.1. Brief description of the \LaTeX{} commands

\scriptscriptstyle[m] \hspace{1em} (5.5.2) - 146
Switches to font size \scriptscriptstyle as the active font inside a math formula.

\scriptsize \hspace{1em} (4.1.2) - 62
Switches to the font size \scriptsize, which is smaller than \footnotesize but larger than \tiny.

\scriptstyle \hspace{1em} (5.5.2) - 146
Switches to font size \scriptstyle as the active font inside a math formula.

\scshape \hspace{1em} (4.1.3), (A.2) - 64, 371
This declaration switches to a font in the current family and series, but with the CAPS AND SMALL CAPS shape attribute.

\searrow (m) produces \hspace{1em} (5.3.5) - 127
\sec (m) \hspace{1em} (5.3.8) - 128
Command to produce the function name 'sec' in formulas.

\section[short title]{title} \hspace{1em} (3.3.3) - 55
Begins a new section, formatting \textit{title} with the current chapter number (book and report classes only) and an automatic sequential section number. If the optional \textit{short title} is given, it appears in place of \textit{title} in the table of contents and the running head at the top of the pages.

\section*{title} \hspace{1em} (3.3.3) - 55
The same as \texttt{\section} but without a number or an entry in the table of contents.

\see{reference} \hspace{1em} (9.4.2) - 227
A command defined in the makeidx package for use with the MakeIndex program. It is called within an \texttt{\index} command to refer to another entry in the keyword index as 'see \textit{reference}'. Given in the form:

\texttt{\index{entry|see{reference}}}

Note: the above text is correct with \texttt{|} in place of \texttt{\ for \texttt{\ see.}}

\seealso{reference} \hspace{1em} (9.4.2) - 227
Like \texttt{\see}, but will print the text 'see also \textit{reference}' in the index. More precisely, it prints the text stored in the command \texttt{\alsoname}. Requires the makeidx package.

\seename \hspace{1em} (D.4.1) - 460
A command defined in the package makeidx containing the text for the command \texttt{\see}. In English, this is 'see' but may be altered for adaptation to other languages.
\selectfont ........................................ (A.1) – 369

Activates the font with the current set of attributes, making it the current font in which text is set. It should normally follow an attribute selection command. Example:

\fontshape{sl}\selectfont

\selectlanguage{language} ................................ (11.1) – 255

Command in multi-language packages such as german and in the babel system for changing the language. The names of titles, the form of the date command \today, special language-specific commands, and the hyphenation patterns are all changed. Example:

\selectlanguage{english}

\setboolean{switch}{value} ................................ (8.3.5) – 194

Requires the standard \LaTeX package ifthen. Sets the value of a boolean switch to (true) or (false), depending on value, which must be true or false. The switch must have been created with \newboolean{switch}. Its value is tested with \boolean{switch}, which may be used as a logical statement in the test part of \ifthenelse and \whiledo.

\setcounter{counter}{value} ................................ (8.1.3) – 182

Assigns the integral number value to the counter counter.

\setlength{\length_cmd}{length_spec} ........................ (8.2) – 184

The length command with the name \length_cmd is assigned the length value length_spec, which may be a fixed or rubber length. See also .......................... (2.4.1), (2.4.2) – 21, 21

\SetMathAlphabet{\cmd}{ver}{code}{fam}{ser}{shp} [p]

......................................................... (A.3.3) – 373

Defines the math alphabet that has been declared with \DeclareMathAlphabet to take the specified font attributes in the math version ver.

\setminus [m] produces \ ................................ (5.3.3) – 125

\SetSymbolFont{sym_fnt}{ver}{code}{fam}{ser}{shp} [p]

......................................................... (A.3.4) – 374

Defines the symbol font that has been declared with \DeclareSymbolFont to take the specified font attributes in the math version ver.

\settime{secs} ........................................ (15.1.3) – 327

In the \slides class, if the option clock has been selected, a time marker, in minutes, appears at the bottom of the notes. This command sets the internal timer to the specified number of seconds. See also \addtime.
H.1. **Brief description of the \LaTeX commands**

\settodepth{\length}{text} .................................. (8.2) - 184

The length command with the name \length is assigned a value equal to the depth of text below the baseline.

\settoheight{\length}{text} .................................. (8.2) - 184

The length command with the name \length is assigned a value equal to the height of text above the baseline.

\settowidth{\length}{text} .................................. (8.2) - 184

The length command with the name \length is assigned a value equal to the length of text as it would be set in an LR box.

\sfdefault ....................................................... (A.3.1) - 372

This command defines the family attribute that is selected with the \sffamily command. It may be redefined with \renewcommand:
\begin{verbatim}
\renewcommand{\sfdefault}{phv}
\end{verbatim}

\sffamily ....................................................... (4.1.3), (A.2) - 64, 371

This declaration switches to a font in the current series and shape, but with the sans serif family attribute.

\shadowbox{text} .................................................. (4.7.9) - 94

With the fancybox package, a variant of \fbox, drawing a shadowed box around text; the thickness of the shadow is given by the length \shadowsize.

\sharp [m] produces # ........................................... (5.3.6) - 127

\shortstack[pos]{text} ........................................... (13.1.4) - 297

Formats the text into a single column, where the individual rows are terminated by \\. The optional positioning argument pos takes on values of l or r to set the text left or right justified, otherwise it is centered. Example:
\begin{verbatim}
\shortstack{aa\bbb\ccc\xx\yy\zzz}
\end{verbatim}

\sideset{pre}{post}\symbol [m][a] .......................... (12.2.2) - 262

With the amsmath package, places superscripts and subscripts snugly before (pre) and after (post) the math symbol \symbol. Example:
\begin{verbatim}
$\sideset{\_\dag^*}{\_\dag^*}\prod_1^n$
\end{verbatim}

\Sigma [m] produces $\Sigma$ ........................................ (5.3.1) - 125

\sigma [m] produces $\sigma$ ....................................... (5.3.1) - 125

\signature{name} ................................................... (16.1) - 351

In the letter document class, supplies the name of the writer that should go below the signature if this is different from the entry in \name.
\texttt{\sim} \ [m] \text{ produces } \sim \hspace{1cm} (5.3.4) - 126 \\
\texttt{\simeq} \ [m] \text{ produces } \simeq \hspace{1cm} (5.3.4) - 126 \\
\texttt{\sin} \ [m] \hspace{1cm} (5.3.8) - 128 \\
\text{Command to produce the function name 'sin' in formulas.} \\
\texttt{\sinh} \ [m] \hspace{1cm} (5.3.8) - 128 \\
\text{Command to produce the function name 'sinh' in formulas.} \\
\texttt{\slshape} \hspace{1cm} (4.1.3), (A.2) - 64, 371 \\
\text{This declaration switches to a font in the current family and series,} \\
\text{but with the slanted shape attribute.} \\
\texttt{\smallskip} \hspace{1cm} (2.7.3) - 32 \\
\text{Inserts large vertical spacing of amount \smallskipamount. See also} \\
\texttt{\medskip} \text{ and \bigskip.} \\
\texttt{\smallskipamount} \hspace{1cm} \\
\text{Standard value for the amount of vertical spacing that is inserted with} \\
\text{the command \smallskip. May be changed with the \setlength} \\
\text{command:} \\
\texttt{\setlength{\smallskipamount}{1ex plus0.5ex minus0.3ex}} \\
\texttt{\smash[pos]{text} [m][a]} \hspace{1cm} (12.2.5) - 269 \\
\text{With the amsmath package, this \TeX \ command acquires an optional} \\
\text{argument pos that may be b or t, to effectively zero the depth or} \\
\text{height of the text. With no pos, both height and depth are zeroed.} \\
\texttt{\smile} \ [m] \text{ produces } \⌣ \hspace{1cm} (5.3.4) - 126 \\
\texttt{\sp} \text{ produces a superscript, identical to the ^ symbol.} \\
\texttt{\spadesuit} \ [m] \text{ produces } \♣ \hspace{1cm} (5.3.6) - 127 \\
\texttt{\sqcap} \ [m] \text{ produces } \sqcap \hspace{1cm} (5.3.3) - 125 \\
\texttt{\sqcup} \ [m] \text{ produces } \sqcup \hspace{1cm} (5.3.3) - 125 \\
\texttt{\sqrt[n]{arg}} \ [m] \hspace{1cm} (5.2.4) - 122 \\
\text{Basic math command to produce a root sign. The height and length} \\
\text{of the sign are made to fit the contents arg. The optional argument} \\
n \text{is the degree of the root: } \sqrt[3]{2} = \sqrt[3]{2}, \sqrt{2} = \sqrt{2}.
\sqsubset [m] produces $\sqsubset$ ........................................... (5.3.4) – 126
\sqsubsetneq [m] produces $\sqsubsetneq$ ........................................... (5.3.4) – 126
\sqsupset [m] produces $\sqsupset$ ........................................... (5.3.4) – 126
\sqsupseteq [m] produces $\sqsupseteq$ ........................................... (5.3.4) – 126
\ss produces ß ................................................................. (2.5.6) – 24
\stackrel{\text{upper}}{\text{lower}} [m] ............................................. (5.4.5) – 137
Places one mathematical symbol upper on top of another lower, such
that the upper one appears in a smaller typeface:
\stackrel{\alpha}{\longrightarrow} = \alpha
\star [m] produces $\star$ ....................................................... (5.3.3) – 125
\stepcounter{counter} .......................................................... (8.1.3) – 182
Increases the value of the number stored in counter by one.
\stretch{decimal.num} .......................................................... (8.2) – 184
A rubber length with a natural value of 0 pt but with a stretchability
that is decimal_num times that of \fill.
\subitem{sub.entry} ............................................................. (9.4.1) – 225
In the \index environment, a command to produce a second-level
entry after an \item command.
\subparagraph[short title]{title} ............................................... (3.3.3) – 55
The last command in the sectioning hierarchy, coming after
\paragraph. It formats title with the current paragraph number
and an automatic sequential sub-paragraph number. If the optional
short title is given, it appears in place of title in the table of contents.
\subparagraph*[title] ........................................................... (3.3.3) – 55
The same as \subparagraph but without a number or an entry in
the table of contents.
\subsection[short title]{title} ............................................... (3.3.3) – 55
The command in the sectioning hierarchy that comes between the
\section and \subsubsection. It formats title with the current
section number and an automatic sequential subsection number. If
the optional short title is given, it appears in place of title in the table
of contents.
\subsection*[title] ........................................................... (3.3.3) – 55
The same as \subsection but without a number or an entry in the
table of contents.
\substack{1st line \\... \\
\last line} [m][a] ........................................ (12.2.2) – 261
With the \amsmath package, produces centered multiline indices or
limits; it must immediately follow ^ or _ and be enclosed in { }.
In the index environment, a command to produce a third-level entry under a \subitem command.

The command in the sectioning hierarchy coming between \subsection and \paragraph. It formats title with the current subsection number and an automatic sequential sub-subsection number. If the optional short title is given, it appears in place of title in the table of contents.

The same as \subsection but without a number or an entry in the table of contents.

Command to produce the function name ‘sup’ in formulas. A lower limit may be set as a subscript.

Any floats given between this command and the end of the current page will be suspended at least until the next page. If the optional loc is given as one of t or b (not both), only floats with that placement parameter are suspended.

Produces the symbol in the current character font that is stored under the internal number n.

Produces a ‘tie-after’ accent over two letters: \t{oo} = ōō.

Determines the spacing between the text ltext and the current tabular stop when ltext' is given in a tabbing environment. A new value may be assigned with the \setlength command.
H.1. Brief description of the \LaTeX{} commands

\texttt{\textbackslash tabcolsep} \hspace{1em} (4.8.2) – 98

Determines the half-column spacing in the \texttt{tabular} environment. A new value may be assigned with the \texttt{\setlength} command:
\[
\setlength{\texttt{\tabcolsep}}{3\text{mm}}
\]

\texttt{\textbackslash tableofcontents} \hspace{1em} (3.4.2) – 58

Prints the table of contents from information in the sectioning commands and additional entries.

\texttt{\textbackslash tablename} \hspace{1em} (D.4.1) – 459

Command containing the name for a table caption. In English, this is ‘Table’ but may be altered for adaptation to other languages.

\texttt{\textbackslash tabularnewline[\texttt{len}]} \hspace{1em} (4.8.1) – 98

Terminates a row in the \texttt{tabular} or \texttt{array} environments, adding vertical spacing \texttt{len} if it is specified. This is equivalent to \texttt{\textbackslash \textbackslash [\texttt{len}]} except that there is no ambiguity as to whether it is terminating a row in the table or a line of text within a column entry. If something like \texttt{\textbackslash raggedright} is given in the last column, then this command must be used in place of \texttt{\textbackslash \textbackslash}.

\texttt{\textbackslash tag\{\texttt{mark}\} [m][a]} \hspace{1em} (12.2.6) – 271

Within one of the \texttt{AMS-L\LaTeX{}} alignment environments, prints \texttt{\textbackslash mark} in place of the equation number, in parentheses. The \texttt{*}-form prints it without parentheses.

\texttt{\textbackslash tan [m]} \hspace{1em} (5.3.8) – 128

Command to produce the function name ‘\texttt{tan}’ in formulas.

\texttt{\textbackslash tanh [m]} \hspace{1em} (5.3.8) – 128

Command to produce the function name ‘\texttt{tanh}’ in formulas.

\texttt{\textbackslash tau [m]} produces $\tau$ \hspace{1em} (5.3.1) – 125

\texttt{\textbackslash binom\{\texttt{over}\}\{\texttt{under}\} [m][a]} \hspace{1em} (12.2.3) – 264

With the \texttt{amsmath} package, produces a binomial as \texttt{\binom} does, but in \texttt{textstyle} size.

\texttt{\textbackslash telephone\{\texttt{number}\}} \hspace{1em} (16.1), (16.2) – 352, 356

In the \texttt{letter} document class, enters the sender’s telephone number. In the standard \LaTeX{} \texttt{letter.sty}, \texttt{\textbackslash number} is only output if \texttt{\textbackslash address} has not been called. It is intended to be used in company letter styles such as \texttt{mpletter}.

\texttt{\textbackslash TeX produces $\text{\LaTeX{}}$} \hspace{1em} (2.1) – 19

\texttt{\textbackslash text\{\texttt{short.text}\} [m][a]} \hspace{1em} (12.2.1) – 259

When one of the packages \texttt{amsmath} or \texttt{amstext} has been loaded, this command prints \texttt{\textbackslash short.text} as normal text within a math formula. If used in subscripts or superscripts, automatic sizing takes place.
Appendix H. Command Summary

\textsym{name} ........................................ (D.6) – 462

An alternative means to produce certain special symbols that otherwise are only available in math mode or through ligature combinations:

\textbullet ; \textemdash ; \textendash ; \textexclamdown ; \textperiodcentered ; \textquestiondown ; \textquotedblleft ; \textquotedblright ; \textquoteleft ; \textquoteright ; \textvisiblespace ; \textasciicircum ; \textasciitilde ; \text\; ; \text| ; \text> ; \text<

\textbf{\textit{text}} ........................................ (4.1.4) – 65

This command sets its argument in a font in the current family and shape, but with the \textbf{\textit{bold}} series attribute. It is equivalent to \{\bfseries \text\}.\textbf{

\textcircled{\textit{char}}} ........................................ (D.6) – 462

Produces the specified character in a circle: \textcircled{s} = \textcircled{S}.

\textcolor{col spec}{\textit{text}} ................................. (6.2) – 167

A command made available with the color package. The \textit{text} is set in the specified color. The \textit{col spec} is the same as for \textcolor.

\textcompwordmark ........................................ (D.6) – 462

Prints an invisible character that may be used to break ligatures:

f\textcompwordmark i = fi

\textfloatsep ............................................. (7.3) – 172

The vertical spacing between floats at the top of the page and the following text or between text and floats at the bottom of the page. A new value is set with the \textfloatsep command.

\setlength{\textfloatsep}{20pt plus 2pt minus 4pt}

\textfraction ............................................. (7.3) – 172

The minimum fraction of a page containing text and floats that must be filled with text. A new value is set with \textfraction command.

\renewcommand{\textfraction}{decimal\_frac}

\textheight .............................................. (3.2.5) – 48

The total height reserved for the text on each page, excluding head and footlines. A new value may be assigned with the \textwidth command:

\setlength{\textheight}{45\baselineskip}
H.1. Brief description of the \LaTeX{} commands

\textit{text} .............................................. (4.1.4) - 65
This command sets its argument in a font in the current family and series, but with the \textit{} shape attribute. It is equivalent to {\itshape text}.

\textmd{text} .............................................. (4.1.4) - 65
This command sets its argument in a font in the current family and shape, \textbf{\textit{but with the medium series attribute.}} It is equivalent to {\mdseries text}.

\textnormal{text} ........................................ (4.1.3) - 65
This command sets its argument in the font with the default family, series, and shape attributes. It is equivalent to \textnormal{font \textit{text}}.

\textquotedbl ........................................... (G.4.5) - 503
When T1 encoding is active, prints the symbol ".

\textregistered produces \textregistered ................................... (D.6) - 462
\textsuperscript{char} ......................................... (D.6) - 462
Produces a superscript in the current text, rather than math, font: \textsuperscript{12} = 12.

\texttrademark produces \textregistered ................................... (D.6) - 462
\texttt{text} .............................................. (4.1.4) - 65
Sets text in a font in the current series and shape, but with the typewriter family attribute. It is equivalent to {\ttfamily text}.

\textstyle [m] ............................................ (5.5.2) - 146
Switches to font size \textstyle inside a math formula.

\textsuperscript{char} ......................................... (D.6) - 462
Sets text in a font in the current family and series, but with the \textbf{\textit{upright shape attribute.}} It is equivalent to {\upshape text}.
Appendix H. Command Summary

\textwidth \hspace{3.2.5} - 48

The total width reserved for the text on a page. For two-column formatting, this is the width of both columns plus the gap between them. A new value may be assigned with the \setlength command.

\tfrac{\text{numerator}}{\text{denominator}} \hspace{12.2.3} - 264

With the amsmath package, produces a fraction as \frac does, but in \textstyle size.

\TH \hspace{G.4.5} - 503

When T1 encoding is active, prints the character Þ.

\th \hspace{G.4.5} - 503

When T1 encoding is active, prints the character þ.

\thanks{\textit{footnote_text}} \hspace{3.3.1} - 54

Produces a footnote to an author's name on the title page when \maketitle is called.

\thecounter \hspace{8.1.4} - 183

Internal commands for formatting and printing counter values, making possible use of other counters. For example, \thesubsection might be defined to be \thesection.roman{subsection}. A new definition may be made with \renewcommand{\thecounter}{def}.

\Theta [m] produces \( \Theta \) \hspace{5.3.1} - 125

\theta [m] produces \( \theta \) \hspace{5.3.1} - 125

\thicklines \hspace{13.1.4} - 300

In the picture environment, this command sets all the sloping lines and arrows, circles, and ovals to be drawn with thicker than normal lines.

\thickspace [m][a] \hspace{12.2.5} - 269

With the amsmath package, this is an alias for \; ; a thick space in a math formula.

\thinspace \hspace{12.2.5} - 269

With the amsmath package, this is an alias for \,, a thin space in a math formula.

\thinlines \hspace{13.1.4} - 300

In the picture environment, resets the line thickness for sloping lines and arrows, circles, and ovals back to the standard value after \thicklines has been given.

\thinspace \hspace{12.2.5} - 269

With the amsmath package, this is an alias for \,, a thin space in a math formula.

\thisfancypage{cmds1}{cmds2} \hspace{4.7.9} - 95

With the fancybox package, places a framed box around the contents of the current page only; cmds1 exclude the head and footlines, cmds2 includes them. The arguments set box parameters like \fboxrule but must end with a box command like \shadowbox. Usually one set of cmds is left blank.
H.1. Brief description of the \LaTeX\ commands

\thispagestyle{style} \hfill (3.2) - 42
Changes the page style for the current page only. Possible values for \texttt{style} are: plain, empty, headings, and myheadings.

\tilde{x} \hfill (5.3.9) - 129
Produces a tilde (squiggle) over the math variable \( x \): \( \tilde{a} \).

\Tilde{x}[m][a] \hfill (12.2.2) - 263
With the \texttt{amsmath} package, can be used like \texttt{\tilde}, but with multiple \texttt{AMS-LaTeX} math accents the positioning will be correct.

\tiny \hfill (4.1.2) - 62
Switches to the smallest font size available \texttt{\tiny}, smaller than \texttt{\scriptsize}.

\times \hfill (5.3.3) - 125
\texttt{\times} produces \( \times \).

\title{text} \hfill (3.3.1) - 52
Enters the \texttt{text} for the title page that is produced by \texttt{\maketitle}.

\to \hfill (5.3.5) - 127
\texttt{\to} produces \( \rightarrow \).

\today \hfill (2.5.11), (D.4.2) - 27, 461
Prints the current date in the American fashion. This form may be changed to British or to that of other languages by redefining the command with the help of the internal \TeX\ commands \texttt{\day}, \texttt{\month}, and \texttt{\year}.

\top \hfill (5.3.6) - 127
\texttt{\top} produces \( \top \).

\topfigrule \hfill (7.3) - 173
A command that is executed after a float at the top of a page. It is normally defined to do nothing, but may be redefined to add a rule between the float and the main text. It must not add any net vertical spacing.

