A Guide to \LaTeX

Draft 4th Edition
May 12, 2003
Trademark notices

METAFONT™ is a trademark of Addison-Wesley Publishing Company.
TEX™, AMS-TEX™, and AMS-LaTEX™ 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 techexplorer Hypermedia Browser™ 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\epsilon, 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\epsilon 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 The old version 2.09 is obsolete and should no longer play any role in such a manual.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\LaTeX{} 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 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 fancyhdr package comes in the section on page styles, 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 pdftex and 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 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 TeXLive 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 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 TeXLive CD, that world is brought even closer to their doorsteps.

Helmut Kopka and Patrick W. Daly
June, 2003
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>vii</td>
</tr>
<tr>
<td><strong>I Basics</strong></td>
<td>1</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>3</td>
</tr>
<tr>
<td>1.1 Just what is \LaTeX{}?</td>
<td>3</td>
</tr>
<tr>
<td>1.2Markup Languages</td>
<td>4</td>
</tr>
<tr>
<td>1.3 \TeX{} and its offspring</td>
<td>6</td>
</tr>
<tr>
<td>1.4 How to use this book</td>
<td>10</td>
</tr>
<tr>
<td>1.5 Basics of a \LaTeX{} file</td>
<td>11</td>
</tr>
<tr>
<td>1.6 \TeX{} processing procedure</td>
<td>14</td>
</tr>
<tr>
<td><strong>2 Text, Symbols, and Commands</strong></td>
<td>17</td>
</tr>
<tr>
<td>2.1 Command names and arguments</td>
<td>17</td>
</tr>
<tr>
<td>2.2 Environments</td>
<td>19</td>
</tr>
<tr>
<td>2.3 Declarations</td>
<td>20</td>
</tr>
<tr>
<td>2.4 Lengths</td>
<td>21</td>
</tr>
<tr>
<td>2.5 Special characters</td>
<td>22</td>
</tr>
<tr>
<td>2.6 Exercises</td>
<td>27</td>
</tr>
<tr>
<td>2.7 Fine-tuning text</td>
<td>28</td>
</tr>
<tr>
<td>2.8 Word division</td>
<td>34</td>
</tr>
<tr>
<td><strong>3 Document Layout and Organization</strong></td>
<td>37</td>
</tr>
<tr>
<td>3.1 Document class</td>
<td>37</td>
</tr>
<tr>
<td>3.2 Page style</td>
<td>42</td>
</tr>
<tr>
<td>3.3 Parts of the document</td>
<td>52</td>
</tr>
<tr>
<td>3.4 Table of contents</td>
<td>58</td>
</tr>
<tr>
<td><strong>4 Displayed Text</strong></td>
<td>61</td>
</tr>
<tr>
<td>4.1 Changing font</td>
<td>61</td>
</tr>
<tr>
<td>4.2 Centering and indenting</td>
<td>67</td>
</tr>
<tr>
<td>4.3 Lists</td>
<td>69</td>
</tr>
<tr>
<td>4.4 Generalized lists</td>
<td>74</td>
</tr>
<tr>
<td>4.5 Theorem-like declarations</td>
<td>80</td>
</tr>
</tbody>
</table>
## 10 PostScript and PDF

10.1 \TeX{} and PostScript ........................................ 231
10.2 Portable Document Format ................................. 236

## 11 Multilingual \TeX{}

11.1 The babel system .............................................. 252
11.2 Contents of the language.dat file ..................... 256

## 12 Math Extensions with AMS-L\TeX{}

12.1 Invoking \textsf{AMS-L}\TeX{} ................................. 258
12.2 Standard features of \textsf{AMS-L}\TeX{} .................. 258
12.3 Further \textsf{AMS-L}\TeX{} packages ................... 280
12.4 The \textsf{AMS} fonts ........................................ 283

## 13 Drawing with \TeX{}

13.1 The picture environment .................................... 287
13.2 Extended pictures ............................................ 302
13.3 Other drawing packages ................................... 307

## 14 Bibliographic Databases and \BibTeX{}

14.1 The \textsf{BibTeX} program ................................. 309
14.2 Creating a bibliographic database ...................... 311
14.3 Customizing bibliography styles ....................... 321

## 15 Presentation Material

15.1 Slide production with \SL\TeX{} ............................ 324
15.2 Slide production with \textsf{seminar} ...................... 330
15.3 Electronic documents for screen viewing ............... 340
15.4 Special effects with PDF .................................. 343