\renewcommand{\topfigrule}{\vspace*{-.4pt}\rule{\columnwidth}{.4pt}}

\topfraction \hfill (7.3) - 172
The maximum fraction of a page that may be occupied at the top by floats at the top of the page. A new value is assigned with \texttt{\renewcommand{\topfraction}{decimal.frac}}

\topmargin \hfill (3.2.5) - 48
The size of the margin from the top of the page to the page head. A new value may be assigned by the \texttt{\setlength} command:

\setlength{\topmargin}{0.5in}
Appendix H. Command Summary

*topnumber* .................................................. (7.3) – 171
The maximum number of floats that may appear at the top of a page.
A new value is assigned with:
\setcounter{topnumber}{num}

*\topsep* ..................................................... (4.4.2), (5.5.4) – 75, 149
The extra vertical spacing, in addition to *\parskip*, inserted at the begin-ning and end of a listing environment. When document class option *fleqn* has been chosen, it is also inserted at the beginning and end of displayed math formulas. A new value may be assigned with the *\setlength* command:
\setlength{\topsep}{4pt plus2pt minus2pt}

*\topskip* ...................................................... (3.2.5) – 48
The vertical distance from the top of the page body to the baseline of the first line of text. A new value may be assigned with the *\setlength* command:
\setlength{\topskip}{12pt}

*\totalheight* ................................................. (4.7.1) – 86
A length parameter equal to the total natural height of a box (height plus depth); it may only be used in the *width* specification of \makebox, \framebox, or \savebox, or in the *height* specification of a \parbox or a minipage environment.
\framebox[6\totalheight]{text}

*totalnumber* ............................................... (7.3) – 172
The total number of floats that may appear on a page regardless of their positions. A new value is assigned with:
\setcounter{totalnumber}{num}

*\triangle[m]* produces $\triangle$ ................................ (5.3.6) – 127
*\triangledown[m]* produces $\triangledown$ ................. (5.3.3) – 125
*\ttdefault* .................................................. (A.3.1) – 372
This command defines the family attribute that is selected with the *ttfamily* command. It may be redefined with *\renewcommand*:
\renewcommand{\ttdefault}{pcr}

*ttfamily* .................................................. (4.1.3), A.2) – 64, 371
This declaration switches to a font in the current series and shape, but with the typewriter family attribute.

*\twocolumn[\text]* ........................................... (3.2.7) – 51
Begins a new page and switches to two-column page format. The optional *text* is set in one column extending over the two columns.
H.1. Brief description of the \LaTeX commands

\typein{cmd}{message} ........................................ (9.1.3) - 211

Prints the \textit{message} to the monitor and stops the program, waiting for
a reply from the user. The text of the response is assigned to the \LaTeX
command named \texttt{@typein}, or to \texttt{cmd} if the optional argument has
been given. After the return key is pressed, the processing continues.
The typed-in text is inserted in place of \texttt{@typein} if the optional
argument was not given, otherwise it may be inserted as one pleases
with the \texttt{cmd} command.

\typeout{message} ........................................ (9.1.3) - 211

Prints the \textit{message} to the monitor and continues the processing. The
\textit{message} is also written to the .log file.

\texttt{u\{x\}} ........................................ (2.5.7) - 24

Produces a breve accent: \texttt{u\{o\} = \= o}.

\unboldmath ........................................ (5.4.9) - 143

Countermands the \texttt{boldmath} command. It must be given outside
of the math mode. Afterwards, formulas are set in standard ‘math
italics’ once more.

\underbrace{sub\_form} [m] ............................. (5.4.4) - 136

Produces a horizontal curly brace beneath the math formula
\textit{sub\_form}. Any following subscript will be placed centered below
the horizontal brace.

\begin{align*}
\underbrace{a+b} & = \underbrace{x+y+z}
\end{align*}

\underbrace{x+y+z}_{\xi\eta\zeta}

\underleftarrow{expr} [m][a] ............................ (12.2.2) - 262

With the amsmath package, places a long leftwards pointing arrow
beneath the mathematical expression \textit{expr}.

\underleftrightarrow{expr} [m][a] ............................ (12.2.2) - 262

With the amsmath package, places a long double arrow beneath the
mathematical expression \textit{expr}.

\underline{text} ........................................ (5.4.4) - 136

Underlines the \textit{text} in both math and normal text modes:
\begin{align*}
\underline{Text} & = \underline{Text}
\end{align*}

\underrightarrow{expr} [m][a] ............................ (12.2.2) - 262

With the amsmath package, places a long rightwards pointing arrow
beneath the mathematical expression \textit{expr}. 

Appendix H. Command Summary

\texttt{\textbackslash underset}\{\textit{char}\}\{\textbackslash symbol\}\texttt{[m][a]} . . . . . . . . . . (12.2.2) - 262

With the amsmath package, places \textit{char} below the math symbol \textbackslash symbol in subscript size.

\texttt{\textbackslash unitlength} . . . . . . . . . . . . . . . . (13.1.1) - 288

Defines the unit of length for the following picture environments.

A value is assigned with the \texttt{\textbackslash setlength} command:

\texttt{\textbackslash setlength}\{\textbackslash unitlength\}\{1.2cm\}

\texttt{\textbackslash unlhd}\texttt{[m]} produces \(\leq\) . . . . . . . . . . (5.3.3) - 125

\texttt{\textbackslash unrhd}\texttt{[m]} produces \(\geq\) . . . . . . . . . . (5.3.3) - 125

\texttt{\textbackslash Uparrow}\texttt{[m]} produces \(\uparrow\) . . . . . . . . . . (5.3.5) - 127

\texttt{\textbackslash uparrow}\texttt{[m]} produces \(\uparrow\) . . . . . . . . . . (5.3.5) - 127

\texttt{\textbackslash updefault} . . . . . . . . . . . . . . . . . . . . . . (A.3.1) - 372

This command defines the shape attribute that is selected with the \texttt{\textbackslash upshape} command. It may be redefined with \texttt{\textbackslash renewcommand}:

\texttt{\textbackslash renewcommand}\{\textbackslash updefault\}\{n\}

\texttt{\textbackslash Updownarrow}\texttt{[m]} produces \(\updownarrow\) . . . . . . . . . . (5.3.5) - 127

\texttt{\textbackslash updownarrow}\texttt{[m]} produces \(\updownarrow\) . . . . . . . . . . (5.3.5) - 127

\texttt{\textbackslash uplus}\texttt{[m]} produces \(\uplus\) . . . . . . . . . . . . . . . . (5.3.3) - 125

\texttt{\textbackslash upshape} . . . . . . . . . . . . . . . . . . . . . (4.1.3), (A.2) - 64, 371

This declaration switches to a font in the current family and series, but with the upright shape attribute.

\texttt{\textbackslash Upsilon}\texttt{[m]} produces \(\Upsilon\) . . . . . . . . . . (5.3.1) - 125

\texttt{\textbackslash upsilon}\texttt{[m]} produces \(\upsilon\) . . . . . . . . . . (5.3.1) - 125

\texttt{\textbackslash uproot}\{\textit{shift}\}\{m\}[a] . . . . . . . . . . . . . . . . (12.2.5) - 269

With the amsmath package, used in the index to a \texttt{\textbackslash sqrt} command to shift it slightly upwards. The \textit{shift} is a number specifying how many units to move it. Example:

\texttt{\textbackslash sqrt\{\textbackslash leftroot\{-1\}\textbackslash uproot\{3\}\textbackslash beta\}\{k\}}

\texttt{\textbackslash url}\{\textit{address}\} . . . . . . . . . . . . . . . . . . . . . (4.9.2) - 112

With the \texttt{url} package, prints \textit{address} literally, normally in typewriter font, with line breaks after non-letters, without hyphens. Intended for Internet and email addresses. With the \texttt{hyperref} package, \textit{address} becomes an active link in a PDF file. The \textit{address} argument may be delimited either by curly braces as usual, or by some other character, as is done with \texttt{\textbackslash verb}. Examples:

\texttt{\textbackslash url\{w_smith@xyz.com\} or \textbackslash url\{w_smith@xyz.com\}}

\texttt{\textbackslash urlstyle}\{\textit{style}\} . . . . . . . . . . . . . . . . . . . . . (4.9.2) - 112

With the \texttt{url} package, sets the typeface for subsequent \texttt{url} commands, where \textit{style} is one of \texttt{tt}, \texttt{rm}, \texttt{sf}, or \texttt{same}, for typewriter, Roman, sans serif, or unchanged font, respectively.
H.1. Brief description of the \LaTeX commands

\usebox{boxname} ........................................ (4.7.1) – 87
Inserts into the text the contents of the box that was saved with the
\sbox or \savebox command under the name \boxname, which has
been previously created with the \newsavebox command.

\usecounter{counter} ..................................... (4.4.1) – 75
Command in the list environment that specifies which counter is to
be employed in the standard labels with the \item commands. This
counter is incremented by one with each \item call.

\usefont{code}{fam}{ser}{shp} .......................... (A.1) – 369
Activates the font with the given set of attributes in the current size.
Is equivalent to selecting the given font attributes and then calling
\selectfont.

\usepackage[options]{packages}[version] [p] ............ (3.1.2) – 41
Loads one or more package files containing additional \LaTeX or \TeX
definitions. The files have the extension .sty. More than one package
may be specified in packages, the names being separated by commas.
Any options listed will be applied to all packages. Furthermore, any
options listed in the \documentclass command will also be applied
to the package files.

The optional version is a date, given in the form yyyy/mm/dd, as for
element 1994/08/01. If the date of the package file is earlier than
this, a warning message is printed.

Example:
\usepackage{bezier,ifthen}[1994/06/01]

\v{x} ........................................................... (2.5.7) – 24
Produces háček accent: \v{o} = ď.

\value{counter} ........................................... (8.1.3) – 183
The current value of the number stored in counter for use with com-
mands that require a number. It does not output this number. For ex-
ample, \setcounter{counter1}{\value{counter2}} sets counter1
to the same value as that of counter2.

\varepsilon[m] produces $\varepsilon$ .......................... (5.3.1) – 125
\varinjlim[m][a] produces ‘lim’ in formulas ........... (12.2.5) – 268
\varliminf[m][a] produces ‘lim’ in formulas ............ (12.2.5) – 268
\varlimsup[m][a] produces ‘lim’ in formulas ........... (12.2.5) – 268
\varphi[m] produces $\varphi$ ............................... (5.3.1) – 125
\varpi[m] produces $\varpi$ ............................... (5.3.1) – 125
\varprojlim[m][a] produces ‘lim’ in formulas .......... (12.2.5) – 268
\varepsilon[m] produces $\varepsilon$ .......................... (5.3.1) – 125
\varsigma[m] produces $\varsigma$ ........................... (5.3.1) – 125
\texttt{\textbackslash vartheta\ [m]} produces $\vartheta$ \hspace{1cm} (5.3.1) – 125

\texttt{\textbackslash vdash\ [m]} produces $\vdash$ \hspace{1cm} (5.3.6) – 127

\texttt{\textbackslash vdots\ [m]} produces $\vdots$ \hspace{1cm} (5.2.6) – 123

\texttt{\textbackslash vec\{x\} \ [m]} \hspace{1cm} (5.3.9) – 129

A vector symbol over the variable $x$: $\vec{a}$.

\texttt{\textbackslash Vec\{x\} \ [m]\[a]} \hspace{1cm} (12.2.2) – 263

With the \texttt{amsmath} package, can be used like $\vec{a}$, but with multiple \texttt{\LaTeX} math accents the positioning will be correct.

\texttt{\textbackslash vector(\Delta x, \Delta y)\{length\}} \hspace{1cm} (13.1.4) – 295

A picture element command within a \texttt{picture} environment for drawing horizontal and vertical arrows of any length as well as slanted arrows at a limited number of angles. For horizontal and vertical arrows, the \texttt{length} argument is the actual length in units of \texttt{\unitlength}. For slanted arrows, \texttt{length} is the length of the projection on to the x-axis (horizontal displacement). The slope is determined by the $(\Delta x, \Delta y)$ arguments, which take on integral values such that $-4 \leq \Delta x \leq 4$ and $-4 \leq \Delta y \leq 4$. This command is the argument of a $\texttt{\put}$ or $\texttt{\multiput}$ command.

\texttt{\textbackslash vee\ [m]} produces $\lor$ \hspace{1cm} (5.3.3) – 125

\texttt{\textbackslash verb\ | \ source\_text \ |} \hspace{1cm} (4.9) – 110

Everything that comes between the |...| symbols is output in the typewriter font exactly as is with no interpretation of special symbols or commands. Any symbol other than * may be used as the switch character, illustrated here as |, as long as it does not appear in \texttt{source\_text}.

\texttt{\textbackslash verb*\ | \ source\_text \ |} \hspace{1cm} (4.9) – 110

The same as $\texttt{\verb}$ except that blanks are made visible with the symbol $\omega$.

\texttt{\textbackslash vfill} \hspace{1cm} (2.7.3) – 32

A vertical rubber spacing with a natural length of zero that can be stretched to any value. Used to fill up parts of a page with blank spacing. This command is an abbreviation for $\texttt{\vspace{\fill}}$.

\texttt{\textbackslash visible} \hspace{1cm} (15.1.2) – 326

In \texttt{slides} class, a declaration that countermands a previous $\texttt{\invisible}$ command, making text printed again. It remains in effect until the end of the environment, or end of the curly braces, in which it was issued, or until $\texttt{\invisible}$ is given. It is used for making overlays.

\texttt{\textbackslash vline} \hspace{1cm} (4.8.1) – 97

Prints a vertical rule within the column entry of a table in the \texttt{tabular} environment.
H.1. Brief description of the \LaTeX commands

\texttt{voffset} .......................... 602, 603

Vertical offset of the output page from the printer border set by the
printer driver. This printer border is normally 1 inch from the top
ege of the paper. The standard value of \texttt{voffset} is 0 pt so that the
top reference margin of the page is identical with the printer margin.
A new value is assigned with the \texttt{setlength} command:

\texttt{setlength\{voffset\}\{-1in\}}

\texttt{voffset[\textit{current}][\textit{non-current}]\{key\}} ............ (9.2.4) - 215

With the \texttt{varioref} package, this is equivalent to on page
\texttt{pageref}\{key\}, unless \texttt{\label}\{key\} is on the current or adjacent
page, in which case appropriate text is automatically substituted,
e.g., 'on the next page'. The first optional argument is inserted for
the current page, the second for other pages.

\texttt{vref\{key\}} ................................. (9.2.4) - 215

With the \texttt{varioref} package, this is equivalent to \texttt{ref\{key\} on page}
\texttt{pageref\{key\}}, unless \texttt{\label\{key\} is on the current or adjacent
page, in which case appropriate text is automatically substituted for
the page specification, e.g., 'on the next page'.

\texttt{vspace\{height\}} ......................... (2.7.3) - 32

Produces vertical spacing of length \textit{height}. It is ignored if it occurs
at the beginning or end of a page.

\texttt{vspace*\{height\}} ......................... (2.7.3) - 32

Produces vertical spacing of length \textit{height} even at the beginning or
end of a page.

\texttt{wedge [m]} produces $\wedge$ ................................. (5.3.3) - 125

\texttt{whiledo\{test\}\{do	extunderscore text\}} ................. (8.3.5) - 193

A conditional command available when the standard package \texttt{ifthen}
has been loaded. The \texttt{do	extunderscore text} is inserted repeatedly as long as
the logical statement \texttt{test} evaluates to \langle \textit{true} \rangle. The logical state-
ment may be relational (two numbers with one of $<=$ $>$ between
them), an even-odd test (\texttt{\isodd\{number\}}, a comparison
of two texts (\texttt{\equal\{text1\}\{text2\}}, a comparison of two lengths
(\texttt{\lengthtest\{length1\}\op length2\}}, \op is one of $<=$ $>$), or a test
of a boolean switch (\texttt{\boolean\{switch\}). Switches are created with
\texttt{\newboolean\{switch\}} and set with \texttt{\setboolean\{switch\}\{value\},
where \textit{value} is \textit{true} or \textit{false}. Logical statements may be combined
with logical operators \texttt{\and}, \texttt{\or}, and \texttt{\not}, and grouped with \texttt{\{ and
\}}.

\texttt{widehat\{xyz\} [m]} ............................ (5.3.9) - 129

Produces a wide $\hat{\ }$ symbol over several characters:

\texttt{\widehat\{xyz\} = \hat{x}y}z.
\texttt{\textbackslash widetilde\{xyz\}[m]} \hspace{1cm} (5.3.9) - 129

Produces a wide \texttt{\textbackslash tilde} symbol over several characters:
\texttt{\textbackslash widetilde\{xyz\} = \widetilde{x\bar{y}z}.}

\texttt{\textbackslash width} \hspace{1cm} (4.7.1) - 86

A length parameter equal to the natural width of a box; it may only be used in the \texttt{width} specification of \texttt{\makebox}, \texttt{\framebox}, or \texttt{\savebox}, or in the \texttt{height} specification of a \texttt{\parbox} or a \texttt{\minipage} environment.
\texttt{\framebox[2\textbackslash width\{}\texttt{text}}} 

\texttt{\wp [m] produces \wp} \hspace{1cm} (5.3.6) - 127
\texttt{\wr [m] produces \wr} \hspace{1cm} (5.3.3) - 125

\texttt{\Xi [m] produces \Xi} \hspace{1cm} (5.3.1) - 125
\texttt{\xi [m] produces \xi} \hspace{1cm} (5.3.1) - 125

\texttt{\xleftarrow[\texttt{below}]{\texttt{above}}}[m][a] \hspace{1cm} (12.2.2) - 262

With the \texttt{amsmath} package, draws leftward pointing arrow with \texttt{above} printed over it in superscript size, and optionally \texttt{below} beneath it in subscript size.

\texttt{\xrightarrow[\texttt{below}]{\texttt{above}}}[m][a] \hspace{1cm} (12.2.2) - 262

With the \texttt{amsmath} package, draws rightward pointing arrow with \texttt{above} printed over it in superscript size, and optionally \texttt{below} beneath it in subscript size.

\texttt{\zeta [m] produces \zeta} \hspace{1cm} (5.3.1) - 125
## H.2 Summary tables and figures

### Table H.1  Font attribute commands (4.1.3) – p. 64.

<table>
<thead>
<tr>
<th>Command</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>\rmfamily</td>
<td>Roman</td>
</tr>
<tr>
<td>\sffamily</td>
<td>sans serif</td>
</tr>
<tr>
<td>\ttfamily</td>
<td>typewriter</td>
</tr>
<tr>
<td>\upshape</td>
<td>upright</td>
</tr>
<tr>
<td>\itshape</td>
<td>italic</td>
</tr>
<tr>
<td>\slshape</td>
<td>slanted</td>
</tr>
<tr>
<td>\scshape</td>
<td>SMALL CAPS</td>
</tr>
<tr>
<td>\mdseries</td>
<td>medium</td>
</tr>
<tr>
<td>\bfseries</td>
<td>bold face</td>
</tr>
</tbody>
</table>

### Table H.2  Math alphabet commands (5.4.2) – p. 133.

<table>
<thead>
<tr>
<th>Command</th>
<th>Alphabet</th>
</tr>
</thead>
<tbody>
<tr>
<td>\mathrm{text}</td>
<td>Roman</td>
</tr>
<tr>
<td>\mathsf{text}</td>
<td>sans serif</td>
</tr>
<tr>
<td>\mathnormal{text}</td>
<td>normal</td>
</tr>
<tr>
<td>\mathtt{text}</td>
<td>typewriter</td>
</tr>
<tr>
<td>\mathit{text}</td>
<td>italic</td>
</tr>
<tr>
<td>\mathbf{text}</td>
<td>boldface</td>
</tr>
<tr>
<td>\mathcal{text}</td>
<td>CAL</td>
</tr>
</tbody>
</table>

### Table H.3  Font sizes (4.1.2) – p. 62.

<table>
<thead>
<tr>
<th>Command</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>\tiny</td>
<td>smallest</td>
</tr>
<tr>
<td>\scriptsize</td>
<td>very small</td>
</tr>
<tr>
<td>\footnotesize</td>
<td>smaller</td>
</tr>
<tr>
<td>\small</td>
<td>small</td>
</tr>
<tr>
<td>\normalsize</td>
<td>normal</td>
</tr>
<tr>
<td>\large</td>
<td>large</td>
</tr>
<tr>
<td>\Large</td>
<td>larger</td>
</tr>
<tr>
<td>\LARGE</td>
<td>even larger</td>
</tr>
<tr>
<td>\huge</td>
<td>still larger</td>
</tr>
<tr>
<td>\Huge</td>
<td>largest</td>
</tr>
</tbody>
</table>

### Table H.4  \TeX 2.09 font declarations (F.2.1) – p. 485.

<table>
<thead>
<tr>
<th>Command</th>
<th>Font</th>
</tr>
</thead>
<tbody>
<tr>
<td>\rm</td>
<td>Roman</td>
</tr>
<tr>
<td>\bf</td>
<td>Bold Face</td>
</tr>
<tr>
<td>\tt</td>
<td>Typewriter</td>
</tr>
<tr>
<td>\it</td>
<td>Italic</td>
</tr>
<tr>
<td>\sl</td>
<td>Slanted</td>
</tr>
<tr>
<td>\sc</td>
<td>SMALL CAPS</td>
</tr>
<tr>
<td>\mit</td>
<td>ΓΠΦ</td>
</tr>
<tr>
<td>\cal</td>
<td>CAL</td>
</tr>
</tbody>
</table>
### Table H.5 Dimensions (2.4.1) – p. 21.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>mm</td>
<td>millimeter</td>
</tr>
<tr>
<td>cm</td>
<td>centimeter</td>
</tr>
<tr>
<td>in</td>
<td>inch (1 in = 2.54 cm)</td>
</tr>
<tr>
<td>pt</td>
<td>point (1 in = 72.27 pt)</td>
</tr>
<tr>
<td>pc</td>
<td>pica (1 pc = 12 pt)</td>
</tr>
<tr>
<td>em</td>
<td>The current width of a capital M</td>
</tr>
<tr>
<td>ex</td>
<td>The current height of the letter x</td>
</tr>
</tbody>
</table>

### Table H.6 Accents (2.5.7) – p. 24.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Accent</th>
</tr>
</thead>
<tbody>
<tr>
<td>`o</td>
<td>'o</td>
</tr>
<tr>
<td>¨o</td>
<td>&quot;o</td>
</tr>
<tr>
<td>¯o</td>
<td>=o</td>
</tr>
<tr>
<td>¯o</td>
<td>=o</td>
</tr>
<tr>
<td>òo</td>
<td>{o}</td>
</tr>
<tr>
<td>òo</td>
<td>{o}</td>
</tr>
<tr>
<td>òo</td>
<td>{o}</td>
</tr>
</tbody>
</table>

### Table H.7 Special letters from other languages (2.5.6) – p. 24.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Letter</th>
</tr>
</thead>
<tbody>
<tr>
<td>æ</td>
<td>ae</td>
</tr>
<tr>
<td>œ</td>
<td>oe</td>
</tr>
<tr>
<td>Ø</td>
<td>{O}</td>
</tr>
<tr>
<td>ø</td>
<td>{o}</td>
</tr>
</tbody>
</table>

### Table H.8 Special symbols (2.5.5) – 23.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>†</td>
<td>\dag</td>
</tr>
<tr>
<td>§</td>
<td>\S</td>
</tr>
<tr>
<td>ⓒ</td>
<td>\copyright</td>
</tr>
<tr>
<td>‡</td>
<td>\ddag</td>
</tr>
<tr>
<td>¶</td>
<td>\P</td>
</tr>
<tr>
<td>£</td>
<td>\pounds</td>
</tr>
</tbody>
</table>

### Table H.9 Command symbols (2.5.4) – 23.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>$</td>
</tr>
<tr>
<td>%</td>
<td>%</td>
</tr>
</tbody>
</table>
| \{    | \{
| \_    | \_    |
| \&    | \&      |
| \#    | \#      |

### Table H.10 Greek letters (5.3.1) – p. 125.

#### Lower case letters

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>\alpha</td>
</tr>
<tr>
<td>β</td>
<td>\beta</td>
</tr>
<tr>
<td>γ</td>
<td>\gamma</td>
</tr>
<tr>
<td>δ</td>
<td>\delta</td>
</tr>
<tr>
<td>ε</td>
<td>\epsilon</td>
</tr>
<tr>
<td>η</td>
<td>\eta</td>
</tr>
<tr>
<td>ζ</td>
<td>\zeta</td>
</tr>
<tr>
<td>θ</td>
<td>\theta</td>
</tr>
<tr>
<td>ι</td>
<td>\iota</td>
</tr>
<tr>
<td>κ</td>
<td>\kappa</td>
</tr>
<tr>
<td>λ</td>
<td>\lambda</td>
</tr>
<tr>
<td>μ</td>
<td>\mu</td>
</tr>
<tr>
<td>ν</td>
<td>\nu</td>
</tr>
<tr>
<td>ρ</td>
<td>\rho</td>
</tr>
<tr>
<td>σ</td>
<td>\sigma</td>
</tr>
<tr>
<td>τ</td>
<td>\tau</td>
</tr>
<tr>
<td>υ</td>
<td>\upsilon</td>
</tr>
<tr>
<td>ψ</td>
<td>\psi</td>
</tr>
<tr>
<td>ω</td>
<td>\omega</td>
</tr>
</tbody>
</table>

#### Upper case letters

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Γ</td>
<td>\Gamma</td>
</tr>
<tr>
<td>Δ</td>
<td>\Delta</td>
</tr>
<tr>
<td>Θ</td>
<td>\Theta</td>
</tr>
<tr>
<td>Λ</td>
<td>\Lambda</td>
</tr>
<tr>
<td>Σ</td>
<td>\Sigma</td>
</tr>
<tr>
<td>Ψ</td>
<td>\Psi</td>
</tr>
<tr>
<td>Ω</td>
<td>\Omega</td>
</tr>
</tbody>
</table>
Table H.11  Binary operation symbols (5.3.3) – p. 125.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>LaTeX</th>
</tr>
</thead>
<tbody>
<tr>
<td>±</td>
<td>\pm</td>
</tr>
<tr>
<td>†</td>
<td>\mp</td>
</tr>
<tr>
<td>×</td>
<td>\times</td>
</tr>
<tr>
<td>÷</td>
<td>\div</td>
</tr>
<tr>
<td>·</td>
<td>\cdot</td>
</tr>
<tr>
<td>†</td>
<td>\ast</td>
</tr>
<tr>
<td>‡</td>
<td>\dagger</td>
</tr>
<tr>
<td>⊢</td>
<td>\models</td>
</tr>
<tr>
<td>⊥</td>
<td>\perp</td>
</tr>
</tbody>
</table>

Table H.12  Relational symbols (5.3.4) – p. 126.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>LaTeX</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤</td>
<td>\le \leq</td>
</tr>
<tr>
<td>≦</td>
<td>\ll</td>
</tr>
<tr>
<td>≃</td>
<td>\sim</td>
</tr>
<tr>
<td>≈</td>
<td>\equiv</td>
</tr>
<tr>
<td>⊆</td>
<td>\in</td>
</tr>
<tr>
<td>⊕</td>
<td>\vdash</td>
</tr>
<tr>
<td>≠</td>
<td>\not&lt;</td>
</tr>
</tbody>
</table>

Table H.13  Negated relational symbols (5.3.4) – p. 127.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>LaTeX</th>
</tr>
</thead>
<tbody>
<tr>
<td>≠</td>
<td>\not&lt;</td>
</tr>
<tr>
<td>≠</td>
<td>\not\le</td>
</tr>
<tr>
<td>≠</td>
<td>\not\prec</td>
</tr>
<tr>
<td>≠</td>
<td>\not\subseteq</td>
</tr>
<tr>
<td>≠</td>
<td>\not\sqsubset</td>
</tr>
<tr>
<td>≠</td>
<td>\not\in</td>
</tr>
</tbody>
</table>

Table H.14  Brackets (5.4.1) – p. 132.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>LaTeX</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>\lfloor</td>
</tr>
<tr>
<td>[</td>
<td>\lceil</td>
</tr>
<tr>
<td>{</td>
<td>\langle</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>/ \backslash</td>
<td>\downarrow</td>
</tr>
<tr>
<td>′</td>
<td>\uparrow</td>
</tr>
<tr>
<td>″</td>
<td>\updownarrow</td>
</tr>
</tbody>
</table>

Table H.15  Arrows  (5.3.5) – p. 127.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\leftarrow</td>
<td>left arrow</td>
</tr>
<tr>
<td>\gets</td>
<td>left arrow</td>
</tr>
<tr>
<td>\longleftarrow</td>
<td>left arrow</td>
</tr>
<tr>
<td>\uparrow</td>
<td>up arrow</td>
</tr>
<tr>
<td>\Leftarrow</td>
<td>left arrow</td>
</tr>
<tr>
<td>\Leftarrow</td>
<td>left arrow</td>
</tr>
<tr>
<td>\longrightarrow</td>
<td>right arrow</td>
</tr>
<tr>
<td>\downarrow</td>
<td>down arrow</td>
</tr>
<tr>
<td>\Rightarrow</td>
<td>right arrow</td>
</tr>
<tr>
<td>\Leftarrow</td>
<td>left arrow</td>
</tr>
<tr>
<td>\Longleftarrow</td>
<td>left arrow</td>
</tr>
<tr>
<td>\Uparrow</td>
<td>up arrow</td>
</tr>
<tr>
<td>\hookrightarrow</td>
<td>hook arrow</td>
</tr>
<tr>
<td>\leftharpoonup</td>
<td>hook arrow</td>
</tr>
<tr>
<td>\rightleftharpoons</td>
<td>rightleftharpoons</td>
</tr>
</tbody>
</table>

Table H.16  Miscellaneous symbols  (5.3.6) – p. 127.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\aleph</td>
<td>aleph</td>
</tr>
<tr>
<td>\prime</td>
<td>prime</td>
</tr>
<tr>
<td>\forall</td>
<td>for all</td>
</tr>
<tr>
<td>\Box</td>
<td>box</td>
</tr>
<tr>
<td>\hbar</td>
<td>h bar</td>
</tr>
<tr>
<td>\emptyset</td>
<td>empty set</td>
</tr>
<tr>
<td>\exists</td>
<td>exists</td>
</tr>
<tr>
<td>\Diamond</td>
<td>diamond</td>
</tr>
<tr>
<td>\nabla</td>
<td>nabla</td>
</tr>
<tr>
<td>\flat</td>
<td>flat</td>
</tr>
<tr>
<td>\clubsuit</td>
<td>clubsuit</td>
</tr>
<tr>
<td>\jmath</td>
<td>j math</td>
</tr>
<tr>
<td>\natural</td>
<td>natural</td>
</tr>
<tr>
<td>\wp</td>
<td>wp</td>
</tr>
<tr>
<td>\top</td>
<td>top</td>
</tr>
<tr>
<td>\sharp</td>
<td>sharp</td>
</tr>
<tr>
<td>\heartsuit</td>
<td>heartsuit</td>
</tr>
<tr>
<td>\Re</td>
<td>Re</td>
</tr>
<tr>
<td>\bot</td>
<td>bot</td>
</tr>
<tr>
<td>\spadesuit</td>
<td>spadesuit</td>
</tr>
<tr>
<td>\Im</td>
<td>Im</td>
</tr>
<tr>
<td>\vdash</td>
<td>dash</td>
</tr>
<tr>
<td>\angle</td>
<td>angle</td>
</tr>
<tr>
<td>\Join</td>
<td>join</td>
</tr>
<tr>
<td>\infty</td>
<td>infty</td>
</tr>
</tbody>
</table>

The underlined commands in Tables H.11, H.12, H.15, and H.16 can only be used in \LaTeX 2.0e if one of the packages \texttt{latexsym} or \texttt{amsfonts} has been loaded.

Table H.17  Mathematical symbols in two sizes  (5.3.7) – p. 128.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\sum</td>
<td>sum</td>
</tr>
<tr>
<td>\bigcap</td>
<td>big cap</td>
</tr>
<tr>
<td>\bigcup</td>
<td>big cup</td>
</tr>
<tr>
<td>\bigotimes</td>
<td>big otimes</td>
</tr>
<tr>
<td>\prod</td>
<td>prod</td>
</tr>
<tr>
<td>\coprod</td>
<td>coprod</td>
</tr>
</tbody>
</table>

Table H.18  Function names  (5.3.8) – p. 128.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\arccos</td>
<td>arccos</td>
</tr>
<tr>
<td>\cosh</td>
<td>cosh</td>
</tr>
<tr>
<td>\det</td>
<td>det</td>
</tr>
<tr>
<td>\inf</td>
<td>inf</td>
</tr>
<tr>
<td>\limsup</td>
<td>limsup</td>
</tr>
<tr>
<td>\Pr</td>
<td>Pr</td>
</tr>
<tr>
<td>\tan</td>
<td>tan</td>
</tr>
<tr>
<td>\arcsin</td>
<td>arcsin</td>
</tr>
<tr>
<td>\cot</td>
<td>cot</td>
</tr>
<tr>
<td>\dim</td>
<td>dim</td>
</tr>
<tr>
<td>\ker</td>
<td>ker</td>
</tr>
<tr>
<td>\ln</td>
<td>ln</td>
</tr>
<tr>
<td>\sec</td>
<td>sec</td>
</tr>
<tr>
<td>\tanh</td>
<td>tanh</td>
</tr>
<tr>
<td>\arctan</td>
<td>arctan</td>
</tr>
<tr>
<td>\coth</td>
<td>coth</td>
</tr>
<tr>
<td>\exp</td>
<td>exp</td>
</tr>
<tr>
<td>\log</td>
<td>log</td>
</tr>
<tr>
<td>\sin</td>
<td>sin</td>
</tr>
<tr>
<td>\arg</td>
<td>arg</td>
</tr>
<tr>
<td>\csc</td>
<td>csc</td>
</tr>
<tr>
<td>\gcd</td>
<td>gcd</td>
</tr>
<tr>
<td>\lim</td>
<td>lim</td>
</tr>
<tr>
<td>\max</td>
<td>max</td>
</tr>
<tr>
<td>\sinh</td>
<td>sinh</td>
</tr>
<tr>
<td>\cos</td>
<td>cos</td>
</tr>
<tr>
<td>\deg</td>
<td>deg</td>
</tr>
<tr>
<td>\hom</td>
<td>hom</td>
</tr>
<tr>
<td>\liminf</td>
<td>lim inf</td>
</tr>
<tr>
<td>\min</td>
<td>min</td>
</tr>
<tr>
<td>\sup</td>
<td>sup</td>
</tr>
</tbody>
</table>
Table H.19  Math accents  (5.3.9) - p. 129.

\[ \hat{a} \quad \breve{a} \quad \grave{a} \quad \bar{a} \quad \check{a} \quad \acute{a} \quad \tilde{a} \quad \vec{a} \quad \dot{a} \quad \ddot{a} \quad \mathring{a} \]

The following symbols are made available with the package amssymb.

<table>
<thead>
<tr>
<th>Table H.20</th>
<th>( A_M ) arrows</th>
</tr>
</thead>
<tbody>
<tr>
<td>\to</td>
<td>\dashrightarrow</td>
</tr>
<tr>
<td>\iff</td>
<td>\Lleftarrow</td>
</tr>
<tr>
<td>\leftarrowtail</td>
<td>\looparrowleft</td>
</tr>
<tr>
<td>\leftthreetimes</td>
<td>\circlearrowright</td>
</tr>
<tr>
<td>\leftarrowtail</td>
<td>\looparrowleft</td>
</tr>
<tr>
<td>\leftthreetimes</td>
<td>\circlearrowright</td>
</tr>
<tr>
<td>\leftarrow</td>
<td>\looparrowleft</td>
</tr>
<tr>
<td>\leftrightarrow</td>
<td>\rightsquigarrow</td>
</tr>
</tbody>
</table>

Negated arrows

<table>
<thead>
<tr>
<th>Table H.21</th>
<th>( A_M ) binary operation symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>\plus</td>
<td>\smallsetminus</td>
</tr>
<tr>
<td>\cup</td>
<td>\cap</td>
</tr>
<tr>
<td>\barwedge</td>
<td>\veebar</td>
</tr>
<tr>
<td>\doublebarwedge</td>
<td>\boxminus</td>
</tr>
<tr>
<td>\boxtimes</td>
<td>\boxdot</td>
</tr>
<tr>
<td>\boxplus</td>
<td>\divideontimes</td>
</tr>
<tr>
<td>\times</td>
<td>\rtimes</td>
</tr>
<tr>
<td>\leftthreetimes</td>
<td>\rightthreetimes</td>
</tr>
<tr>
<td>\curlywedge</td>
<td>\curlyvee</td>
</tr>
<tr>
<td>\circledast</td>
<td>\circledcirc</td>
</tr>
<tr>
<td>\centerdot</td>
<td>\intercal</td>
</tr>
</tbody>
</table>
### Table H.22 \( \mathcal{AMS} \) Greek and Hebrew letters

<table>
<thead>
<tr>
<th>( F )</th>
<th>( \digamma )</th>
<th>( \kappa )</th>
<th>( \varkappa )</th>
<th>( \beth )</th>
<th>( \daleth )</th>
<th>( \gimel )</th>
</tr>
</thead>
</table>

### Table H.23 \( \mathcal{AMS} \) delimiters

<table>
<thead>
<tr>
<th>( \ulcorner )</th>
<th>( \urcorner )</th>
<th>( \llcorner )</th>
<th>( \lrcorner )</th>
</tr>
</thead>
</table>

### Table H.24 \( \mathcal{AMS} \) relational symbols

<table>
<thead>
<tr>
<th>( \leqq )</th>
<th>( \leqslant )</th>
<th>( \lessim )</th>
<th>( \lesseqgtr )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \preccurlyeq )</td>
<td>( \precsim )</td>
<td>( \preceq )</td>
<td>( \preccurlyeq )</td>
</tr>
<tr>
<td>( \precapprox )</td>
<td>( \precapprox )</td>
<td>( \preceq )</td>
<td>( \preceq )</td>
</tr>
<tr>
<td>( \vartriangleleft )</td>
<td>( \vartriangleleft )</td>
<td>( \vartriangleright )</td>
<td>( \vartriangleright )</td>
</tr>
<tr>
<td>( \vDash )</td>
<td>( \vDash )</td>
<td>( \backepsilon )</td>
<td>( \backepsilon )</td>
</tr>
<tr>
<td>( \blacktriangleleft )</td>
<td>( \blacktriangleright )</td>
<td>( \blacktriangleright )</td>
<td>( \blacktriangleright )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \therefore )</th>
<th>( \therefore )</th>
<th>( \therefore )</th>
<th>( \therefore )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \because )</td>
<td>( \because )</td>
<td>( \because )</td>
<td>( \because )</td>
</tr>
</tbody>
</table>
Table H.25 $\mathcal{AMS}$ negated relational symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>\nless</td>
<td>\nleq</td>
</tr>
<tr>
<td>\nleqslant</td>
<td>\nleqq</td>
</tr>
<tr>
<td>\lineq</td>
<td>\lineqq</td>
</tr>
<tr>
<td>\lvertneqq</td>
<td>\lnsim</td>
</tr>
<tr>
<td>\lnapprox</td>
<td>\nprec</td>
</tr>
<tr>
<td>\npreceq</td>
<td>\precnsim</td>
</tr>
<tr>
<td>\nshortapprox</td>
<td>\nsim</td>
</tr>
<tr>
<td>\nvDash</td>
<td>\nvDash</td>
</tr>
<tr>
<td>\ntriangledown</td>
<td>\ntriangledown</td>
</tr>
<tr>
<td>\nsupseteq</td>
<td>\supseteq</td>
</tr>
<tr>
<td>\supseteqq</td>
<td>\supseteqq</td>
</tr>
<tr>
<td>\supsetneq</td>
<td>\supsetneq</td>
</tr>
<tr>
<td>\supsetneqq</td>
<td>\supsetneqq</td>
</tr>
<tr>
<td>\gtr</td>
<td>\gneq</td>
</tr>
<tr>
<td>\gneqq</td>
<td>\gneqq</td>
</tr>
<tr>
<td>\gnapprox</td>
<td>\gnapprox</td>
</tr>
<tr>
<td>\nsucceq</td>
<td>\nsucceq</td>
</tr>
<tr>
<td>\succnsim</td>
<td>\succnsim</td>
</tr>
<tr>
<td>\nshortparallel</td>
<td>\nparallel</td>
</tr>
<tr>
<td>\nvDash</td>
<td>\nvDash</td>
</tr>
<tr>
<td>\ntriangleright</td>
<td>\ntriangleright</td>
</tr>
<tr>
<td>\nsupseteq</td>
<td>\supseteq</td>
</tr>
<tr>
<td>\supseteqq</td>
<td>\supseteqq</td>
</tr>
<tr>
<td>\supsetneq</td>
<td>\supsetneq</td>
</tr>
<tr>
<td>\supsetneqq</td>
<td>\supsetneqq</td>
</tr>
</tbody>
</table>

Table H.26 Miscellaneous $\mathcal{AMS}$ symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hbar$</td>
<td>$\hbar$</td>
</tr>
<tr>
<td>$\vartriangle$</td>
<td>$\vartriangle$</td>
</tr>
<tr>
<td>$\square$</td>
<td>$\square$</td>
</tr>
<tr>
<td>$\circledS$</td>
<td>$\circledS$</td>
</tr>
<tr>
<td>$\measuredangle$</td>
<td>$\measuredangle$</td>
</tr>
<tr>
<td>$\mho$</td>
<td>$\mho$</td>
</tr>
<tr>
<td>$\Game$</td>
<td>$\Game$</td>
</tr>
<tr>
<td>$\backprime$</td>
<td>$\backprime$</td>
</tr>
<tr>
<td>$\blacktriangle$</td>
<td>$\blacktriangle$</td>
</tr>
<tr>
<td>$\blacklozenge$</td>
<td>$\blacklozenge$</td>
</tr>
<tr>
<td>$\bigstar$</td>
<td>$\bigstar$</td>
</tr>
<tr>
<td>$\complement$</td>
<td>$\complement$</td>
</tr>
<tr>
<td>$\diagup$</td>
<td>$\diagup$</td>
</tr>
<tr>
<td>$\diagdown$</td>
<td>$\diagdown$</td>
</tr>
</tbody>
</table>
Figure H.1  Single column page format
Figure H.2  Double column page format
Remarks on the page format figures

The reference margins in the \LaTeX processing are shifted from the *logical* margins by the amounts $\hoffset$ and $\voffset$. These in turn are displaced from the *physical* margins by $h$ and $v$ in the DVI driver. The default values for $\hoffset$ and $\voffset$ are 0 pt, so that the reference margins are equal to the *logical* ones. The usual values for $h$ and $v$ are 1 inch. Thus the logical page margins on the left and at the top are shifted 1 inch from the physical edge of the paper. The user may alter this by changing the values of $\hoffset$ and $\voffset$.

\LaTeXe recognizes the parameters $\paperwidth$ and $\paperheight$ which contain the full dimensions of the paper, including the 1 inch margins. These are set by the paper size option to the class specification.

The parameter $\footheight$ was specified but never used in \LaTeX 2.09; it has been dropped from \LaTeXe.

---

**Preceding text**

![Diagram](image)

**Following text**

*Figure H.3* Format of the *list* environment (4.4.2) - p. 75.

**Note 1:** The default values for the three parameters $\itemindent$, $\listparindent$, $\rightmargin$ are 0 pt.

**Note 2:** The default values in the *trivlist* environment are 0 pt for $\itemindent$, $\leftmargin$, $\rightmargin$, and $\labelwidth$; on the other hand, $\parsep$ and $\listparindent$ are assigned the respective values of $\parskip$ and $\parindent$. 
Bibliography


Botway L. and Biemesderfer C. (1985). \textit{\LaTeX Command Summary}. Providence RI: \TeX Users Group

Eijkhout V. (1992). \textit{\TeX by Topic, a \TeXnician’s Reference}. Harlow: Addison-Wesley


Marchal, B. *XML by Example*. Indianapolis IN: Que-Programming


Samuel A. L. (1985). *First Grade \TeX: A Beginner’s \TeX Manual*. Providence, RI: \TeX Users Group


Sojka P., Thanh H. T. and Zlatuška J. The joy of \TeX\textsuperscript{2}\texttt{PDF}—Acrobatics with an alternative to DVI format. *TUGboat*, 17(3), 244–251


Urban M. (1986). *An Introduction to \LaTeX*. Providence RI: \TeX Users Group

Williamson, H. A. *XML from A to Z*. Redmond WA: Redmond Technology Press
Index

For purposes of alphabetization, the backslash character \ is ignored at the start of an entry. Otherwise, the ordering is by the ASCII sequence.