## 16 Letters

16.1 The \textsf{\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
H Command Summary

H.1 Brief description of the \LaTeX commands .......................... 507
H.2 Summary tables and figures ........................................... 595

Bibliography

Index

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 series attributes ....................................... 369
A.3 Attributes of the Computer Modern fonts .................... 370

D.1 Input coding schemes for \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 A_M\$ arrows ....................................................... 599
H.21 A_M\$ binary operation symbols .............................. 599
H.22 \textit{AMS} Greek and Hebrew letters .......................... 600
H.23 \textit{AMS} delimiters ............................................. 600
H.24 \textit{AMS} relational symbols .................................... 600
H.25 \textit{AMS} negated relational symbols ........................... 601
H.26 Miscellaneous \textit{AMS} symbols ................................. 601

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 \textit{list} parameters ........................................... 76
6.1 An embellished image file .......................................... 160
10.1 Output produced by pdf\TeX{} with the \texttt{hyperref} package 240
13.1 Comparison of \textit{eepic} with \textit{eepicemu} .................... 306
15.2 Title page of a pdf\textit{screen} document ....................... 341
B.1 The \textit{TeXLive} welcome ......................................... 383
B.2 The \textit{TeXLive} documentation browser .......................... 384
B.3 The TDS directory tree ............................................ 388
B.4 Partial directory tree of CTAN servers .......................... 390
E.1 Example of \TeX{}\textit{4ht} and \textit{techexplorer} output .... 482
H.1 Single column page format ........................................ 602
H.2 Double column page format ....................................... 603
H.3 Format of the \textit{list} environment ............................ 604
Part I

Basics
1

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 like \TeX is an open software system, available free of charge. Its core is maintained by the \LaTeX3 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 pf\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 \TeXLive, annually to its
members, containing a \TeX/L\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 \TeXLive CD for Windows, Macintosh, and Linux, which also contains a directory specific to this book (\texttt{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 \emph{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 \emph{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 \emph{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 \emph{source text}, which can be prepared by a simple, dumb \emph{text editor} program, is converted into typographically set output by a separate program.

For example, to code the line

He took a \textbf{bold step} forward.

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

\begin{verbatim}
He took a \textbf{bold step} forward.
\end{verbatim}

In Plain \TeX, the same sentence would be coded as:

\begin{verbatim}
He took a \textbf{bold step} forward.
\end{verbatim}
1.2. Markup Languages

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:

\[<h1>Logical Markup</h1>\]

The equivalent \LaTeX\ entry would be:

\[\text{\section{Logical Markup}}\]

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 \TeX 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 \TeX, which is a rendering in capitals of the Greek letters \text{τεχ}. For this reason the last letter is pronounced not as an \text{x}, but as the \text{ch} in Scottish \text{loch} or German \text{ach}, or as the Spanish \text{j} or Russian \text{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 \TeX, the same author has developed a further program called \textsc{Metafont} for the production of character fonts. The standard \TeX 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 \TeX 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 \TeX. This indeed has occurred, and several \TeX variants do exist, including \textsc{pdf\TeX} which we deal with later in this Chapter. Only Knuth is allowed to alter \TeX itself, which he does only to correct any obvious bugs. Otherwise, he considers \TeX 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 \TeX program

The basic \TeX 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 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 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 \TeX 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 \TeX program.
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 initex. For example, this is just what must be done to upgrade \texttt{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 \texttt{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, \texttt{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, \texttt{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 \texttt{LATEX} 2.09 became frozen.
1.3.4 \LaTeX \text{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{} 2\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 \textit{The \LaTeX{} Companion} (Goossens \textit{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);
Chapter 1. Introduction

- 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\,2\epsilon, 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 \textit{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 \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.
1.5.2 Contents of a \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 \textit{indentation}, \textit{equations}, temporary change of \textit{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