Bold page numbers indicate the place where the command or concept is introduced, explained, or defined. Slanted page numbers refer to the Command Summary, Appendix H.

The keyword index is set up with main and two sub-entries. If a keyword cannot be found as a main entry, one should try to find it as a sub-entry to some more general term. Such major topics are

AMS-LaTeX, bibliographic database, bibliography, box, command, command (user-defined), cross-reference, error messages, exercises, file types, float, fonts, footnote, formula, hyphenation, LaTeX, LaTeX counters, letter, line breaking, lists, package, page breaking, page formatting, page numbering, page style, picture, PostScript, programming, sectioning, seminar, slides, spacing, style parameter, symbols, tabbing, table, table examples, \TeX, text.
\(=\) (\textasciitilde\ accent), 24, 510
\(>\) (tabbing), 81, 82–4, 420, 510
\(?'\), 24, 510
@\texttt{BibTeX}, 312–14, 510
@\texttt{MakeIndex}, 226, 510
@, 29, 510
@-expression, 96, 101, 110
@date, 363
@evenfoot, 362, 455
@evenhead, 362, 455
@for, 450
@ifnextchar, 449
@ifstar, 449
@ifundefined, 449
@latexerr, 439
@namedef, 449
@nameuse, 449, 458
@oddfoot, 362, 455
@oddhead, 362, 455
@onlypreamble, 455
@seccntformat, 457
@startsection, 456
@warning, 439
\[, 120, 124, 416, 425, 510, 549
\[, 18, 132, 426, 510
\\, 31, 32, 53, 67, 68, 81, 84, 97, 99,
\quad, 104, 271, 297, 352, 419,
\quad, 425, 434, 510
\\*, 31, 279, 510
\\, 120, 124, 416, 510
\\, 18, 132, 510
\^, 23
\`, 17, 121, 123, 424, 511
\^\ (\textasciitilde\ accent), 24, 511
\~, 23, 414, 511
\~, 17, 121, 123, 413, 424, 511
\('\) (tabbing), 83, 511
\', 23
\', 23
\'\ (\textasciitilde\ accent), 24, 511
\{, 17, 18, 312, 425, 511
\{, 23, 120, 132, 511
\{, 126, 132, 270, 511
\{ (\texttt{MakeIndex}), 227, 511
\{, 126, 127, 132, 270, 511
\{ (table), 96
\{, 17, 18, 312, 425, 426, 428, 511
\{, 23, 120, 132, 511
\^, 17, 22, 28, 511
\^\ (\textasciitilde\ accent), 24, 511
10pt option, 38, 61
11pt option, 38, 61
12pt option, 38, 61, 456
8r.enc, 235, 236
\., 22, 28, 507
\' (tabbing), 83, 512
a4paper option, 38, 359, 447, 452, 456
a5paper option, 38
\Wa, 24, 512
\aa, 24, 512
abbrv bibliography style, 311
abbrvnat bibliography style, 221, 311
\abovedisplayshortskip, 149, 512
\abovedisplayskip, 149, 512
Abrahams, P. W., 605
abstract, 55
abstract environment, 55, 516
\abstractname, 460, 512
accents
in tabbing environment, 83
in other languages, 24
mathematical, 129
Acrobat, 16, 236, 237, 242, 247, 249
Acrobat Distiller, 237, 239
\Acrobatmenu (hyperref), 247, 342
\acute, 129, 263, 512
\Acute (\textasciitilde\mathcal{A\mathcal{M}}), 263, 512
\addButton (pdfscreen), 342
\addcontentsline, 59, 60, 174,
\quad, 427, 436, 512
\adddialect (babel), 255
\addlanguage (babel), 255
\address, 351, 352, 353, 356, 512
\addtime (slides), 327, 512
\addtoartlength (seminar), 332
\addtocontents, 59, 60, 174,
\quad, 427, 436, 512
\addtocounter, 21, 110, 115, 182,
\quad, 183, 418, 513
\addtotocounter, 21, 110, 115, 182,
\quad, 183, 418, 513
\addtolength, 184, 200, 453, 513
\addtoslide (seminar), 332
\addtoslidereset (seminar), 337
\Hat, 263, 546
\hdotsfor, 267
\idotsint, 260, 548
\iiiiint, 260, 551
\iiint, 260, 551
\iint, 260, 551
\intertext, 259, 550
\leftroot, 269, 554
limits
multiline, 261
special, 261
\lVert, 270, 557
\lvert, 270, 557
math symbols, 285
\mathbb, 284
matrix, 266
matrix environment, 266, 519
\medspace, 269, 560
\mod, 268, 560
\mspace, 269, 560
multiline equations, 270–7
page breaks, 278
multiple integrals, 260
multiline environment, 270, 271, 520
\multlinegap, 271, 561
\negmedspace, 269, 561
\negthickspace, 269, 561
\negthinspace, 269, 561
\notag, 271, 565
\numberwithin, 277, 565
options, 279
\overleftarrow, 262, 566
\overleftbracket, 262, 566
\overrightarrow, 262, 567
\overset, 262, 567
pmatrix environment, 266, 521
\pmb, 258, 570
\pod, 268, 570
\raisetag, 271
roots, 269
\rVert, 270, 576
\rvert, 270, 576
\shoveleft, 271
\shoveright, 271
\sideset, 262, 579
smallmatrix environment, 267
\smash, 269, 580
split environment, 270, 272, 276, 521
subarray environment, 261, 522
subequations environment, 277, 522
\substack, 260, 581
\tag, 271, 573, 583
\tbinom, 265, 583
\text, 259, 583
\tfrac, 264, 586
theorems, 280
\theoremstyle, 281
\thickspace, 269, 586
\thinspace, 269, 586
\Tilde, 263, 587
\underleftarrow, 262, 589
\underleftbracket, 262, 589
\underrightarrow, 262, 589
\underset, 262, 590
upright references, 282
\uproot, 269, 590
\varinjlim, 268, 591
\varliminf, 268, 591
\varlimsup, 268, 591
\varprojlim, 268, 591
\Vec, 263, 592
vertical bars, 270
Vmatrix environment, 266, 524
vmatrix environment, 266, 524
\xleftarrow, 263, 594
\xrightarrow, 263, 594
amsldoc.tex, 257, 282
amsmath package, 192, 258
amsopn package, 258, 267–8
amsproc class, 258
amssymb package, 284, 285
\AMS\TeX, 257
amstex format, 257
amstext package, 258–9
amsthm package, 81, 258, 280–2, 395
\and, 53, 513
INDEX  611

\angle, 127, 513
ANSI coding, 26, 462
apalike package, 220
appendix, 57
appendix environment, 516
appendix, 57
appendixname, 460, 513
Apple Macintosh coding, 26, 462
approx, 126, 514
Arabic page numbering, 45
arabic, 73, 75, 77, 78, 114, 183, 191, 277, 514

argin(EEPIC), 306
arccos, 128, 514
arcsin, 128, 514
arctan, 128, 514
arg, 128, 514
argument, command, 18
array environment, 95, 107, 134, 146, 147, 149, 150, 266, 394, 417, 418, 425, 428, 429, 516
array package, 107, 394
arraycolsep, 98, 149, 514
arrayrulewidth, 98, 514
arrays, 133–6
arraystretch, 98, 514
arrows, 127
extended (AM\S), 262
Arseneau, Donald, 112, 224
article class, 37, 41, 55, 113, 131, 173, 191, 218, 225, 277, 391, 397, 454, 459, 471
article.cls, 397, 430
article.sty (\TeX{} 2.09), 430
articlemag (seminar), 332
ASCII, 5, 14, 17, 228, 231, 382, 487
askforoverwritefalse (DocStrip), 465
ast, 126, 514
asymp, 126, 514
AtBeginDocument, 444, 472, 514
AtEndDocument, 444, 514
AtEndOfClass, 444, 515
AtEndOfPackage, 444, 515
atop (\TeX{}), 137, 146, 147, 189
attributes, font, 64–5, 368–72
defaults, 372
author, 52, 53, 54, 69, 431, 515
aux file, 209, 210, 214, 396, 397, 418, 427, 430, 436
auxiliary file, 214
avant package, 234
AvantGarde, PostScript font, 234, 504
axioms, 80
b5paper option, 38
\b (accent), 24, 515
babel package, 252–6
babel.def (babel), 253
babel, multilingual \TeX{}, 216, 252–6, 385, 460
background package, 345–6
backgroundcolor (pdfscreen), 342
backmatter, 57, 210, 515
backslash, 127, 132, 515
Bahasa, 252
bakoma fonts, 506
bakomaextra.map, 506
\bar, 129, 263, 515
\Bar (AM\S), 263, 515
Barroca, Leonor, 159
baseline, 46, 63
baselineskip, 63, 362, 369, 515
basinestretch, 46, 47, 63, 515
Basque, 252
batchinput (DocStrip), 466
.bbl file, 219, 224, 310, 397
Beccari, Claudio, 605
\begin, 19, 64, 416–18, 516
\belowdisplayshortskip, 149, 524
\belowdisplayskip, 149, 524
Berry naming scheme, 504
Berry, Karl, 381, 504, 605
\beta, 125, 524
bf (\TeX{} 2.09), 65, 367, 485
\bfdefault, 233, 372, 524
\bfseries, 19, 20, 64, 371, 416, 524
bgadd (background), 346
bgaddcenter (background), 346
bgclear (background), 346
.bib file, 309, 397
bibitem, 217, 219–21, 322, 395, 429–31, 524
bibliographic database, 219, 309–21
abbreviations, 320
INDEX

creating, 311
cross-referencing, 316, 317
entry type, 311, 312, 314
field, 312
  abbreviations, 319
  ignored, 312
  list of, 314, 316
  names, 317, 318
  optional, 312, 314
  required, 312, 314
  special formats, 317
titles, 318, 319
journal abbreviations, 322
structure, 311
template, 311, 320
bibliography, 216
  abbrv, 311
  abbrvnat, 221, 311
  alpha, 311
  author–year style, 220–4
  BIBTEX, 217, 219, 224, 309–21
customizing, 223, 321–2
date, 22, 23
  entry, 217
  format, see style
  multiple, 224
  numerical, 219
  open style, 39
  plain, 219, 310
  plainnat, 219, 221, 311
  reference in text, 219, 310
  style, 219, 221, 310
unsrt, 310
unsrtnat, 221, 311
\bibliography, 210, 219, 224, 309, 310, 397, 524
\bibliographystyle, 219, 221, 224, 310, 397, 524
\bibname, 460, 525
BibTEX, 15, 217, 219, 224, 309–21, 382, 388, 397, 430
  writing styles for, 321
Biemesderfer, C., 605
\Big (\TeX\), 148, 525
\big (\TeX\), 148, 525
\bigcap, 128, 525
\bigcirc, 126, 525
\bigcup, 128, 525
\Bigg (\TeX\), 148, 525
\bigodot, 128, 525
\bigoplus, 128, 525
\bigotimes, 128, 525
\bigskip, 33, 525
\bigskipamount, 525
\bigsqcup, 128, 525
\bigtriangledown, 126, 525
\bigtriangleup, 126, 525
\biguplus, 128, 525
\bmod, 129, 268, 526
body, 47
  bold face in formulas, 143, 394
\boldmath, 143, 144, 150, 151, 373, 394, 433, 526
\boldsymbol (\AMS), 144, 258, 526
  book class, 37, 41, 55, 57, 113, 130, 173, 183, 187, 191, 218, 224, 225, 277, 391, 459
  bookman package, 234
  Bookman, PostScript font, 234, 504
\boolean (\ifthen\), 194, 195, 360, 451, 453
\bot, 127, 526
\botfigrule, 173, 526
  bottom margin, 48
\bottombuttons (pdfscreen), 342
\bottomfraction, 172, 526
bottomnumber, 171, 526
Botway, L., 605
\bowtie, 126, 526
box, 85–94
calling, 87
framed, 86
LR, 85, 87
nested, 92
paragraph, 85
parbox, 88
positioning
horizontal, 86
vertical, 88, 89, 92
raising, 87
rule, 85, 91
saving, 87
style parameters, 93
\TeX primitive, 85
vertical, 88
\Box, 126, 127, 526
boxed formula, 142, 270
\boxed (\AMS), 142, 270, 526
bp (big point), 21, 155, 158
Braams, Johannes, 252, 464
brace (\TeX), 192
bracket symbol, see symbols
Breton, 252
\breve, 129, 263, 526
\Breve (\AMS), 263, 526
British spelling, 195
bsr-interpolated.map, 506
bsr.map, 506
.bst file, 220, 221, 310, 321, 397
btxdoc.tex, 321
btxhak.tex, 321
Bulgarian, 252
\bullet, 126, 526
\c (° accent), 24, 526
calc package, 394
calligraphic letters, 125, 284
\cap, 126, 527
\caption, 60, 108, 169, 173, 174, 177, 178, 182, 213, 427, 436, 527
\captionslang (non-standard), 255, 460, 527
Carlisle, David, 154, 159, 193, 215, 330, 480
cases environment (\AMS), 277, 517
Catalan, 252, 462
catalogue.html, 13
\cbinput (chapterbib), 224
cbunit environment (chapterbib), 224
\cc, 353, 527
cc (cicero), 21
\ccname, 353, 460, 527
CD environment (\AMS), 282
\cdot, 126, 527
\cdots, 123, 134, 137, 263, 527
center environment, 67, 69, 79, 98, 176, 289, 516
centered text, 67
centering and indenting, 67–9
\centering, 67, 69, 79, 178, 458, 527
\centerline (\TeX), 67, 176, 408, 527
\centerslidefalse (seminar), 334
\centerslidetrue (seminar), 334
centertags option, 272, 279
.cfg file, 253, 384, 385, 397, 472
cfoot (fancyhdr), 44
cfrac (\AMS), 265, 527
changes (doc), 470
chapter counter, 57, 181, 183
chapter opening page, 39
\chapter, 44, 55, 58, 449, 527
chapter*, 55, 527
chapterbib package, 224
\chaptername, 460, 528
character spacing, see spacing
character specification, 66
decimal, 66
hexadecimal, 66
octal, 66
\CharacterTable (doc), 471
\chead (fancyhdr), 44
\check, 129, 263, 526
\Check (\AMS), 263, 528
\CheckCommand, 430, 445, 528
\CheckCommand*, 445
\CheckSum (doc), 471
chemical formulas, 143
Chen, Pehong, 228
\chi, 125, 528
chicago package, 220
Chinese, 6
\choose (\TeX), 137, 146, 147, 189, 192
\circ, 126, 528
\circle, 416
\circle (eepic), 305
\circle (picture), 295, 299, 300, 431, 528
\circle* (eepic), 305
\circle* (picture), 295, 528
\cite, 219, 310, 312, 314, 395, 429, 430, 528
\citeauthor (natbib), 222
\citep (natbib), 219, 221, 310, 528
\Citet (natbib), 222
\citet (natbib), 219, 221, 310, 528
\citeyear (natbib), 222
ClassError, 446, 528
classes.dtx, 463
ClassInfo, 447, 529
ClassWarning, 446, 529
ClassWarningNoLine, 446, 529
Clausen, Joern, 25
cleardoublepage, 34, 171, 420, 529
clearpage, 34, 84, 171, 209, 420, 428, 529
forbidden, 84
\cline, 97, 102, 106, 529
.cls file, 384, 385, 397
\closing, 353, 529
.cls file, 384, 385, 398, 484
\ clubsuit, 127, 529
cm (centimeter), 21
\ cm-super fonts, 506
CodelineIndex (doc), 469
CodelineNumbered (doc), 470
Codepages, IBM, 26, 462
color package, 153, 166, 324, 325, 335, 338, 345, 439
\color (color), 167, 335, 529
color.cfg, 166
\colorbox (color), 167, 335, 529
colors, 166–8
column breaking, 34
column separation
two-column pages, 40
column separation symbol, 97, 139
\columnsep, 40, 51, 52, 395, 530
\columnseprule, 40, 52, 395, 530
\columnwidth, 51, 173
command
*-form, 18
distinguishing from text, 11
followed by blank, 19, 186
invisible, 201
levels, 438
multi-character, 17
name, 17
single character, 17
syntax, 18
two-character, 17
command characters, printing of, 23
command, user-defined, 185–92
examples\TeX as \LaTeX, 189
equation numbering, 190
footnotes, 190
framing text, 189
followed by blank, 186
for math and text, 186
general comments, 200
nested, 204
order of, 203
scope of, 202
global, 202
local, 202
storing, 200
unwanted spacing, 201
with arguments, 187, 188
calling, 188
with optional argument, 189
without arguments, 185, 187
calling, 186
comment, 118
character, 118
comment environment (verbatim), 111, 118, 396
commenting out, 118
compatibility mode, 392, 398, 418, 484
Comprehensive \TeX Archive
Network, see CTAN
compressed graphics files, 166
Computer Modern fonts, 8, 64, 283, 369, 488–97, 499
Process version, 235, 505
conditional text, 193
config.cms, 235
config.ps, 233, 235
configuration file, local
color, 166
graphics, 164
hyperref, 246
hyphen, 253, 385
\LaTeX installation, 384
\texdoc, 397, 447, 472
\pdfscreen, 343
\seminar, 339, 340
configuring \LaTeX, 384
\cong, 126
constant, see formula
\contentsline, 530
\contentsname, 459, 530
continued fraction, 146, 149, 265
continuing dots, 123, 263
\coprod, 128
copyright sign, 23, 503
\copyright, 23, 503, 530
\copyrightspace, 391
\cornersize (fancybox), 94
\cos, 128
\cosh, 128
\cot, 128
\coth, 128
counter value, 183
multiple printing, 183
printing, 183
counter, \LaTeX, see \LaTeX counter
counter, user-defined, 182
auto. reset, 182
changing, 182
creation, 182
incrementing, 182
setting, 182
Courier, PostScript font, 234, 379, 504
Croatian, 252
cross-reference, 213–16
external document, 215
to bibliography, 219
to counters, 182, 213
to equations, 131, 139, 213, 278
to figures, 177, 213
to lists, 213
to pages, 213
to sections, 56, 213
to tables, 177, 213
to text, 213
to theorems, 213
variable, 215
\csc, 128
\cup
\CurrentOption, 359, 443, 530
cyracc.def, 284
Cyrillic fonts, 6, 283, 379, 497
transliteration, 284
Czech, 252
\d (accent), 24
\dag, 23
\dagger, 126
Dahlgren, Mats, 178
Daly, Patrick W., 220, 321
Danish, 252
DANTE, German-speaking \TeX
Users, 251
dash, 23
as hyphen, 23
as minus sign, 23
\dashbox (picture), 291, 292, 300, 531
dashjoin environment (epic), 302, 304
\dashline (epic), 302, 303
\dashlinestretch (epic), 305
\dashv, 126, 127
database, see bibliographic database
\date, 52
\datelang (non-standard), 255, 461, 531
\day (\TeX), 27, 460
\dbinom (\AMS), 265, 531
\dblfigrule, 173
\dblfloatpagefraction, 172, 531
\dblfloatsep, 172, 532
\texttt{\textbackslash dbltextfloatsep}, 172, 532
\texttt{\textbackslash dbltopfraction}, 172, 532
dbltopnumber, 172, 532
DC fonts, see EC fonts
dcolumn package, 107, 394
dd (ddot), 21
\texttt{\textbackslash ddag}, 23, 532
\texttt{\textbackslash dagger}, 126, 532
\texttt{\textbackslash ddddot} ($\mathcal{A_\text{MS}}$), 263, 532
\texttt{\textbackslash ddot} ($\mathcal{A_\text{MS}}$), 263, 532
\texttt{\textbackslash ddot}, 129, 263, 532
\texttt{\textbackslash Ddot} ($\mathcal{A_\text{MS}}$), 263, 532
\texttt{\textbackslash ddot}, 123, 532
de Boer, Ingo H., 15
DEC Multinational coding, 462
declaration, 20
local, 20
scope of, 20
\texttt{\textbackslash DeclareBoldMathCommand} (bm), 394
\texttt{\textbackslash DeclareErrorFont}, 376, 423, 424, 532
\texttt{\textbackslash DeclareFixedFont}, 66, 361, 373, 533
\texttt{\textbackslash DeclareFontEncoding}, 376, 378, 379, 533
\texttt{\textbackslash DeclareFontEncodingDefaults}, 376, 533
\texttt{\textbackslash DeclareFontFamily}, 376, 378, 423, 533
\texttt{\textbackslash DeclareFontShape}, 377, 378, 423, 533
\texttt{\textbackslash DeclareFontSubstitution}, 376, 533
\texttt{\textbackslash DeclareGraphicsExtensions} (graphics), 165, 533
\texttt{\textbackslash DeclareGraphicsRule} (graphics), 165, 533
\texttt{\textbackslash DeclareInputMath} (inputenc), 461
\texttt{\textbackslash DeclareInputText} (inputenc), 461
\texttt{\textbackslash DeclareMathAccent}, 375, 423, 534
\texttt{\textbackslash DeclareMathAlphabet}, 191, 374, 375, 416, 423, 534
\texttt{\textbackslash DeclareMathDelimiter}, 375, 534
\texttt{\textbackslash DeclareMathOperator} ($\mathcal{A_\text{MS}}$), 268, 282, 534
\texttt{\textbackslash DeclareMathRadical}, 375, 534
\texttt{\textbackslash DeclareMathSizes}, 375, 534
\texttt{\textbackslash DeclareMathSymbol}, 375, 534
\texttt{\textbackslash DeclareMathVersion}, 374, 422, 423, 535
\texttt{\textbackslash DeclareOldFontCommand}, 373, 535
\texttt{\textbackslash DeclareOption}, 359, 421, 442, 535
\texttt{\textbackslash DeclareOption*}, 359, 443, 535
\texttt{\textbackslash declarepostamble} (DocStrip), 465
\texttt{\textbackslash declarepreamble} (DocStrip), 465
\texttt{\textbackslash DeclareRobustCommand}, 445, 535
\texttt{\textbackslash DeclareRobustCommand*}, 445, 535
\texttt{\textbackslash DeclareSymbolFont}, 374, 423, 424, 535
\texttt{\textbackslash DeclareSymbolFontAlphabet}, 375, 424, 536
\texttt{\textbackslash DeclareTextAccent}, 379, 536
\texttt{\textbackslash DeclareTextAccentDefault}, 380, 536
\texttt{\textbackslash DeclareTextCommand}, 379, 417, 536
\texttt{\textbackslash DeclareTextCommandDefault}, 380, 536
\texttt{\textbackslash DeclareTextComposite}, 379, 422, 536
\texttt{\textbackslash DeclareTextCompositeCommand}, 379, 536
\texttt{\textbackslash DeclareTextFontCommand}, 373, 536
\texttt{\textbackslash DeclareTextSymbol}, 379, 536
\texttt{\textbackslash DeclareTextSymbolDefault}, 380, 537
\texttt{\textbackslash def} (\TeX), 439, 445, 450
\texttt{\textbackslash def} file, 153, 253, 376, 378, 379, 385, 392, 398
\texttt{\textbackslash definecolor} (color), 167, 345, 537
\texttt{\textbackslash deg}, 128, 537
dehyphn.tex, 256
dehyphntex, 256
delarray package, 108, 394
\texttt{\textbackslash DeleteShortVerb} (shortvrb), 111, 393, 468, 537
\texttt{\textbackslash Delta}, 125, 537
\texttt{\textbackslash delta}, 125, 537
\texttt{\textbackslash depth}, 86, 90, 537
\texttt{\textbackslash DescribeEnvironment} (doc), 468
\texttt{\textbackslash DescribeMacro} (doc), 468
description environment, 69, 70, 74, 79, 230, 419, 420, 517
design size, 501
\det, 128, 129, 280, 537
determinant, 133–6
\dfrac (\mathcal{M}), 264, 537
\DH, 503, 537
dh, 503, 538
diacritical marks, see accents
\Diamond, 126, 127, 538
\diamond, 126, 538
\diamondsuit, 127, 538
dim, 128, 538
\DisableCrossrefs (doc), 470
discretionary, 35, 538
displaybreak (\mathcal{M}), 278, 538
displayed formula, 119
displayed text
-centered, 67
-indented, 67
-left or right justified, 67
-nested, 68
displaymath environment, 120, 517
displaystyle, 142, 146, 150, 264, 538
div, 126, 538
\DJ, 503, 538
\dj, 503, 538
doc package, 392, 399, 441, 467–73
doc.dtx, 468
DocBook DTD, 479
\DocInput, 467
DocStrip, 321, 398, 464–7
docstrip.dtx, 467
docstrip.tex, 385, 391, 465
document class, 37
-options, 120
document environment, 12, 517
Document Type Definition, 479
document, major subdivisions, 52–7
documentation
-browser, 383
-integrated, 463, 467
-with doc, 467–73
documentation of packages, 13
documentclass, 12, 37, 41, 398, 416, 422, 443, 538
\documentstyle (\TeX 2.09), 418, 422, 483
dollar sign, 23, 27
\DoNotIndex (doc), 470
dot, 129, 263, 539
\Dot (\mathcal{M}), 263, 539
doteq, 126, 539
dotfill, 30, 84, 86, 135, 539
dots, 124, 539
dots (\mathcal{M}), 263, 539
dotsb (\mathcal{M}), 264, 539
dots (\mathcal{M}), 264, 539
dotsi (\mathcal{M}), 264, 539
dotimes (\mathcal{M}), 264, 539
dottedjoin environment (epic), 302, 304
dottedline (epic), 302, 303
double dagger, 23
\doublebox (fancybox), 94, 539
doublerulesep, 98, 539
\Downarrow, 127, 132, 539
\downarrow, 127, 132, 539
Downes, Michael, 257
draft option, 39
Drakos, Nikos, 476
drawjoin environment (epic), 302, 304
\drawline (epic), 161, 302, 303
\drawlinestretch (epic), 305
driver, printer, 15, 153, 389, 391, 398, 453, 488, 497
.drv file, 463, 464
DTD, 479
DocBook, 479
TEL, 479, 480
.dtx file, 13, 384, 389, 398, 463, 465
Duchier, Denys, 464
Dutch, 252
.dvi file, 14, 27, 153, 161, 236, 305, 393, 403, 477
dvipdf, graphics option, 154
dvipdfm, PDF driver, 162, 237, 305, 346, 387, 389
dvipdfm, graphics option, 154, 237
dvips, graphics option, 154
dvips, printer driver, 15, 162, 167, 168, 231, 236, 305, 332, 333, 335, 337, 387, 389, 476
dvipson, graphics option, 154
dviwin, graphics option, 154
INDEX

e.tex, 394, 413
EC fonts, 393, 499–503
calling, 501
character assignments, 500
special character commands, 503
with NFSS, 379
\edef (TEX), 450
eepic package, 305–6
eepicemu package, 305
Eijkhout, Victor, 450, 605
electron volt, symbol, 30, 192
electronic documents, 340, 475
electronic projection, 323
electronic publishing, 236
\el, 127, 539
\ellipse (eepic), 306
eppsis, 123
\else (TEX), 451
em dash, 23
\em, 20, 62, 65, 433, 485, 539
em, 21
emacs, 382
email addresses, 112
\emclena (pdfscreen), 341
\emph, 62, 65, 372, 540
empty page style, 42, 327, 335, 352, 452
\emptyset, 127, 540
emTeX for DOS, 382
emtex, graphics option, 154
en dash, 23
\EnableCrossrefs (doc), 470
encapsulated PostScript, see PostScript
\encl, 353, 540
\enclname, 353, 460, 540
encoding, see font
encoding commands in NFSS, 379
encoding scheme
OML, 491, 495
OMS, 491, 495
OMX, 491, 496
OT1, 490, 492, 506
OT2, 283, 497
T1, 500, 501, 503, 506
TS1, 502, 506
encoding, font attribute, 368
\encodingdefault, 372, 501
end of line, as a blank, 22
\end, 19, 64, 416, 540
\endbatchfile (DocStrip), 465
\end{document}, 12, 517
\endfirsthead (longtable), 108, 540
\endfoot (longtable), 108, 540
\endhead (longtable), 108, 540
\endinput (TEX), 451
\endlastfoot (longtable), 108, 540
\endpostamble (DocStrip), 465
\endpreamble (DocStrip), 465
English, 252
english.1df, 255
\enlargethispage, 34, 419, 540
\enlargethispage*, 34, 540
\ensuremath, 186, 187, 418, 450, 541
enumerate environment, 69, 70, 71, 72, 74, 79, 213, 394, 419, 420, 517
enumerate package, 74, 394
\enumcounter, 73, 181
environment, 19, 516
command name as, 20, 517
global, 20
math, 119
nameless, 20, 62, 63
environment environment (doc), 469
environment, user-defined, 195–200
general comments, 200
scope of, 202
global, 202
local, 202
storing, 200
with arguments, 198, 199
calling, 198
with optional argument, 199
without arguments, 196, 197
calling, 196
epic package, 302–5
 eps file, 155
espsf package, 159
\epsf (epsfig), 159
epsfig package, 159
\epsfig (epsfig), 159
\epsfxsize (epsfig), 159
\texttt{\textbackslash epsfysize (epsfig)}, 159
\texttt{\textbackslash epsilon}, 125, 541
\texttt{eqnarray} environment, 120, 138, 142, 213, 270, 420, 517
\texttt{eqnarray*} environment, 120, 138, 142, 270, 518
\texttt{\textbackslash eqref (A_M S)}, 278, 541
\texttt{\textbackslash equal (ifthen)}, 194, 363
equation, see formula
equation environment, 181, 277
equation environment, 120, 124, 213, 271, 276, 518
equation number, 120, 271
changing hierarchy, 277
right or left, 39, 120, 131, 271
subnumbered, 277
user-defined, 191
vertically centered, 142, 272
\texttt{equation*} environment (A_M S), 272
equations, set of, 133–6
\texttt{\textbackslash equiv}, 126, 541
error messages
\texttt{\textbackslash errorcontextlines} counter, 404
basic structure for \TeX, 403
continue program, 402
emergency stop, 410
error indicator, 402
error line, 402, 405
from \LaTeX, 404–7
from \TeX, 401–4
from \TeX{} macros, 407
going to deeper levels, 404
list
hard-to-find errors, 435
\LaTeX font errors, 422–4
\LaTeX font warnings, 433
\LaTeX{} general errors, 415–20
\LaTeX{} general warnings, 429–31
\LaTeX{} package errors, 421–2
\LaTeX{} package warnings, 432
\TeX, 424–9
\TeX{} warnings, 433–5
mathematical, 413, 414
multi-file, 414
propagation, 409–10
stop program
with \texttt{\textbackslash I\textbackslash stop}, 403, 410
with editor, 403
with X, 403, 410
unknown file name, 412, 417
user response
call editor, 403
continue program, 402
correction, 403
deletion, 403
help, 403
recommendation, 407
\texttt{\textbackslash errorcontextlines} counter, 404
\texttt{\textbackslash eso-pic} package, 346
Esperanto, 252, 462
Esser, Thomas, 381
Estonian, 252
\texttt{\textbackslash eta}, 125, 541
eucal package, 284–5
eufrak package, 285
\texttt{\textbackslash EUR (europs)}, 25, 541
euro symbol, 24–5, 387, 462, 503
PostScript fonts, 25, 387
\texttt{\textbackslash euro (eurosym)}, 25, 28, 541
\texttt{\textbackslash EUROfc (europs)}, 25, 541
eurofont.exe, 387
European Commission, 25
European paper sizes, 12
europs package, 25, 387
eurosans package, 25, 387
eurosym package, 25, 28
\texttt{\textbackslash evensidemargin}, 48, 49, 361, 453, 541
\texttt{\textbackslash ex}, 21
\texttt{\textbackslash ExecuteOptions}, 165, 443, 453, 541
\texttt{\textbackslash executivepaper} option, 38
exercises
Chapter 2, 27, 28
Chapter 3, 40, 45, 47–9, 54, 57, 58, 60
Chapter 4, 68, 69, 72, 74, 78, 84, 93, 103, 105, 110, 111, 116, 118
Chapter 5, 121, 124, 129, 130, 132, 135–7, 141, 149, 150
Chapter 6, 157, 159, 168
Chapter 8, 183, 187, 192, 200
Chapter 9, 208, 212, 218, 219
Chapter 13, 290, 293, 295, 297, 301
\texttt{\textbackslash exists}, 127, 541
\exp, \textit{128, 541}
\expandafter, \textit{128, 541}
explicit names, changing, \textit{459}
exponents, 121
\texttt{exscale} package, 392
Extended Computer fonts, \textit{see EC fonts}
\texttt{externaldocument (xr)}, 215, 243
\texttt{extracolsep}, \textit{96, 110, 455, 541}
\texttt{extrarowheight (array)}, 107
\texttt{extraslang (non-standard)}, 255
\texttt{extralineheight (seminar)}, 336
\texttt{\f@baselineskip}, 376
\texttt{\f@encoding}, 376
\texttt{\f@family}, 376
\texttt{\f@series}, 376
\texttt{\f@shape}, 376
\texttt{\f@size}, 376
Fairbairns, Robin, 47
falign environment (\texttt{AM}), 270, 275, 518
family, font attribute, 64, 368
\texttt{\familydefault}, 372
fancy page style, 43
fancybox package, 94–5, 334, 338, 345
\texttt{\fancyfoot (fancyhdr)}, 44
fancyhdr package, 43–5, 249, 335, 454
\texttt{\fancyhead (fancyhdr)}, 44
\texttt{\fancyhf (fancyhdr)}, 44
\texttt{\fancypage (fancybox)}, 95, 542
\texttt{\fancypagenestyle (fancyhdr)}, 45
\texttt{\fancyput (fancybox)}, 338
\texttt{\fbox, \textit{86, 92–4, 115, 142, 334, 542}}
\texttt{\fbox (picture)}, 298
\texttt{\fboxrule, \textit{93, 94, 164, 334, 542}}
\texttt{\fboxsep, \textit{93, 94, 164, 298, 334, 542}}
\texttt{\fcolorbox (color), \textit{167, 335, 542}}
\texttt{.fdd file}, 398
\texttt{.fd file}, 233, 234, 283, 376, 378, 385, 398, 424, 490
\texttt{\fi (\texttt{\LaTeX})}, 451
\texttt{figure, see float}
\texttt{figure counter}, 181
\texttt{figure caption, see float, caption}
\texttt{figure environment, \textit{60, 108, 169, 173, 182, 213, 417, 419, 430, 518}}
\texttt{\figurename}, \textit{459, 542}
\texttt{figures, list of}, 59
\texttt{file}
including, 448
inputting safely, 447
listing, 447
transcript, 14, 50, 208, 211, 399, 401, 447
file transfer protocol, \textit{see FTP}
file types
\texttt{.aux}, 209, 210, 214, 396, 397, 418, 427, 430, 436
\texttt{.bbl}, 219, 224, 310, 397
\texttt{.bib}, 309, 397
\texttt{.blg}, 397
\texttt{.bst}, 220, 221, 310, 321, 397
\texttt{.cfg}, 253, 384, 385, 397, 472
\texttt{.clo}, 384, 385, 397
\texttt{.cls}, 384, 385, 398, 484
\texttt{.dat}, 253
\texttt{.def}, 153, 253, 376, 378, 379, 385, 392, 398
\texttt{.drv}, 463, 464
\texttt{.dtx}, 13, 384, 389, 398, 463, 465
\texttt{.dvi}, 14, 27, 153, 161, 236, 305, 393, 403, 477
\texttt{.eps}, 155
\texttt{.fdd}, 398
\texttt{.fd}, 233, 234, 283, 376, 378, 385, 398, 424, 490
\texttt{.fmt}, 398
\texttt{.gif}, 476
\texttt{.glo}, 230, 398, 470
\texttt{.glsl}, 399, 470
\texttt{.gz}, 166
\texttt{.html}, 476
\texttt{.idv}, 478
\texttt{.idx}, 226–8, 399, 470
\texttt{.ilg}, 399
\texttt{.ind}, 228, 399
\texttt{.ins}, 389, 399, 463
\texttt{.ist}, 385, 399, 470
\texttt{.jpeg}, 162
\texttt{.jpg}, 162
\texttt{.idf}, 253
\texttt{.lof}, 59, 179, 399, 436
INDEX 621

.log, 14, 246, 399, 401, 403, 414, 429
.lot, 59, 399, 436
.map, 234
.mbs, 321, 322
 mf, 389, 400, 488, 498, 506
.pbm, 476
.pdf, 162, 237
.pfa, 389
.pfb, 234, 389, 506
.pk, 233, 234, 236, 388, 400, 488, 498, 504
.png, 162
.ppm, 478
.ps, 232, 236
.sty, 9, 12, 41, 384, 385, 463, 484
.tcx, 481
.tex, 14, 17, 208, 209, 391, 396, 400, 403, 436
.tfm, 234, 400, 487, 498, 503, 506
.tif, 162
tif, 162
toc, 59, 400, 436
txt, 384
.vf, 234, 236, 400, 488, 504
.zip, 166

\file (DocStrip), 465
filecontents environment, 432, 488, 518
filecontents* environment, 449, 518
\filedate (doc), 471
fileerr.dtx, 394
\fileinfo (doc), 471
filename (doc), 471
fileversion (doc), 471
\fill, 22, 30, 32, 96, 110, 184, 455, 542
filler, with dots, rules, spacing, 30
final option, 39
\Finale (doc), 469, 473
Finnish, 252
\firstline (array), 107
firstpage page style, 362
fixed length, 21
flafter package, 171, 392
\flat, 127, 542
fleqn option, 39, 120, 124, 149, 271, 280
float, 110, 169–77
caption, 173
width, 174
causing buffer overflow, 428
double columns, 169
examples, 109, 174–7
figure title, 173
head and foot text, 109
numbering, 173
placement, 109, 169–71
style parameter, 171
table title, 173
float package, 179–80
floatfig package, see floatflt
floatflt package, 178–9
floatingfigure environment, 178
floatingtable environment, 179
\floatname (float), 179
\floatpagefraction, 172, 542
\floatplacement (float), 180
\floatsep, 172, 542
\floatstyle (float), 180
\flushbottom, 48, 542
\flushleft environment, 67, 79, 518
\flushright environment, 67, 79, 518
\flushright, 117
Flynn, Peter, 605
.fmt file, 398
\fnsymbol, 114, 183, 190, 543
font
slides class, 324
amsfonts, 283–5
attribute, 64–5, 368–72
defaults, 372
encoding, 368
family, 64, 368
internals, 376
series, 64, 368
shape, 64, 368
size, 64, 369
bitmaps, 488
character assignment
cmsc10 (small caps), 492
cmex10 (var. symbols), 496
cmmi10 (math text), 495
cmr10 (standard), 492
cmsy10 (math symbols), 495
cmti10 (text italic), 493
cmtt10 (typewriter), 493
EC fonts, 500
TC fonts, 502
wncyr10 (Cyrillic), 497
classification, 489
commands, 65, 370
Computer Modern, 8, 64, 283, 369, 488–97, 499
Cyrillic, 283, 379, 497
decorative, 494
design size, 501
EC, 393, 499–503
calling, 501
with NFSS, 379
encoding, 487
euro, PostScript, 25, 387
eurosym, 25
extended, 498–503
Extended Computer, see EC families, 489
file names, 490
\LaTeX, 65, 494
\LaTeX 2.09 commands, 485
line spacing for additional, 562
loading, 66
logos, 494
magnification, 66
math, 491
symbols, 374
math alphabets, 133, 258, 373
NFSS, 367–72, 376–9
pixel, 8
PostScript, 233–6, 378, 503–5
standard, 65, 367
standard line spacing, 63
TC, 24, 393, 501
text, 490
TrueType, 238
type 1, 8, 25, 231, 238, 387, 498, 503
type 3, 8, 234, 238
virtual, 488, 503
font definition files, 236, 376, 398
font environment, 64
font size, 62
changing, 62, 367
class option, 37
declarations, 63
in formulas, 146
standard line spacing, 63
font size commands (\LaTeX 2.09), 485
font style commands (\LaTeX 2.09), 485
fontenc package, 378–9, 392, 501
\fontencoding, 368, 370, 543
\fontfamily, 368, 369, 370, 543
\fontmath.cfg, 385
\fontmath.ltx, 385
\fontseries, 368, 369, 370, 543
\fontshape, 368, 369, 370, 372, 543
\fontsize, 368, 369, 370, 543
\fottmpl package, 394
\fonttext.cfg, 385
\fonttext.ltx, 385
foot, 42, 44, 47, 50, 85, 249, 335, 347, 362, 435, 453
customizing, 43, 454
\footheight (\LaTeX 2.09), 604
footnote, 112–16
change markers, 114
forbidden mode, 114
in minipages, 113, 115
in tables, 116
marker without text, 114
non-standard, 113
standard, 113
forbidden, 113
standard marker, 113
style parameter, 117
text external to forbidden mode, 114
\footnote counter, 113, 181, 190
\footnote, 112, 113, 114, 116, 543
\footnotemark, 114, 115, 543
\footnoterule, 118, 543
\footnotesep, 117, 544
\footnotesize, 63, 371, 428, 544
\footnotetext, 110, 114, 115, 544
\footruulewidth (fancyhdr), 44
\footskip, 48, 50, 453, 544
\foott1, 127, 544
\foreignlanguage (babel), 254, 544
format file, 6, 384, 398, 440
formula, 119
arrays, 133–6, 266
blanks within, 120
bold face in, 143, 258, 394
chemical, 143
constants, 120
continued fraction, 146, 149, 265
continuing dots, 123, 263
displayed, 119
centered, 39, 120, 280
flush left, 39, 120, 280
indented, 40
ellipsis, 123
exponent, 121
extra T\TeX commands, 137
fine-tuning, 145–8
  bracket sizing, 148
  font size, 146–7
  horizontal spacing, 145, 269
font size in, 146
  detailed, 147
  simplified, 146
fraction, 122
framed, 142
function names, 128, 267
Greek letters, 125
horizontal spacing, 145, 269
in text, 119
index, 121
integral, 123
limits
  for 2 sizes, 128
  multiline, 261
  positioning, 123, 128
  with functions, 129
lowering in, 121
math style parameters, 148
matrix, 133–6, 266
multiline, 120, 138–41
  bracket sizing, 140, 148
  breaking, 138
  centered, 138
  column separation, 138
  left part, 140
  numbered, 139
  page breaks, 278
  with \AMS, 270–7
numbered, 120
overbrace, 136
overlining, 136
raising in, 121
root, 122, 269
side-by-side, 142
subnumbered, 277
summation, 123
symbols in, see symbols
text within, 133, 259
underbrace, 136
underlining, 136
variables, 120
fp\TeX for Windows, 381, 385
\frac, 122, 146, 189, 264, 544
fractions, 122
  with \AMS, 264
fragile commands, 440, 444
\frame, 299, 300, 544
\framebox, 86, 90, 93, 94, 106, 189, 272, 300, 544
\framebox (picture), 291, 292, 298, 544
framed formulas, see boxed formula
framed table, 100
framed text, 86
French, 252
french package, 251
\frenchspacing, 29, 545
frhyphen.tex, 256
\from (DocStrip), 465
\fromname, 363
\frontmatter, 57, 210, 545
\frown, 126, 545
ftnright package, 394
FTP, 475
fullpage package, 452, 465
function names, 128
  defining, 268
  with \AMS, 267
\fussy, 36, 545