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

**tools** packages are a set written by members of the \LaTeX3 Team, and should always be in the installation;

**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;

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

**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 \TeXLive 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 \TeXLive CD at

```
\textsf{\texttt{\texmf/doc/html/catalogue/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 .dtx. A special program \texttt{DocStrip} (Section D.7.1) is used to extract the actual package file or files, while \texttt{\LaTeX}ing the original .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{\texmf/doc/latex}. However, you might have to generate the documentation output yourself by processing the .dtx file, which should be found in \texttt{\texmf/source/latex}. (Section B.3 explains the organization of the \TeX directory system.)

Some package authors write their manuals as an extra .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 \textbf{\LaTeX} 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 \textit{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 .\texttt{tex}; it is then processed by \TeX{} to produce a new file with the same base name and the extension .\texttt{dvi}, for \textit{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 \texttt{hi.tex} is

\begin{verbatim}
tex \&latex hi
\end{verbatim}

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

\begin{verbatim}
latex hi
\end{verbatim}

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

During the processing, \TeX{} writes information, warnings, error messages to the computer monitor, and to a \textit{transcript} file with the extension .\texttt{log}. It is well worth inspecting this file when unexpected results appear.
1.6. TeX processing procedure

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

\texttt{dvips hi}

to obtain \texttt{hi.ps} from \texttt{hi.dvi}. And then one sends \texttt{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

\texttt{xdvi hi}

to view \texttt{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, \BIB{}TeX, MakeIndex (these are explained later) with a mouse click.

One such editor for Windows, available on the enclosed \TeX{}Live CD in the \texttt{support} directory, is called \texttt{WinShell}, written by Ingo H. de Boer (\url{www.winshell.de}). Although free of charge, its author appreciates donations to offset his expenses.

Another such editor and \LaTeX{} interface is \texttt{WinEdit} by Aleksander Simonic (\url{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 \texttt{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 \url{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{}: pdf\TeX{}

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
modification is called pdfLATEX, created by H\`an Th\^e Th\`anh. This program
does everything T\TeX does, but it optionally writes its output directly to a
PDF file, bypassing the DVI output of regular T\TeX. It therefore combines
the T\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 T\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 pdfLATEX produces something one could call
pdfLATEX. 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
T\TeX or with pdfT\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.
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 # $ & ˜ _ ˆ % { } 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 \texttt{\textbackslash large}, \texttt{\Large} and \texttt{\LARGE} are distinct commands.

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

Some commands take \textit{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

\texttt{\textbackslash name[optional]\{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 \texttt{[ ]} and the mandatory ones into curly braces \texttt{\{} \texttt{\}}. 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 \texttt{\{} \texttt{\}} pair and their sequence must be maintained as given in the command description. For example,

\texttt{\textbackslash \texttt{\textbackslash rule}[lift]\{width\}\{height\}}

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

Some commands have a so-called \texttt{*-form} in addition to their normal appearance. A \texttt{*} is added to their name to modify their functionality somehow. For example, the \texttt{\textbackslash section} command has a \texttt{*-form} \texttt{\textbackslash section*} which, unlike the regular form, does not print an automatic section number. For each such command, the difference between the normal and \texttt{*-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 \texttt{\LaTeX} to produce the \LaTeX logo. If such a command is followed by
2.2. Environments

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 \TeX 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 \LaTeX2\epsilon logo is produced with \LaTeXe. Can you see why this logo command cannot be named \LaTeXe?

2.2 Environments

An environment is initiated with the command \begin{name} and is terminated by \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 quote environment,

\begin{quote}
previous text
\begin{quote}
text1 \small text2 \bfseries text3
\end{quote}
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 text1, text2, and text3. After text1 comes the command \small, which has the effect of setting the next text in a smaller typeface. After text2, there is an additional command \bfseries to switch to bold face type. Both these commands only remain in effect up to the \end{quote}.

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

After the end of the 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 `\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 `\bfseries` and `\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 `\setlength` which assigns a value to a *length parameter* (see Sections 2.4 and 8.2).

Examples:

```latex
{\bfseries This text appears in bold face} The `\bfseries` declaration changes the typeface: *This text appears in bold face*. The effect of this declaration ends with the closing brace `}`.
```

```latex
\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 `\setlength{\parindent}`, or at the latest with the `\end` command that terminates the current environment.
```