Galician, 252, 462
\Gamma, 125, 545
\gamma, 125, 545
gather environment (\AMS), 270, 273, 519
gathered environment (\AMS), 276, 519
Gaulle, Bernard, 251
\gcd, 128, 129, 280, 545
\gdef (\TeX{}), 450
\ge, 126, 545
\generate (DocStrip), 465
\genfrac (\AMS{}), 265, 545
\generate (\DocStrip{}), 465
\generate (\AMS{}), 265, 545
\generate (\DocStrip{}), 465
\generate (\AMS{}), 265, 545
geometry package, 49–51, 452
\geq, 126, 545
Gerry, 252
spelling reform, 35
\german{} package, 251, 252, 360
\GetFileInfo (doc), 441, 471, 472
\gets, 127, 545
\gg, 126, 545
\gglo.ist, 399, 470
Ghostscript, 236, 476
GhostView, 164, 232, 239
GIF images, 476, 478
\gif{} file, 476
\gind.ist, 399, 470
\GetFileInfo (doc), 441, 471, 472
\gets, 127, 545
\gglo.ist, 399, 470
Ghostscript, 236, 476
GhostView, 164, 232, 239
GIF images, 476, 478
\gif{} file, 476
\gind.ist, 399, 470
Girou, Denis, 330
\glo{} file, 230, 398, 470
glossary, 230
\glossary, 230, 398, 545
\glossaryentry, 230, 398, 545
\glossaryname, 460
\gls{} file, 399, 470
Goossens, M., 605
graphics
  importing files, 154
rotation, 156
scaling, 156
\graphics{} package, 153, 155–7
\graphics.cfg, 164
\graphicspath (graphics), 165
\graphicx{} package, 157–9
\graphpap{} package, 301, 392
\graphpaper, 392
\graphpaper (\graphpap{}), 301, 546
\grave, 129, 263, 546
\Grave (\AMS{}), 263, 546
Greek, 252
Greek letters, 125
grfguide.tex, 154
\grid (epic), 302, 303
\guillemotleft, 503, 546
\guillemotright, 503, 546
\guilsinglleft, 503, 546
\guilsinglright, 503, 546
Guntermann, Klaus, 343
Gurari, Eitan M., 477
\gz{} file, 166
gzip program, 166
\h{} file, 394, 413
\H{} (˘ accent), 24, 546
Haralambous, Y., 605
Hargreaves, K. A., 605
harvard package, 220
\hat, 129, 546
\Hat (\AMS{}), 263, 546
\hbar, 127, 546
\bbox (\TeX{}), 439
\hdotsfor (\AMS{}), 267
head, 42, 44, 47, 50, 85, 335, 362, 435, 453
customizing, 43, 454
right page, 43
\headheight, 48, 50, 361, 363, 453, 455, 546
heading, 43
left page, 43
setting values, 47
headings page style, 42, 44, 55, 58, 327, 335, 352, 363, 452
\headrulewidth (fancyhdr), 44
\headsep, 48, 50, 361, 363, 453, 455, 546
\headtoname, 359, 360, 363, 460, 547
\heartsuit, 127, 547
Hebrew, 252
\height, 86, 90, 547
helvet package, 234
Helvetica, PostScript font, 234, 324, 379, 504
here package, 178
\hfill, 30, 32, 77, 84, 88, 89, 117, 142, 149, 176, 362, 547
\hhline package, 395
\hline, 97, 100, 102, 103, 106, 547
\hoffset (\TeX{}), 547, 604
\hom, 128, 547
\hookleftarrow, 127, 547
\hookrightarrow, 127, 547
\href (background), 345
\href (hyperref), 247
\hrulefill, 30, 84, 91, 547
\hspace, 29, 30, 82, 96, 105, 106, 110, 332, 547
\hspace*, 29, 82, 547
\hss (\TeX), 408
HTML, 3–5, 9, 476–8
html package, 476
.html file, 476
\Huge, 63, 371, 548
\huge, 63, 371, 548
Hungarian, 252
\hyperbaseurl (hyperref), 247
\hyperdef (hyperref), 247
\hyperimage (hyperref), 247
\hyperlink (hyperref), 161, 247, 349, 548
\hyperref (hyperref), 247
hyperref.cfg, 246
\hypersetup (hyperref), 241, 246, 349, 548
\hypertarget (hyperref), 247, 349, 548
hyphen, 23
hyphen.cfg, 252, 256, 385
babel, 253
hyphen.ltx, 385
hyphen.tex, 385
hyphenation, 34, 427
exception list, 35
in other languages, 499
manual, 35
multilingual, 36, 251, 256
other languages, 389
overfull \hbox, 434
turning off, 36
with accents, 426
\hyphenation, 36, 426, 427, 548
\i, 24, 548
\indent, 46, 550
\indentfirst package, 46, 395
index of keywords, 225–8
\index, 225, 226, 228, 391, 392, 399, 420, 428, 550
\indexentry, 226, 399, 550
\indexname, 225, 459, 550
\indexspace, 225, 550
\ind file, 228, 399
indices, 121
\inf, 128, 129, 280, 550
\infty, 127, 550
\iffalse (\TeX), 468
\iffileexists, 447, 548
\iflanguage (babel), 254, 548
\ift then package, 193–5, 359, 392, 439, 452
\iftthenelse (ifthen), 193, 195, 360, 363, 439, 451, 453
\iiint (\AMS), 260, 551
\iiint (\AMS), 260, 551
\iiint (\AMS), 260, 551
\ilg file, 399
\imath, 127, 549
\imagebutton (pdfscreen), 342
\imath, 127, 129, 549
importing graphics files, 154–66
compressed, 166
configuring, 164
problems, 161
\in, 126, 549
\inch, 21
\include, 182, 209, 210, 212, 215, 224, 396, 397, 402, 411, 412, 414, 415, 417, 549
\includegraphics, 156, 159–61, 163, 164, 169, 249, 332, 346
\includegraphics (graphics), 155, 549
\includegraphics (graphicx), 157, 550
\includegraphics* (graphics), 156, 549
\includegraphics* (graphicx), 159
\includeonly, 209, 210, 211, 215, 328, 416, 550
\includegraphics, 156, 549
\includegraphics, 156, 549
\includegraphics, 156, 550
\includespace, 225, 459, 550
\ind file, 228, 399
indices, 121
\inf, 128, 129, 280, 550
\infty, 127, 550
initex, 6, 251, 253, 256, 384, 398
input coding, 26, 461
\input, 192, 201, 207, 208, 209, 396, 402, 411, 412, 414, 415, 442, 447, 551
nested, 208
inputenc package, 26, 392, 461–2
\InputIfFileExists, 447, 551
.ins file, 389, 399, 463
install.txt, 384
instr-l.tex, 257
\int, 123, 128, 551
integral sign, 123
Internet, 9, 316, 390, 475, 590
giving addresses, 112
\intertext (AMS), 259, 550
\intextsep, 172, 551
intlimits option, 260, 279
Irish Gaelic, 252
invisible commands, 201
\invisible (slides), 326, 551
\iota, 125, 551
Irish Gaelic, 252
ISO math standards, 144
ISO-Latin codings, 462
\isodd (ifthen), 193
.ist file, 385, 399, 470
\it (L\TeX 2.09), 367, 373, 485
Italian, 252
italic correction, 62, 65, 487
\itdefault, 372, 551
\item, 69, 70, 73–5, 77, 78, 200, 214, 225, 409, 416, 418, 419, 551
\item[option], 71, 72, 79, 551
\itemindent, 77, 79, 551, 604
itemize environment, 69, 70, 71, 72, 74, 79, 419, 420, 519
\itemsep, 76, 77, 552, 604
\itshape, 64, 371, 372, 416, 458, 552
\j, 24, 552
Japanese, 6
\jmath, 127, 129, 552
\Join, 127, 552
\jot, 149, 552
JPEG images, 162, 237, 238
.jpg file, 162
\jput (epic), 302, 304
justification, 28, 34
\k (accent), 503, 552
\kappa, 125, 552
\keepsilent (DocStrip), 465
\ker, 128, 552
keyboard input, 211
keyboard symbols, 120
\kill, 82, 552
Knappen, Jörg, 499
Kneser, Thomas, 178
Knuth, Donald E., 6, 7, 119, 151, 218, 381, 488, 494, 497, 605
Kopka, Helmut, 606
Kwok, Conrad, 305
\L, 24, 552
\l, 24, 552
label
for bibliography, see bibliography entry
for lists, see list labels
\label, 56, 131, 139, 177, 182, 183, 213, 247, 278, 395, 396, 430, 431, 552
\labelenum, 73, 552
\labelitem, 73, 553
\labelsep, 77, 200, 553, 604
\labelwidth, 77, 79, 200, 553, 604
lablst.tex, 214, 385, 391
\Lambda, 125, 553
\lambda, 125, 553
Lamport, Leslie, 7, 8, 75, 193, 207, 218, 228, 229, 321, 437, 467, 606
landscape environment (lscape), 159
landscape option, 38, 232
\landscapeonly (seminar), 332
\langle, 132, 270, 553
language.dat, 252, 256, 385
language.dat (babel), 253
\language (\LaTeX), 36, 251, 255, 256, 553
\languagesname (babel), 254
languages
accents and special letters, 24
babel, 252
Lappish, 462
\LARGE, 63, 325, 371, 553
\Large, 63, 371, 458, 553
\large, 63, 367, 371, 553
\lasthline (array), 107
\LaTeX compatibility with 2.09, 392, 398, 418, 484
files, 12, 396–400
fonts, 61–6
installing, 381–6
logo, 18
sources, 389
version 3, 8
\LaTeX counters
enum, 73, 181
chapter, 57, 181, 183
equation, 181, 277
errorcontextlines, 404
figure, 181
footnote, 113, 181, 190
MaxMatrixCols (AMS), 266
mpfootnote, 115, 181
page, 45, 181, 183, 190, 195
paragraph, 57, 181
part, 57, 181
secnumdepth, 56, 58, 458
section, 57, 181, 183, 277
subparagraph, 57, 181
subsection, 57, 181
subsubsection, 57, 181
table, 181
tocdepth, 58
\LaTeX format, 14, 384, 398
\LaTeX, 18, 440, 553
\latex .fmt, 385, 398
\latex .ltx, 384, 390, 399, 463
\latex .tex, 467
\LaTeX 2.09, 7, 9, 46, 65, 153, 159, 324, 330, 373, 374, 392, 422, 430, 440, 443, 445, 483–6
\latex209 .def, 398
\LaTeX2HTML translator, 476, 480
\LaTeX3 Project, 3, 8
\latexbug .tex, 385
\LaTeXe, 19, 418, 553
\LaTeX error messages, see error messages
latexsym package, 126, 285, 392, 422, 494
Latin, 252
layout package, 47, 395
\lceil, 132, 553
ldots, 123, 137, 263, 553
le, 126, 553
\leadsto, 127, 553
leftmargin, 48, 50
two-sided, 48
\left, 108, 131, 134, 137, 140, 553
\Leftarrow, 127, 554
\leftarrow, 127, 554
\leftarrow, 127, 554
\leftharpoondown, 127, 554
\leftharpoonup, 127, 554
\leftmargin, 76, 77, 79, 80, 554, 604
\leq, 126, 554
\leftmargini, 80, 554
\leftmark, 44
\Leftarrow, 127, 554
\Leftarrow, 127, 554
\leftroot (AMS), 269, 554
legalpaper option, 38
length, 21, 184
0 value, 21
fixed, 21
rubber, see rubber lengths
setting, see \setlength
units of, 21
zero, 21
length, user-defined, 184
\lengthtest (ifthen), 194
\leq, 126, 554
\leqno option, 39, 120, 131, 271, 272, 280
\let (\LaTeX), 451
letter, 351–64
closing, 353
commands
forbidden, 351
local, 356
standard, 351–6
distribution list, 353
enclosures, 353
house style, 356–64
\LaTeX{} standard, 351–6
letterhead, company, 356–64
postscript, 353
recipient name and address, 352
reference entry, 356
return name and address, 351
salutation, 353
stickers, 356
subject entry, 356
list labels, 69
changing style, 72
in lists, 74
list markers, see list labels
list of references, see bibliography
\listfigurename{}, 459, 555
\listfiles, 208, 441, 442, 447, 555
listings, 69–74
\listof (float), 179
\listoffigures, 59, 210, 399, 427, 556
\listoftables, 59, 210, 399, 427, 556
\listparindent, 76, 79, 556, 604
lists, 74–80
format diagram, 76
nested, 71, 72, 79
numbered, 75
standard label, 74
style parameters, 75
trivial, 79
user's environment, 78
\listtablename, 459, 556
literal text, printing, 110, 393
live.pdf, 383
\lhd, 126, 554
\ll, 126, 554
\llap (\TeX{}), 348
\ln, 128, 556
\LoadClass, 359, 398, 421, 422, 442, 443, 556
\LoadClassWithOptions, 442, 454, 556
\location, 352, 556
.\lof{} file, 59, 179, 399, 436
\log{}, 128, 556
.\log{} file, 14, 246, 399, 401, 403, 414, 429
\Longrightarrow, 117, 127, 557
\longrightarrow, 127, 557
longtable environment
(\longtable{}), 108, 519
longtable package, 108, 395
.\lot{} file, 59, 399, 436
Lower Sorbian, 252
lowering text, see text
lowering, in formulas, see formula
\text{£ (LaTeX), 557
LR box, 85, 87
lrbox environment, 87, 418, 519
lscape package, 159
ltxdoc class, 392, 397, 447, 471
ltxdoc.cfg, 472
ltxdoc.dtx, 472
\text{lower}, \text{in formulas, see formula}
\text{MathML, 480
Maltese, 462
Malyshiev, Basil, 506
.map file, 234
mapsto, 127, 558
Marchal, B., 606
margin, 48–50
margin mark, 116
marginal notes, 116–18
marginpar, 116, 117, 417, 419, 420, 430, 558
marginparpush, 118, 558
marginparsep, 50, 118, 558
marginparwidth, 50, 118, 228, 558
margins (pdfscreen), 342
markboth, 43, 454, 558
markright, 43, 58, 454, 558
markup, 4–5, 11, 14, 17, 197, 207
logical, 5, 7, 61, 112, 456, 479
typographical, 4, 28, 479
markup languages, 4
math accents, 129
math alphabets, 133, 258, 373
math environment, 119, 519
math equation, see formula
math fine-tuning, 145–8
bracket sizing, 148
font size, 146–7
horizontal spacing, 145, 269
with A_{\text{MS}}, 269
math formula, see formula
math mode, 119
math standards, 144
math style parameters, 148
math symbols, see symbols
math variable, see formula
math version, 373
\text{mathalpha}, 375
\text{mathbb (A_{\text{MS}}), 284
\text{mathbf}, 133, 143, 258, 373, 416, 422, 558
\text{mathbin}, 375
\text{mathcal}, 125, 133, 258, 284, 373, 559
\text{mathclose}, 375
\text{mathfrak}, 285
\text{mathindent}, 40, 120, 124, 142, 149, 280, 559
\text{mathit}, 133, 258, 373, 416, 559
MathML, 480
\text{mathnormal}, 125, 133, 373, 559
\text{mathop}, 375
\text{mathopen}, 375
\text{mathord}, 375
\text{mathpunct}, 375
\text{mathrel}, 375
\text{mathring}, 129, 559
\text{mathrm}, 123, 133, 143, 144, 258, 373, 559
mathscr option, 284
\text{makebst.tex, 321
\text{makeglossary}, 230, 398, 416, 557
makeidx package, 228, 392, 399, 460
MakeIndex, 15, 228–9, 382, 389, 392, 399, 470, 550
options, 229
style file, 229
\text{makeindex}, 226, 228, 396, 399, 416, 557
makeindex.tex, 229
\text{makelabel}, 75, 77, 557
\text{makelabels, 355, 557
MakeLowercase, 455, 557
\text{MakeShortVerb (shortvrb), 111, 393, 468, 558
\text{maketitle, 52–4, 54, 69, 419, 431, 558
\text{MakeUppercase, 455, 457, 458, 558
Maltese, 462
Malyshiev, Basil, 506
.map file, 234
mapsto, 127, 558
Marchal, B., 606
margin, 48–50
margin mark, 116
marginal notes, 116–18
marginpar, 116, 117, 417, 419, 420, 430, 558
marginparpush, 118, 558
marginparsep, 50, 118, 558
marginparwidth, 50, 118, 228, 558
margins (pdfscreen), 342
markboth, 43, 454, 558
markright, 43, 58, 454, 558
markup, 4–5, 11, 14, 17, 197, 207
logical, 5, 7, 61, 112, 456, 479
typographical, 4, 28, 479
markup languages, 4
math accents, 129
math alphabets, 133, 258, 373
math environment, 119, 519
math equation, see formula
math fine-tuning, 145–8
bracket sizing, 148
font size, 146–7
horizontal spacing, 145, 269
with A_{\text{MS}}, 269
math formula, see formula
math mode, 119
math standards, 144
math style parameters, 148
math symbols, see symbols
math variable, see formula
math version, 373
\text{mathalpha}, 375
\text{mathbb (A_{\text{MS}}), 284
\text{mathbf}, 133, 143, 258, 373, 416, 422, 558
\text{mathbin}, 375
\text{mathcal}, 125, 133, 258, 284, 373, 559
\text{mathclose}, 375
\text{mathfrak}, 285
\text{mathindent}, 40, 120, 124, 142, 149, 280, 559
\text{mathit}, 133, 258, 373, 416, 559
MathML, 480
\text{mathnormal}, 125, 133, 373, 559
\text{mathop}, 375
\text{mathopen}, 375
\text{mathord}, 375
\text{mathpunct}, 375
\text{mathrel}, 375
\text{mathring}, 129, 559
\text{mathrm}, 123, 133, 143, 144, 258, 373, 559
mathscr option, 284
INDEX

netpbm library, 476
network servers, 390
New Font Selection Scheme, see
NFSS, see NFSS
\newboolean (ifthen), 194, 195,
359, 451, 452, 561
newcent package, 234
NewCenturySchlbk, PostScript font,
234, 504
\newcolumntype (array), 107
\newcommand, 25, 185, 202, 416,
418, 425, 426, 438, 449,
562
\newcommand*, 445, 562
\newcounter, 21, 75, 77, 182,
203, 416, 418, 562
not with \include, 210
\newenvironment, 78, 196, 202,
416, 418, 425, 439, 562
\newenvironment*, 445, 562
\newfloat (float), 179
\newfont, 66, 364, 373, 416, 562
\newif (\TeX), 451
\newlength, 21, 185, 361, 416, 426,
562
\newline, 31, 32, 419, 434, 563
\newpage, 33, 34, 60, 84, 108, 428,
563

forbidden, 84
\newpagemode (seminar), 335
\newsavebox, 21, 87, 106, 199, 203,
300, 361, 416, 426, 563
\newslide (seminar), 331, 336
\newslideframe (seminar), 334
\newtheorem, 80, 81, 214, 395, 416,
563
\newtheorem (amsthm), 281
\newtheorem* (amsthm), 281
\newtheoremstyle (amsthm), 281
NeXt coding, 462
NFSS, 9, 64, 367-80, 398, 484, 489
EC fonts, 379
encoding commands, 379
font attributes, 64-5, 368-72
defaults, 372
font definition, 376-9
installing, 372-80
\nfssfont.tex, 385, 391
\NG, 503, 563
\ng, 503, 563
\ni, 126, 563
\nobottombuttons (pdfscreen),
342
\nbreak, 364
\nocite, 310, 563
\noexpand (\TeX), 451
\noextras lang (non-standard), 255
\nofiles, 396, 397-400, 416, 436,
563
\noindent, 46, 563
nointlimits option, 279
\nolimits, 128, 280, 563
\nolinebreak, 32, 33, 564
nonamelimits option, 280
\nourfrenchspacing, 29, 564
\nonumber, 139, 142, 564
\nopagebreak, 33, 564
\normalcolor (color), 167, 564
\normalfont, 65, 167, 371, 455,
458, 564
\normalmarginpar, 117, 564
\normalsize, 63, 371, 412, 424,
458, 564
North Sami, 252
Norwegian, 252
nosumlimits option, 279
\not, 126, 564
\notag (\AMS), 271, 565
\notag env (seminar), 337
\notag env (slides), 326,
520
\notin, 126, 565
notitlepage option, 39
\noxcomment (seminar), 337
\nu, 125, 565
null.tex, 413
numbering in lists, 75
\numberwithin (\AMS), 277, 565
\narrow, 127, 565
\O, 24, 379, 565
\o, 24, 565
\oddsidemargin, 48, 50, 361, 453,
565
\odot, 126, 565
\OE, 24, 565
\oe, 24, 565
ogonek accent, 503
\oint, 128, 565
\Omega, 125, 565
\omega, 125, 565
\ominus, 126, 565
one-column page formatting
for single pages, 51
onecolumn option, 38
\oneline, 51, 565
oneside option, 38
\OnlyDescription(doc), 468, 472
\onlynotes(slides), 328, 565
\onlyslides(seminar), 326
\onlyslides(slides), 327, 565
Oostrum, Piet van, 43, 454
openany option, 39
openbib option, 39
\opening, 353, 355, 356, 363, 566
openright option, 39
oplus, 126, 566
\OptionNotUsed, 443, 566
options
defining, 442
global, 42, 443
local, 42, 444
processing, 443
to classes, 37
to packages, 41
\or(\TeX), 451
\original\TeX(babel), 255
\oslash, 126, 566
OT1 encoding, 490, 492, 506
ot1cmr.fd, 378, 398
ot1enc.def, 236, 378
ot1ptm.fd, 236
OT2 encoding, 283, 497
otherlanguage environment
(babel), 254
otherlanguage* environment
(babel), 254
\otimes, 126, 566
outline fonts, see type 1 fonts
\oval(eepic), 305
\oval(picture), 296, 297, 300, 431, 566
\Ovalbox(fancybox), 94, 566
\ovalbox(fancybox), 94, 566
\overbrace, 136, 566
overbraces in formulas, 136
Overfull \hbox warning, 433
Overfull \vbox warning, 434
overlapping text, 86, 160, 348
overlay environment (slides), 326, 521
\overlay(pdfscreen), 342
\overlay(seminar), 337
\overleftarrow_AMS, 262, 566
\overrightarrow_AMS, 262, 566
\overline, 136, 567
overlining, 136
\overleftrightarrow_AMS, 262, 567
\overset_AMS, 262, 567
oztex, graphics option, 154
\P, 23, 567
package, 8, 9, 12, 13, 41
afterpage, 171, 394
alltt, 111, 392
amsbsy, 144, 192, 258–9
amscd, 258, 263, 282
amsfonts, 126, 284–5, 422, 494
amsmath, 192, 258
amsopn, 258, 267–8
amssymb, 284, 285
amstext, 258–9
amsthm, 81, 258, 280–2, 395
apalike, 220
array, 107, 394
avant, 234
babel, 252–6
background, 345–6
bm, 144, 394
bookman, 234
calc, 394
chapterbib, 224
chicago, 220
color, 153, 166, 324, 325, 335, 338, 345, 439
dcolumn, 107, 394
delarray, 108, 394
doc, 392, 399, 441, 467–73
documentation, 13
eepic, 305–6
eepicemu, 305
<table>
<thead>
<tr>
<th>Package</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>enumerate</td>
<td>74, 394</td>
</tr>
<tr>
<td>epic</td>
<td>302-5</td>
</tr>
<tr>
<td>epsf</td>
<td>159</td>
</tr>
<tr>
<td>epsfig</td>
<td>159</td>
</tr>
<tr>
<td>eso-pic</td>
<td>346</td>
</tr>
<tr>
<td>eucal</td>
<td>284-5</td>
</tr>
<tr>
<td>eufrak</td>
<td>285</td>
</tr>
<tr>
<td>europs</td>
<td>25, 387</td>
</tr>
<tr>
<td>eurosans</td>
<td>25, 387</td>
</tr>
<tr>
<td>eurosym</td>
<td>25, 28</td>
</tr>
<tr>
<td>exscale</td>
<td>392</td>
</tr>
<tr>
<td>fancybox</td>
<td>94-5, 334, 338, 345</td>
</tr>
<tr>
<td>fancyhdr</td>
<td>43-5, 249, 335, 454</td>
</tr>
<tr>
<td>flafter</td>
<td>171, 392</td>
</tr>
<tr>
<td>float</td>
<td>179-80</td>
</tr>
<tr>
<td>floatfig</td>
<td>see floatflt</td>
</tr>
<tr>
<td>floatflt</td>
<td>178-9</td>
</tr>
<tr>
<td>fontenc</td>
<td>378-9, 392, 501</td>
</tr>
<tr>
<td>fontsml</td>
<td>394</td>
</tr>
<tr>
<td>french</td>
<td>251</td>
</tr>
<tr>
<td>ftnright</td>
<td>394</td>
</tr>
<tr>
<td>fullpage</td>
<td>452, 465</td>
</tr>
<tr>
<td>geometry</td>
<td>49-51, 452</td>
</tr>
<tr>
<td>german</td>
<td>251, 252, 360</td>
</tr>
<tr>
<td>graphics</td>
<td>153, 155-7</td>
</tr>
<tr>
<td>graphicx</td>
<td>157-9</td>
</tr>
<tr>
<td>graphpap</td>
<td>301, 392</td>
</tr>
<tr>
<td>harvard</td>
<td>220</td>
</tr>
<tr>
<td>helvet</td>
<td>234</td>
</tr>
<tr>
<td>here</td>
<td>178</td>
</tr>
<tr>
<td>hhline</td>
<td>395</td>
</tr>
<tr>
<td>html</td>
<td>476</td>
</tr>
<tr>
<td>hyperref</td>
<td>112, 161, 220, 239-49, 330, 339, 340, 344</td>
</tr>
<tr>
<td>ifthen</td>
<td>193-5, 359, 392, 439, 452</td>
</tr>
<tr>
<td>indentfirst</td>
<td>46, 395</td>
</tr>
<tr>
<td>inputenc</td>
<td>26, 392, 461-2</td>
</tr>
<tr>
<td>latexsym</td>
<td>126, 285, 392, 422, 494</td>
</tr>
<tr>
<td>layout</td>
<td>47, 395</td>
</tr>
<tr>
<td>longtable</td>
<td>108, 395</td>
</tr>
<tr>
<td>landscape</td>
<td>159</td>
</tr>
<tr>
<td>makeidx</td>
<td>228, 392, 399, 460</td>
</tr>
<tr>
<td>multicol</td>
<td>51-2, 395</td>
</tr>
<tr>
<td>natbib</td>
<td>219-24, 476</td>
</tr>
<tr>
<td>newcent</td>
<td>234</td>
</tr>
<tr>
<td>palatino</td>
<td>234</td>
</tr>
<tr>
<td>parskip</td>
<td>46-7</td>
</tr>
<tr>
<td>pause</td>
<td>346-9</td>
</tr>
<tr>
<td>pdfscreen</td>
<td>340-3</td>
</tr>
<tr>
<td>pp4link</td>
<td>349</td>
</tr>
<tr>
<td>pstricks</td>
<td>308</td>
</tr>
<tr>
<td>rotating</td>
<td>159</td>
</tr>
<tr>
<td>sempdftx</td>
<td>340</td>
</tr>
<tr>
<td>shortvrb</td>
<td>111, 393</td>
</tr>
<tr>
<td>showidx</td>
<td>228, 392</td>
</tr>
<tr>
<td>showkeys</td>
<td>395</td>
</tr>
<tr>
<td>somedefs</td>
<td>395</td>
</tr>
<tr>
<td>standard</td>
<td>392</td>
</tr>
<tr>
<td>syntonly</td>
<td>393</td>
</tr>
<tr>
<td>tlenc</td>
<td>393</td>
</tr>
<tr>
<td>tabularx</td>
<td>107, 395</td>
</tr>
<tr>
<td>TeX4ht</td>
<td>477</td>
</tr>
<tr>
<td>textcomp</td>
<td>24-5, 393, 503</td>
</tr>
<tr>
<td>theorem</td>
<td>81, 282, 395</td>
</tr>
<tr>
<td>times</td>
<td>234</td>
</tr>
<tr>
<td>tracefnt</td>
<td>393, 433</td>
</tr>
<tr>
<td>upref</td>
<td>282</td>
</tr>
<tr>
<td>url</td>
<td>112</td>
</tr>
<tr>
<td>varioref</td>
<td>215-16, 253, 396</td>
</tr>
<tr>
<td>verbatim</td>
<td>111, 118, 396</td>
</tr>
<tr>
<td>with options</td>
<td>41</td>
</tr>
<tr>
<td>xr</td>
<td>215, 243, 396</td>
</tr>
<tr>
<td>xspace</td>
<td>186, 396</td>
</tr>
<tr>
<td>\PackageError</td>
<td>439, 446, 567</td>
</tr>
<tr>
<td>\PackageInfo</td>
<td>447, 567</td>
</tr>
<tr>
<td>packages, catalogue of, 13</td>
<td></td>
</tr>
<tr>
<td>\PackageWarning</td>
<td>439, 446, 567</td>
</tr>
<tr>
<td>\PackageWarningNoLine</td>
<td>446, 567</td>
</tr>
<tr>
<td>page breaking</td>
<td>33</td>
</tr>
<tr>
<td>discouraged</td>
<td>33</td>
</tr>
<tr>
<td>encouraged</td>
<td>33</td>
</tr>
<tr>
<td>forced</td>
<td>33</td>
</tr>
<tr>
<td>with justification</td>
<td>33</td>
</tr>
<tr>
<td>two columns</td>
<td>34</td>
</tr>
<tr>
<td>two-sided</td>
<td>34</td>
</tr>
<tr>
<td>with tabbing</td>
<td>84</td>
</tr>
<tr>
<td>with figures and tables</td>
<td>34</td>
</tr>
<tr>
<td>page counter</td>
<td>45, 181, 183, 190, 195</td>
</tr>
<tr>
<td>page format</td>
<td>47</td>
</tr>
<tr>
<td>diagrams</td>
<td>48</td>
</tr>
<tr>
<td>full page</td>
<td>452-4</td>
</tr>
<tr>
<td>one-column</td>
<td>452-4</td>
</tr>
<tr>
<td>one-sided</td>
<td>48, 51</td>
</tr>
<tr>
<td>setting values</td>
<td>47</td>
</tr>
</tbody>
</table>
two-column, 38, 51
two-sided, 38, 50
with geometry, 49–51

page numbering
alphabetic, 45
Arabic, 45
changing, 45
in heading, 42
Roman numerals, 20, 45
setting, 45
suppression of, 42

page style
align (seminar), 335
customizing, 43, 454
empty, 42, 327, 335, 352, 452
fancy, 43
firstpage, 362
headings, 42, 44, 55, 58, 327, 335, 352, 363, 452
myheadings, 42, 45, 58, 335, 452, 454
plain, 42, 45, 327, 335, 352, 363, 452
in letters, 352

page transitions, PDF, 245, 344
\pagebreak, 33, 34, 84, 178, 278, 567
no effect, 84
\pagecolor (color), 167, 335, 338, 567
\PageIndex (doc), 469
\pagenumbering, 20, 21, 45, 58, 60, 568
\pageref, 56, 178, 209, 213, 214, 247, 282, 395, 396, 430, 431, 568
\pageref* (hyperref), 247
\pagestyle, 42, 58, 326, 327, 335, 352, 452, 455, 568
pagetranstex, 344
pagination, see page numbering
palatino package, 234
Palatino, PostScript font, 234, 504
\paneloverlay (pdfscreen), 343
\panelwidth (pdfscreen), 342
paper size, option, 38
\paperheight, 48, 50, 452, 453, 568, 604
\paperwidth, 48, 50, 195, 452, 453, 568, 604
\par, 23, 364, 419, 426, 445, 568
paragraph, 11
box, 85, 88
paragraph counter, 57, 181
paragraph character, 23
paragraph termination
with blank line, 11
with \par, 23
\paragraph, 55, 568
\paragraph*, 55, 569
paragraphing
extra line space, 11, 22, 46
indentation, 11, 46
after section command, 46
setting, 20, 46
suppression of, 46
\parallel, 126, 569
\parbox, 88, 89, 90, 92, 142, 151, 174, 290, 292, 362, 417, 419, 439, 569
\parindent, 20, 46, 79, 569
\parsep, 76, 77, 79, 569, 604
parskip package, 46–7
\parskip, 22, 46, 79, 184, 364, 569, 604
part counter, 57, 181
\part, 55, 569
\part*, 55, 569
\partial, 127, 569
Partl, H., 46, 251
\partname, 460, 569
\partopsep, 75, 570, 604
Passive\TeX, 480
\PassOptionsToClass, 359, 443, 570
\PassOptionsToPackage, 443
Patashnik, Oren, 321
\path, 568
\path (eepic), 306
\path (url), 112
pause package, 346–9
\pause (pause), 346
\pausebuild (pause), 348
\pausecolors (pause), 348
\pausehighlight (pause), 348
\pauselevel (pause), 347
.pbm file, 476
pc (pica), 21
pctex32, graphics option, 154
pctexhp, graphics option, 154
pctexps, graphics option, 154
pctexwin, graphics option, 154
PDF, 3, 8, 9, 16, 162–3, 231, 236–9, 242–6, 307, 338, 339, 480, 481, 498
slide show, 323, 330, 333, 338
with PostScript fonts, 234, 505
with PPowe4, 343
.pdf file, 162, 237
pdf\LaTeX, 238, 324
pdflatex format, 238, 385
pdflatex.fmt, 385
\pdfoutput, 238
\pdfpageheight (pdf\LaTeX), 339
\pdfpagewidth (pdf\LaTeX), 339
pdfscreen package, 340–3
pdf\LaTeX, 6, 16, 238–9, 305, 346, 385, 387, 389, 481
graphics inclusion, 162–3
with seminar, 339
pdftex, graphics option, 154, 238
pdftex.cfg, 238
pdftex.d11, 386
Perl, 476
\perp, 126, 570
.pfa file, 389
.pfb file, 234, 389, 506
\Phi, 125, 570
\phi, 125, 570
\Pi, 125, 570
\pi, 125, 570
\picsquare (epic), 302, 304
picture
arrow, 294
allowed slopes, 295
min. length, 295
box, 291–3
dashed, 291, 292
examples, 291, 292
framed, 291
parbox inside, 292
positioning argument, 291
unframed, 291, 292
with text, 291
circle, 295
filled, 295
open, 295
commands, 289–300
coordinate system, 287
curves, 299
elements, 289
arrow, 294
box, 291–3
circle, 295
half circle, 297
line, 294
oval, 296
positioning, 289
quarter circles, 297
repeated, 289
text, 290
floating, see float
framed text, 298
half circles, 297
length unit, 287, 288
limited command set, 289
line, 294
allowed slopes, 293, 294
min. length, 294
sloping, 293
vertical, 293
line thickness, 289, 300
multiline text, 290
ref. point, 290
oval, 296
examples, 296, 297
positioning, 287
positioning commands, 289
quarter circles, 297
saving, 300
shifting, 301
superimposing images, 160
vertical text, 297
x-axis, 287
y-axis, 287
picture environment, 93, 160, 175, 288, 289, 295, 298, 302, 428, 431, 521
with offset, 301
pixel fonts, 8
.pk file, 233, 234, 236, 388, 400, 488, 498, 504
plain bibliography style, 219, 310
plain format, 398
plain page style, 42, 45, 327, 335, 352, 363, 452
plain.fmt, 398
plainnat bibliography style, 219, 221, 311
Plain TeX, 7, 398, 408
plus, 22, 47, 77, 78
\pm, 126, 570
pmatrix environment (AMS), 266, 521
\pmb (AMS), 258, 570
\pmod, 129, 268, 570
PNG images, 162, 237, 238
.png file, 162
\pod (AMS), 268, 570
Podar, Sunil, 302
poetry, 68
pointers, 127
Polish, 252
Popineau, Fabrice, 381
\poptabs, 83, 84, 419, 570
Portable Document Format, see PDF
\portraitonly (seminar), 332
Portuguese, 252
\postamble (DocStrip), 465
PostScript, 3, 15, 64, 154, 162, 163, 231–6, 239, 308, 335, 337, 338, 367, 370, 475, 498
compressing, 166
driver, 153, 154, 162, 165, 168, 231
encapsulated, 153, 155, 162–5, 232
euro fonts, 25, 387
fonts, 233–6, 238, 239, 331, 369, 372, 378, 389, 503–5
installing, 234
printer, 232, 236, 305
\TeX fonts, 235, 505
under NFSS, 233, 378
viewer, 164, 232, 305
pound sign, 23, 27, 28, 570
\pounds, 23, 27, 570
pp4link package, 349
.ppm file, 478
PPower4, 343–9
\Pr, 128, 129, 280, 570
preamble, 12, 208
\preamble (DocStrip), 465
\prec, 126, 571
\preceq, 126, 571
\preceq, 126, 571
\preceq, 126, 571
\preceq, 126, 571
\prefacename, 460
preload.cfg, 385
preload.ltx, 385
presentation material, see slides
\prime, 127, 571
print environment (pdftex), 343
\PrintChanges (doc), 470
printer driver, 15, 153, 389, 391, 398, 453, 488, 497
\PrintIndex (doc), 470
\printindex, 210, 228, 399, 571
\printlandscape (seminar), 333
proc class, 391
processing messages, 399, 414
\ProcessOptions, 359, 443, 447, 571
\ProcessOptions*, 443, 444, 453, 571
\prod, 128, 137, 571
programming \LaTeX, 437–51
deferred coding, 444
durability, 440
error messages, 446
examples
custom headline, 454–6
customized letter, 359–64
full page, 452–4
sectioning, 456–9
file checking, 440–2
guidelines, 438
loading files, 442
options, 442–4
portability, 437
robust commands, 444
warnings, 446
proof environment (amsthm), 282
\proofname, 460
\propto, 126, 571
\protect, 211, 426, 429, 440, 445, 446, 535, 571
protected space, 22, 29, 144
protocol, see transcript file
\providecommand, 187, 363, 439, 449, 571
\providecommand*, 445, 571
INDEX 637

\ProvidesClass, 359, 416, 432, 441, 454, 571
\ProvidesFile, 442, 448, 472, 571
\ProvidesPackage, 416, 432, 441, 572
\ProvideTextCommand, 379, 572
\ProvideTextCommandDefault, 380, 572
\ps, 353, 572
.ps file, 232, 236
\psfig (epsfig), 159
\psfonts.map, 235
\Psi, 125, 572
\psi, 125, 572
\psnfss, PostScript support, 233
\pstricks package, 308
\pt (point), 21
\ptsize (seminar), 331
\put (picture), 161, 289, 290–9, 302, 572
\putfile (epic), 302, 304
\raisebox, 87, 88, 92, 102, 104, 362, 573
\raisetag (AMS), 271
raising text, see text
raising, in formulas, see formula
\range, 132, 270, 573
\rceil, 132, 573
\Re, 127, 573
Reckdahl, Keith, 606
\RecordChanges (doc), 470
\ref, 56, 131, 139, 177, 182, 209, 214, 247, 278, 282, 395, 396, 430, 431, 573
\ref* (hyperref), 247
reference, cross, see cross-reference
references, list of, see bibliography
\reflectbox (graphics), 156, 573
\refname, 460, 574
\refstepcounter, 182, 574
\reftext... (varioref), 216
relational symbols, 126
\relax (TEX), 451
\renewcommand, 44, 185, 202, 415, 425, 426, 438, 574
\renewcommand*, 445, 574
\renewenvironment, 196, 203, 415, 425, 439, 574
\renewenvironment*, 445, 574
\renewpagestyle (seminar), 335
\report class, 37, 41, 55, 113, 130, 173, 183, 187, 191, 218, 224, 225, 277, 391, 459
reqno option, 271, 272, 280
\RequirePackage, 400, 421, 432, 442, 574
\RequirePackageWithOptions, 442, 574
\resizebox (graphics), 156, 574
\resizebox* (graphics), 574
\restylefloat (float), 180
\reversemarginpar, 117, 575
\rfloor, 132, 575
\rfoot (fancyhdr), 44
\rhead (fancyhdr), 44
\rho, 125, 575
\right, 108, 131, 134, 137, 140, 575
\Rightarrow, 74, 127, 575
\rightarrow, 127, 575

r.txt, 413
\r (˚ accent), 24, 573
Radhakrishnan, C. V., 340
\raggedbottom, 48, 573
\raggedleft, 67, 573
\raggedright, 67, 573
\raggedslides (seminar), 334
Rahtz, Sebastian, 154, 159, 239, 330, 382, 480, 605
Raichle, Bernd, 251

\quotedblbase, 503, 573
\quotesinglbase, 503, 573

r.tex, 413

\text, 22, 573
\textbf, 21, 573
\textit, 21, 573
\texttt, 112, 269, 273, 573

\rightharpoonup, 127, 575
\rightleftharpoons, 127, 575
\rightmargin, 76, 77, 79, 575, 604
\rightmark, 44
\rightbox (\TeX), 348
\rm (\LaTeX 2.09), 367, 485
\rmdefault, 233, 372, 379, 575
\rmfamily, 64, 371, 379, 575
robust commands, 440, 445
Rokicki, Tomas, 159, 162, 231, 606
Roman page numbering, 45, 58, 60
\Roman, 73, 75, 114, 183, 192, 575
\roman, 73, 114, 183, 575
Romanian, 252
roots, 122, 269
Rose, Kristoffer H., 307
rotate environment (rotating), 159
\rotatebox (graphics), 156, 576
\rotatebox (graphicx), 159
\rotateheaderstrue (seminar), 333
rotating graphics and text, 156
rotating package, 159
Rowley, Chris, 8
\rq (\TeX), 576
rubber lengths, 21, 22, 30, 32, 46, 77, 85, 149, 173, 184, 457
rule box, 85
\rule, 18, 91, 106, 116, 173, 332, 362, 364, 455, 576
rules
horizontal, 91
vertical, 91
Russian, 252, 283
\rVert (\AMS), 270, 576
\rvert (\AMS), 270, 576
\S, 23, 576
s.tex, 394, 413
Saarela, J., 605
Samarin, A., 605
sample2e.tex, 385, 391
Samuel, Arthur L., 606
\savebox, 87, 93, 203, 576
\savebox (picture), 300, 576
\sb (\TeX), 576
sbox, 87, 93, 106, 199, 361, 439, 576
\sc (\LaTeX 2.09), 367, 485
\scalebox (graphics), 156, 576
scaling graphics and text, 156
\scdefault, 372, 576
Schenk, Christian, 381
Schmidt, Walter, 25
Schöpf, Rainer, 8, 257, 367
Schrod, J., 251
Schwarz, Norbert, 499, 606
Scientific Workplace, 382
Scottish Gaelic, 252
screen environment (pdfscreen), 343
\screensize (pdfscreen), 342
\scriptscriptstyle, 146, 375, 577
\scriptsize, 63, 371, 577
\scriptstyle, 146, 282, 375, 577
\scshape, 343, 372, 379, 577
\searrow, 127, 577
sec, 128, 577
secnumdepth counter, 56, 58, 458
section counter, 57, 181, 183, 277
\section, 44, 55, 58, 113, 455, 577
\section*, 55, 577
sectioning
*-form, 55, 56
hierarchy, 55
numbering, 55
depth of, 56
secnumdepth, 57
suppression of, 55
title, 55
short form, 58
sectioning commands, 55
redefining, 456
\see, 227, 228, 577
\seealso, 227, 577
\seename, 460, 577
\selectfont, 369, 370, 372, 376, 578
selective file processing, 208
controlling, 210, 212
\selectlanguage (babel), 254, 256, 578
sem-user.tex, 330
\semcm (seminar), 332
\semin (seminar), 332
seminar
\addtoartlength, 332
\addtoslidelength, 332
\addtoslidereset, 337
\articlemag, 332
\centerslidefalse, 334
\centerslidetrue, 334
\extraslideheight, 336
\landscapeonly, 332
\newpagestyle, 335
\newslide, 331, 336
\newslideframe, 334
\noxcomment, 337
\onlyslides, 336
\overlay, 337
\portraitonly, 332
\printlandscape, 333
\ptsize, 331
\raggedslides, 334
\renewpagestyle, 335
\rotateheaderstrue, 333
\semcm, 332
\semin, 332
\setartlength, 332
\setslidelength, 332
\slidebottommargin, 334
\slidefootfont, 336
\slideframe, 331, 334
\slideframesep, 334
\slideframewidth, 334
\slideheadfont, 336
\slideleftmargin, 334
\slidepagestyle, 336
\slidereset, 337
\sliderrightmargin, 334
\slidesmag, 331
\slidetopmargin, 334
\thenote, 335, 337
\theslide, 335, 337