```latex
\pagenumbering{roman} The page numbering is to be printed in Roman numerals.
```

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:

- `\newcounter`
- `\setcounter`
- `\addtocounter`
- `\newlength`
- `\thispagestyle`
- `\newsavebox`
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:

- \texttt{cm} centimeter
- \texttt{mm} millimeter
- \texttt{in} inch (1 in = 2.54 cm)
- \texttt{pt} point (1 in = 72.27 pt)
- \texttt{bp} big point (1 in = 72 bp)
- \texttt{pc} pica (1 pc = 12 pt)
- \texttt{dd} didot point (1157 dd = 1238 pt)
- \texttt{cc} cicero (1 cc = 12 dd)
- \texttt{em} a font-specific size, the width of the capital M
- \texttt{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 \textit{period} or a \textit{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 \texttt{\setlength}, which is described in Section 8.2 along with other commands for dealing with lengths. Its syntax is:

\texttt{\setlength{\length_name}{length_spec}}

For example, the width of a line of text is specified by the parameter \texttt{\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:

\texttt{\setlength{\textwidth}{12.5cm}}

2.4.2 Rubber lengths

Some parameters expect a \textit{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 interpreted as commands. To print them as text, one must give a command consisting of \ plus that character.

\[ $ = \$ \quad & = \& \quad % = \% \quad # = \# \quad _ = \_ \quad \{ = \{ \quad \} = \} \]

### 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:

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

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:

\begin{itemize}
  \item \oe = \{\oe\}
  \item \OE = \{\OE\}
  \item \aa = \{\aa\}
  \item \AA = \{\AA\}
  \item \= \{\=\}
  \item \o = \{\o\}
  \item \O = \{\O\}
  \item \l = \{\l\}
  \item \L = \{\L\}
  \item \ss = \{\ss\}
  \item \SS = \{\SS\}
  \item \= \{\=\}
\end{itemize}

\(\text{Angström}\) 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:

\begin{itemize}
  \item \'o = \'{o}
  \item \'{o} = \'{o}
  \item \^{o} = \^{o}
  \item \~{o} = \~{o}
  \item \={o} = \={o}
  \item \.{o} = \.{o}
  \item \u{o} = \u{o}
  \item \v{o} = \v{o}
  \item \H{o} = \H{o}
  \item \t{oo} = \t{oo}
  \item \c{o} = \c{o}
  \item \d{o} = \d{o}
  \item \b{o} = \b{o}
  \item \r{o} = \r{o}
\end{itemize}

The \o above is given merely as an example: any letter may be used. With \i and \j it should be pointed out that the dot must first be removed. This is carried out by prefixing these letters with a backslash: the commands \i and \j yield \ı and \ĳ. In this way \ı and \ĳ are formed by typing \u{\i} and \H{\j}.

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

\begin{itemize}
  \item \= \{\=\}
  \item \= \{\=\}
  \item \= \{\=\}
  \item \= \{\=\}
  \item \= \{\=\}
  \item \= \{\=\}
\end{itemize}

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

2.5.8 The euro symbol

The euro symbol \(\euro\) (or \(\euro\)) 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 \textit{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.

\textbf{Package: textcomp}

The package textcomp must be loaded in the preamble with

\begin{verbatim}
\usepackage{textcomp}
\end{verbatim}
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><code>\texteuro</code></td>
<td>Text Companion</td>
<td>Non standard symbol</td>
</tr>
<tr>
<td>eurosym</td>
<td><code>\euro</code></td>
<td>Eurosym</td>
<td>Sans serif, variable</td>
</tr>
<tr>
<td>europs</td>
<td><code>\EUR</code></td>
<td>PostScript</td>
<td>Varies with font family</td>
</tr>
<tr>
<td></td>
<td><code>\EURofc</code></td>
<td></td>
<td>Invariable, official</td>
</tr>
<tr>
<td>eurosans</td>
<td><code>\euro</code></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 Gauå meets AmpŠre and on a Macintosh as Gau· meets Ampére. 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. \( \LaTeX \) processes the letter combinations ff, fi, fl, ffi, and ffl not as

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

but rather 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{/} between the letters. This is sometimes desired for such words as \texttt{shelfful} (shelf\texttt{/}ful), which looks rather strange when printed with the normal \texttt{ff} ligature, \texttt{shelfful}.
2.6. Exercises

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 \TeX 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 \TeX 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 \pounds 1 = \$1.58, an increase of 1\% over yesterday.
\end{document}
```

Process this source file with \TeX by clicking the appropriate icon, or by issuing `latex exer` in a command window. If the processing occurs without any error messages, the `.dvi` file `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 \TeX 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 ‘.’ or exclamation mark ‘!’, there is more space than after a comma ‘,’. 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 *i. e.*, *Prof. Jones*, or *Phys. Rev.*, where the normal spacing is required. This can be achieved by using the characters `˜` or `\_` instead of the normal blank. (The character `\_` is
simply a symbol for the blank which is otherwise invisible.) Both these methods insert the normal interword spacing; in addition, \textbackslash ~ is a \emph{protected space} that prevents the line from being broken at this point. The above examples should be typed in as \textit{i.\textbackslash-}e., \textit{Prof.}\~\textit{Jones}, and \textit{Phys.}\~\textit{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 \textbackslash@ 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 \textbackslash@ 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 \texttt{“ and ”} from the corresponding single quotes \texttt{‘ and ’} when they appear together. For example, the text \texttt{“‘\,.‘Beginning’ and ‘End’’”} produces \texttt{“‘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 \texttt{1.5cm} or \texttt{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 *) 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 *-form, on the other hand, inserts the spacing no matter where it occurs.

A blank before or after the command will also be included:
This is \hspace{1cm} 1cm
This is \hspace{1cm} 1cm
This is \hspace{1cm} 1cm
This is \hspace{1cm} 1cm
This is \hspace{1cm} 1cm

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 \textit{electron volt}, abbreviated ‘eV’, which looks much better if the two letters are nearer together, as ‘eV’, with \texttt{e\hspace{-0.12em}V}.

The command \texttt{\hfill} is an abbreviation for \texttt{\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

Left \hfill Right

Multiple occurrences of \texttt{\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

Left \hfill Center \hfill Right

If \texttt{\hfill} comes at the beginning of a line, the spacing is suppressed in accordance with the behavior of the standard form for \texttt{\hspace}. If a rubber space is really to be added at the beginning or end of a line, \texttt{\hspace*{\hfill}} must be used instead. However, L\TeX 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{itemize}
  \item \texttt{\quad} and \texttt{\qquad}
\end{itemize}

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

\section*{Inserting variable \ldots and \ldots sequences}

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

\begin{itemize}
  \item \texttt{\dotfill} and \texttt{\hrulefill}
\end{itemize}

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

\begin{itemize}
  \item \texttt{Start \dotfill\ Finish\\ and}
  \item \texttt{Left \hrulefill\ Center \hrulefill\ Right\\ produce}
\end{itemize}

Start .......................................................... Finish
Left ......................................................... Center ........................................ Right
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` 8:30 `\hfill` Arrival `\hrulefill` 11:45

Departure \ldots 8:30 \hfill Arrival 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 command `\`

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

\`
\[space\]
\`

The optional argument `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 `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 `\linebreak` is used to encourage or force a line break at a certain point in the text. Its form is

`\linebreak[num]`
where \textit{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 \textit{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 \ or \newline, however, the line is filled with empty space after the last word and the interword spacing remains normal.

The opposite command

\noindent\linebreak[\textit{num}]

discourages a line break at the given position, with \textit{num} specifying the degree of discouragement. Again, \noindent\linebreak without a \textit{num} argument has the same effect as \noindent\linebreak[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 \textbf{Vertical spacing}

It is possible to add extra vertical spacing of amount \textit{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 \textit{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
2.7. Fine-tuning text

\bigskip \medskip \smallskip
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
\pagebreak[num]
\nopagebreak[num]
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
\newpage
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

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

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

\[
\text{Boris Be\textbackslash discretionary\{k-\}\{k\}\{ck\}er}
\]

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

**Note:** In today's new German spelling, *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 \textit{a–z}, with no special characters or accents. However, if the \texttt{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 \texttt{\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 \texttt{\sloppy} is in effect, it is possible to undo it temporarily and to turn hyphenation back on with the command \texttt{\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 \texttt{\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 \texttt{\today}. This topic is treated in more detail in Chapter 11.
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:

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

The default is `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 portrait mode. With the option `landscape` the shorter dimension becomes the vertical one, the 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

- `onecolumn`
- `twocolumn`

The default is `onecolumn`. In the case of the `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

- `oneside`
- `twoside`

With `oneside`, all pages are printed the same; however, with `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:

**1eqno**  
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 columnsep} 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{\textbackslash setlength}. For example, to change \texttt{\textbackslash mathindent} to 2.5 cm, give

\begin{verbatim}
\setlength{\mathindent}{2.5cm}
\end{verbatim}

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.