seminar class, 330–40
seminar.con, 339, 340
sempdftx package, 340
sempdftx.sty, 340
Serbian, 252
series, font attribute, 64, 368
\seriesdefault, 372
servers, see network servers
\setartlength (seminar), 332
\setboolean (ifthen), 194, 359, 451, 452, 578
\setbox (\TeX), 439
\setcounter, 21, 182, 183, 203, 418, 578
\setlength, 20–2, 75, 83, 173, 184, 200, 439, 578
\SetMathAlphabet, 374, 423, 578
\setminus, 126, 578
\setminus (slides), 327, 578
\settodepth, 184, 579
\settowidth, 184, 189, 200, 579
\sf (\LaTeX 2.09), 367, 485
\sf@size, 376
\sfdefault, 233, 372, 379, 579
\sfamily, 64, 66, 325, 371, 379, 579
\shadowbox (fancybox), 94, 335, 579
\shadowsize (fancybox), 94, 579
\shape, font attribute, 64, 368
\shapedefault, 372
\sharp, 127, 579
shorthand.tex, 322
\shortstack, 297, 298, 299, 579
shortvrb package, 111, 393
\shoveleft (\AMS), 271
\shoveright (\AMS), 271
\showidx package, 228, 392
\showkeys package, 395
\sidestack (\AMS), 262, 579
\sideways environment (rotating), 159
\Sigma, 125, 579
\sigma, 125, 579
\signature, 351, 352, 353, 358, 579
\sim, 126, 580
\simeq, 126, 580
Simonic, Aleksander, 15
\sin, 128, 580
\size, font attribute, 64, 368
\sl (\TeX 2.09), 367, 485
\sldefault, 372, 580
slide environment (seminar), 331, 332, 521
slide environment (slides), 325, 521
slide* environment (seminar), 331, 332
\slidebottommargin (seminar), 334
\slidefootfont (seminar), 336
\slideframe (seminar), 331, 334
\slideframesep (seminar), 334
\slidewidth (seminar), 334
\slideheight (seminar), 336
\slidelength (seminar), 334
\slidemargin (seminar), 334
\slidemarginbottom (seminar), 334
\slidemarginleft (seminar), 334
\slidemarginright (seminar), 334
\slidemarginupper (seminar), 334
\slidemarginwidth (seminar), 334
\slideleftmargin (seminar), 334
\slidemargin (seminar), 334
\slideheadfont (seminar), 336
\slidetopmargin (seminar), 334
\slideheadsep (seminar), 334
\slideheadwidth (seminar), 334
\slidetitlefont (seminar), 336
\slideframewidth (seminar), 334
\slides (seminar), 323–40
leading page, 325
notes, 326
overlays, 325–6
page styles, 326
selective processing, 327
special environments, 325–6
timing notes, 327
slides class, 324–30, 392
\slidesmag (seminar), 331
\slidetopmargin (seminar), 334
Su\TeX, see slides
\slapproach, 36, 88, 434, 580
\sloppypar environment, 36, 198, 434, 521
Slovakian, 252
\Slovenian, 252
\sshape, 64, 78, 199, 363, 371, 580
\small environment, 110, 517
\small, 19, 63, 371, 580
\small2e.tex, 385, 391
\smallmatrix environment (\AMS), 267
\smallskip, 33, 580
\smallskipamount, 580
\smash (\AMS), 269, 580
\smile, 126, 580
Snow, W., 606
Sojka, Petr, 606
\somedefs package, 395
source file, 5, 6
source text, 4, 11, 14, 17
\sp (\TeX), 580
space, see blank
\space, 446
spacing
character, 29
French spacing, 29
horizontal, 29, 30
line
local change, 32
standard, 63
manual adjustment, 28
paragraph, 11, 22, 46
unwanted, 201
vertical, 31–3
rubber, 32
word, 28
after period, 29
after punctuation, 28
\spadesuit, 127, 580
Spanish, 252
Spannagel, Christian, 343
special characters, 23
special letters in other languages, 24
\special (\TeX), 153, 232, 305, 477
spelling, American or British, 195
Spivak, Michael, 257
\spline (eepic), 306
\split environment (\AMS), 270, 272, 276, 521
\sqcap, 126, 580
\sqcup, 126, 580
\sqrt, 122, 269, 580
\sqsubset, 126, 581
\sqsupset, 126, 581
\sqsupseteq, 126, 581
\sqsubseteq, 126, 581
\ss, 24, 576
\ss, 24, 581
\sst@size, 376
\ssf@size, 376
\stackrel, 137, 581
\star, 126, 581
\stepcounter, 110, 115, 182, 581
\StopEventually (doc), 469
\stretch, 184, 581
\strut, 92, 103, 105
\strut, 298
\st@file, 9, 12, 41, 384, 385, 463, 484
style files (\TeX 2.09), 483
INDEX  641

style parameter
  float, 171
  footnote, 117
  frame box, 93
  lists, 75
  marginal note, 118
  math, 148
  table, 98

subarray environment (AMS), 261, 522
subequations environment (AMS), 277, 522
\subitem, 225, 262, 581
\subparagraph counter, 57, 181
\subparagraph, 55, 581
\subparagraph*, 55, 581
  subscript, 121
  subsection counter, 57, 181
\subsection, 44, 55, 181, 581
\subsection*, 55, 581
\subsubsection, 55, 582
\subset, 126, 582
\subseteq, 126, 582
\substack (AMS), 261, 582
\subsection, 55, 582
\subseteq, 126, 582
\surd, 127, 582
\sum, 123, 128, 137, 582
\sumlimits option, 279
\sup, 128, 129, 280, 582
\superscript, 121
\suppressfloats, 171, 175, 582
\supset, 126, 582
\supseteq, 126, 582
\surd, 127, 582
\texttt{Sutor, Robert}, 605
\swapnumbers (amsthm), 281
\swarrow, 127, 582
Swedish, 252
\switch.def (babel), 253
\symbol, 66, 88, 582

symbols
  above one another, 137, 261
  alternative commands, 462
  arrows, 127, 262, 282
  binary operators, 125
  \texttt{brackets}, 131
  \texttt{automatic size}, 131
  \texttt{invisible}, 132
  \texttt{manual sizing}, 148
  \texttt{multiline equations}, 140, 148, 276
  \texttt{direct input}, 26, 461
  \texttt{integral}, 123, 260
  \texttt{keyboard}, 120
  \texttt{letters}
  \texttt{calligraphic}, 125, 284
  \texttt{Greek}, 125
  \texttt{limits}, 128, 261
  \texttt{miscellaneous}, 127
  \texttt{negation}, 126
  \texttt{pointers}, 127
  \texttt{relational}, 126
  \texttt{summation}, 123
  \texttt{two sizes}, 128
\syntaxonly (syntonly), 393
\texttt{syntonly} package, 393

\texttt{T1 encoding}, 500, 501, 503, 506
\texttt{t1enc} package, 393
\texttt{t1enc.def}, 398, 501
\texttt{t} (~ accent), 24, 582
tab stop, 81
tabbing, 81–5
  \texttt{accents in}, 83
  \texttt{additional commands}, 83
  \texttt{backwards}, 82
  \texttt{end of line}, 81
  \texttt{left margin}, 82
  \texttt{setting}, 82
  \texttt{nesting}, 84
  \texttt{page breaking}, 84
  \texttt{sample line}, 82
  \texttt{setting stops}, 81
  \texttt{storing}, 83
tabbing environment, 81, 83, 84, 415, 419, 420, 428, 522
\texttt{tabbingsep}, 83, 582
\texttt{tabcolsep}, 98, 106, 583
table, 95–110
  \texttt{column alignment}, 101
  \texttt{on decimals}, 107
  \texttt{construction}, 95–8
    \texttt{<!--expression}, 96, 101, 110
  \texttt{column formatting}, 96
column repetition, 96
column separation, 97
horizontal line, 97
math arrays, 133–6
merging columns, 97, 101
paragraph column, 96, 104
row end, 97
vertical line, 96, 97
vertical positioning, 95
examples, see table examples
extension packages, 106, 394
floating, see float, 169
footnotes inside, 116
intercolumn spacing, 96
math, 133–6
multi-page, 108
multipage, 108
numbering, see float, caption
setting width, 96, 107
style parameter, 98
title, see float
table
counter, 181
table environment, 60, 108–10,
  169, 173, 182, 213, 417,
  419, 430, 522
table examples, 98–106
  @-expression, 110
  changing fonts, 102
  column alignment, 101
  column format, 99
  empty box, 105
  floating, 110
  framed, 100
  horizontal line, 100, 102
  math arrays, 133–6
  merging columns, 101
  paragraph column, 104
  line breaking, 105
  row termination, 99
  setting widths, 110
  text between columns, 101
  unframed, 99
  vertical line, 99, 102, 106
  with all table features, 106
table of contents, 55, 58
  automatic entries, 58
  for figures, 59
  for tables, 59
  level of entry, 58
tocdepth, 58
manual entry, 59
printing, 58
table* environment, 169, 522
\tablename, 255, 459, 583
\tableofcontents, 58, 60, 210,
  396, 400, 427, 436, 583
tables, list of, 59
tabular environment, 10, 95, 99,
  102, 107, 116, 179, 297,
  362, 394, 417, 418, 425,
  428, 429, 522
tabular* environment, 95, 107,
  110, 395, 455, 523
\tabularnewline, 98, 105, 583
\tabularx environment, 107, 395
\tabularx package, 107, 395
\tabulator, see tabbing
\tag (AMS), 271, 573, 583
\tan, 128, 583
\tanh, 128, 583
\tau, 125, 583
Taylor, Philip, 606
\tbinom (AMS), 265, 583
\tbtags option, 272, 279
TC fonts, 24, 501
tcidvi, graphics option, 154
.tcx file, 481
TDS, \TeX Directory Structure, 388
techexplorer Hypermedia
  Browser, 481
TEI DTD, 479, 480
\telephone, 352, 356, 583
testpage.tex, 385, 391
te\TeX for Unix, 381
\TeX
  \TeX logo, 19
\TeX, 6, 398
  implementations, 381
\TeX commands, 438
  adjusting bracket sizes, 148
  allowed, 450–1
\atop, 137, 146, 147, 189
  avoiding, 439
\Big, 148, 525
\big, 148, 525
\Bigg, 148, 525
\bigh, 148, 525
\brace, 192
INDEX 643

\brack, 192
\centerline, 67, 176, 408, 527
\choose, 137, 146, 147, 189, 192
\day, 27, 460
\def, 439, 445, 450
\edef, 450
\else, 451
\endinput, 451
\expandafter, 451
\fi, 451
forbidden, 137
\gdef, 450
\hbox, 439
\hoffset, 547, 604
\hss, 408
\if, 451
\ifcase, 451, 461
\iffalse, 468
\language, 36, 251, 255, 256, 553
\let, 451
\llap, 348
\lq, 557
\month, 27, 460
\newif, 451
\noexpand, 451
\or, 451
\relax, 451
\rlap, 348
\rq, 576
\sb, 576
\setbox, 439
\sp, 580
\special, 153, 232, 305, 477
\vbox, 439
\voffset, 593, 604
\xdef, 450
\year, 27, 460
\TeX, 19, 583
tex.d11, 386
\TeX4ht program, 477, 479, 480
\TeX4ht package, 477
\TeX error messages, see error messages
 тек. file, 14, 17, 208, 209, 391, 396, 400, 403, 436
\TeXLive, 3, 13, 15, 160, 344, 381-9, 478, 479, 506
texmf.cnf, 386
texsys.cfg, 384
text
commands in text, 11
comments in, 118
components, 11
framed, 86
in formulas, 133
literal, printing, 110, 393
lowering, 87
math, 119
merging, 207
multiple files, 207
overlapping, 86
raising, 87
selective processing, 208
underlining, 136
vertically stacked, 297
width calculation, 184
text companion fonts, 24, 393, 501, 506
text editor, 4, 17
text formula, 119
text height, 48, 50
\textsymbol, 463, 584
\text (AMS), 259, 583
\textbf, 65, 372, 373, 584
\textcircled, 463, 584
\textcolor (color), 167, 335, 584
texcomp package, 24-5, 393, 503
\textcompwordmark, 463, 584
\texteuro, 25
\texteuro (textcomp), 503
\textfloatsep, 172, 584
\textfraction, 172, 584
\textheight, 48, 50, 332, 361, 434, 452, 453, 584
\textit, 65, 372, 585
\textmd, 65, 372, 585
\textnormal, 65, 372, 585
\textquotedbl, 503
\textregistered, 463, 585
\texttt, 65, 372, 585
\textsf, 25, 65, 372, 585
\textsc, 44, 65, 372, 585
\textstyle, 137, 146, 264, 375, 585
\textsuperscript, 463, 585
\texttrademark, 463, 585
\texttt{65}, 68, 372, 585
\textup{65}, 372, 585
Textures, 382
textures, graphics option, 154
\textwidth, 21, 48, 50, 184, 195,
  332, 361, 452, 453, 455,
  586
\tf@size, 376
.tfm file, 234, 400, 487, 498, 503,
  506
\tfraction (\AMS), 264, 586
\textwidth, 503, 586
\th, 503, 586
Th`anh, H`an Thˆe´, 16, 238, 606
\thanks, 54, 424, 586
\the, 586
\thecounter, 183, 586
thebibliography environment,
  217, 220, 221, 224, 228,
  309, 311, 419, 523
\thechapter, 187
\theequation, 191, 277
\thefootnote, 113, 116, 190
Theiling, Henrik, 25
theindex environment, 225, 228,
  399, 523
\thenote (seminar), 335, 337
theorem environment, 80, 523
theorem package, 81, 282, 395
theorems, 80
  with \AMS, 280
\theoremstyle (\AMS), 281
\thepage, 44, 183, 335, 363, 455
\thesection, 183, 187, 277
\theslide (seminar), 335, 337
\thesubsection, 187
\Theta, 125, 586
\theta, 125, 586
\thicklines (eepic), 306
\thicklines (picture), 94, 289,
  293, 300, 497, 586
\thickspacel (\AMS), 269, 586
\thinlines (picture), 94, 289, 300,
  494, 586
\thinspace (\AMS), 269, 586
\thisfancypage (fancybox), 95,
  586
\thispagemode, 21, 43, 327, 335,
  363, 587
\thmtest.tex, 281
.tif file, 162
TIFF images, 162, 238
.tiff file, 162
\tilde, 129, 263, 587
\Tilde (\AMS), 263, 587
times package, 234
\times, 126, 587
Times-Roman, PostScript font, 234,
  379, 504
times.sty, 378
\tiny, 63, 371, 587
title
  of sections, 55
  of table, see float
title page, 39, 52
  free format, 54
  standard format, 52
\title, 52, 53, 54, 69, 419, 431, 587
titlepage environment, 52, 54,
  69, 210, 523
titlepage option, 39, 54, 55, 351
\to, 127, 587
\toaddress, 359, 363
tocdepth counter, 58
.toc file, 59, 400, 436
\today, 27, 28, 36, 255, 363, 460,
  587
\toname, 359, 360, 363
top margin, 48, 50
\top, 127, 587
\topcodeline, 173, 587
\topfraction, 172, 587
\toplink (pp4link), 349
\topmargin, 48, 50, 361, 453, 587
\totalheight, 171, 588
\topskip, 75, 149, 588, 604
\topskip, 48, 588
\toptarget (pp4link), 349
\totalheight, 86, 90, 588
\totalnumber, 172, 588
\tracefnt package, 393, 433
\transcript file, 14, 50, 399, 401
\transliteration, Cyrillic, 284
\triangle, 127, 588
\triangletext, 126, 588
\triangleright, 126, 588
  trivial lists, 79
\trivlist environment, 79, 523
INDEX  645

truetex, graphics option, 154
TrueType fonts, 238
TS1 encoding, 502, 506
\tt (EmText), 65, 367, 485
\ttdefault, 233, 372, 379, 588
\ttfamily, 64, 371, 379, 588
TUG, TeX Users Group, 3, 382, 388
Turkish, 252, 462
turn environment (rotating), 159
\turnbox (rotating), 159
two-column page formatting, 38
column dividing rule, 40
column separation, 40
for single pages, 51
two-sided page formatting, 38, 50
left margin, 48
twocolumn option, 34, 38, 40, 51, 116, 117, 351
\twocolumn, 34, 51, 395, 431, 588
twoside option, 34, 38, 43, 44, 49, 50, 58, 116, 117
\txt file, 384
typefaces, see fonts
\typein, 211, 212, 328, 589
\typeout, 211, 589
type 1 fonts, 8, 25, 231, 238, 387, 498, 503
type 3 fonts, 8, 234, 238

\u (˘ accent), 24, 589
ukhyphen.tex, 256
Ukrainian, 252
Umeki, Hideo, 49
\unboldmath, 143, 373, 433, 589
\underbrace, 136, 589
underbraces in formulas, 136
Underfull \bbox warning, 434
Underfull \bbox warning, 435
\underleftarrow (\AMS), 262, 589
\underleftrightarrow (\AMS), 262, 589
\underline, 136, 362, 458, 589
underlining, 136
\underrightarrow (\AMS), 262, 589
\underset (\AMS), 262, 589
\unitlength (picture), 288, 289, 290, 590
units of length, 21
Unix, 381
\unlhd, 126, 590
unpack.ins, 384
\unrhd, 126, 590
unsrt bibliography style, 310
unsrtnat bibliography style, 221, 311
\Uparrow, 127, 132, 590
\uparrow, 127, 132, 590
\updefault, 233, 372, 590
\Updownarrow, 127, 590
\updownarrow, 127, 590
Upper Sorbian, 252
upref package, 282
\uproot (\AMS), 269, 590
\upshape, 125, 590
\upsilon, 125, 590
Urban, Michael, 606
url package, 112
\url (url), 112, 590
\urldef (url), 112
\urlid (pdfscreen), 342
\urlstyle (url), 112, 590
\usebox, 87, 93, 106, 199, 300, 362, 591
\usecounter, 75, 77, 78, 591
\usefont, 369, 591
\usedef, 12, 24, 41, 42, 201, 222, 395, 412, 416, 418, 421, 422, 432, 441, 591
\usepostamble (DocStrip), 465
\userereamble (DocStrip), 465
\usyhyphen.tex, 256
\v (ˇ accent), 24, 591
\value, 183, 191, 193, 195, 426, 591
van Dongen, Marc, 344
van Zandt, Timothy, 94, 308, 330
\varepsilon, 125, 591
\varinjlim (\AMS), 268, 591
\varliminf (\AMS), 268, 591
\varliimsup (\AMS), 268, 591
\varkappa, 125, 591
\varkappa, 125, 591
\vartimes, 126, 590
\varprojlim (\AMS), 268, 591
\varrho, 125, 591
\varsigma, 125, 591
\vartheta, 125, 591
\vbox (\TeX), 439
\vdash, 126, 127, 592
\vdots, 126, 127, 592
\vec, 129, 144, 263, 592
\Vec (\AMS), 263, 592
\vector (picture), 161, 295, 299, 300, 416, 592
\vee, 126, 592
\verb, 110, 393, 420, 592
\verb*, 110, 592
\verbatim environment, 110, 467, 523
\verbatim input (verbatim), 111, 396
\verse, 68
\verse environment, 61, 68, 79, 523
version, math, 373
\.vf file, 234, 236, 400, 488, 504
\.vfill, 32, 592
viewgraphs, see slides
virtual fonts, 488, 503
\visible (slides), 326, 592
\vline, 97, 106, 592
\Vmatrix environment (\AMS), 266, 524
\vmatrix environment (\AMS), 266, 524
\voffset (\TeX), 593, 604
Volovich, Vladimir, 506
\vpagcolor (background), 345
\vpageref (varioref), 215, 396, 593
\vpagerefrange (varioref), 216
\vref (varioref), 215, 396, 593
\vrefrange (varioref), 216
\vspace, 32, 92, 332, 364, 419, 455, 593
\vspace*, 32, 69, 173, 593
\vspace (\TeX), 27, 460
Washington, University, 283
Web Consortium, 479
Web2c \TeX, 381
\web2ctex.txt, 384
\wedge, 126, 593
Welsh, 252
what-you-see-is-what-you-get, 4, 382
\whiledo (ifthen), 193, 195, 593
Wicks, Mark A., 162, 237
\widehat, 593
\widetilde, 594
\width, 86, 90, 594
Williams, Graham, 13
Williamson, H. A., 606
Windows, 381
Windows coding, 26, 462
\windvi previewer, 154, 234, 305
WinEdt, 15, 382
WinShell, 15, 382
Woliński, Marcin, 464
Woeding, Mark, 464
word division, see hyphenation
word spacing, see spacing
World Wide Web, 4, 8, 9, 475-82
\.wp, 127, 594
\.wr, 126, 594
\.x.tex, 394, 413
\.xdef (\TeX), 450
\.xdvi previewer, 15, 154, 234, 305
\.Xi, 125, 594
\.xi, 125, 594
\.xleftarrow (\AMS), 263, 594
XML, 3, 5, 10, 478-81
xmltex format, 481
xmltex.tex, 480
\.xr package, 215, 243, 396
\.xrightarrow (\AMS), 263, 594
XSL, 5, 479
xspace package, 186, 396
Xy-pic, 307
Y&Y, 382, 489, 506
\\year (\TeX), 27, 460
\.zeta, 125, 594
\zip program, 166
\.zip file, 166
Zlatuška, Jiří, 606