\textit{Note:} if there are some improper word divisions, you call tell \LaTeX{} where the correct division should occur with the command \textbackslash -, for example, 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}, \LaTeX{} was not able to break the lines cleanly. In the output, these lines will extend beyond the right margin. The usual cause is that \LaTeX{} was not able to divide some word, either because it is indivisible or because \LaTeX{}'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
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

\begin{verbatim}
\usepackage{package}
\end{verbatim}

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

\begin{verbatim}
\usepackage{makeidx,ifthen}
\end{verbatim}

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:

\begin{verbatim}
\usepackage[opt1,opt2,...]{package1,package2,...}
\end{verbatim}

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></td>
<td></td>
<td>Section</td>
</tr>
<tr>
<td>article</td>
<td>one-sided</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>two-sided</td>
<td>Section</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Subsection</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

Package: fancyhdr

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
Chapter 3. Document Layout and Organization

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]{...}

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 \rightmark 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}
3.2. Page style

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

\begin{verbatim}
\usepackage[parfill]{parskip}
\end{verbatim}

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

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

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

\textbf{Exercise 3.4:} Add the following to the preamble of your exercise file:

\begin{verbatim}
\usepackage{parskip}
\renewcommand{\baselinestretch}{1.2}
\end{verbatim}

After processing this exercise, repeat it with another value for the parameter \texttt{\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 \texttt{\setlength} (Section 8.2). For example, give

\begin{verbatim}
\setlength{\textwidth}{12.5cm}
\end{verbatim}

to make the text line width to be 12.5 cm.

There is also a parameter \texttt{\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 \LaTeX 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 \LaTeX 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:

\begin{verbatim}
  \setlength{\textwidth}{13cm} \setlength{\textheight}{20.5cm}
\end{verbatim}

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

**Package:** `geometry`

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

```latex
\usepackage{geometry}
\geometry{a4paper,textwidth=15cm,textheight=25cm}
```

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

```latex
\geometry{letterpaper,margin=1in}
```

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

```latex
\usepackage[a4paper,left=3cm,right=2cm]{geometry}
```

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=\{pwidth,pheight\}.

- 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=\textwidth, or with body=\{width,height\}. The margins are then set to center. Here height means total text height 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=h and vscale=v, or scale=\{h,v\}. With scale=s, both h and v are set to s. For example, \geometry[\texttt{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 \textit{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: \texttt{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{\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{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 \texttt{\premulticols} and \texttt{\postmulticols}: if the remaining space on the current page is less than \texttt{\premulticols}, a new page is started before switching to multicolumns. Similarly, if at the end of the environment, there is less than \texttt{\postmulticols} on the page, a page break is inserted before continuing. The standard values of these lengths may be altered by the user with \texttt{\setlength}, or in the case of \texttt{\premulticols} may be overridden by the second optional argument \texttt{pre_space}.

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

There is also a starred version \texttt{\begin{multicols*}{num_cols}}... \texttt{\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{verbatim}
\begin{titlepage} Title page text \end{titlepage}
\end{verbatim}

or with the commands

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

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
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\nInstitute1\nAddress1
\and Author2\nInstitute2\nAddress2}
```

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 environ-
ment, 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 Exer-
cises 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}:

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

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

\documentclass[titlepage]{article}

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
n is the current chapter number. At this point in the present book, we are in Chapter 3, Section 3.3, Subsection 3.3.3.

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

\section ... \subsection ... \subsubsection* ...

numbers the headings for \section and \subsection while leaving the headings for \subsubsection without any numbering.

The sectioning command \part is a special case and does not affect the numbering of the other commands.

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

\label{name} \ref{name}

the first of which assigns a keyword name to the section number, while the second may be used as reference in the text for printing that number. The keyword name may be any combination of letters, numbers, or symbols. For example, in this book the command \label{sec:xref} has been typed in at the start of Section 9.2.1, so that this sentence contains the input text at the start of Section \ref{sec:xref}.

A second referencing command is \pageref for printing the page number where the corresponding \label is defined.

The referencing commands may be used in many other situations for labeling items that are numbered automatically, such as figures, tables, equations.

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

In order to extend (or reduce) the level of the section numbering, it is necessary to change the value of secnumdepth. This is done with the command

\setcounter{secnumdepth}{num}
3.3. Parts of the document

(The command \setcounter is explained in Section 8.1.) In article, num may take on values from 0 to 5, and in book and report from −1 to 5.

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

\setcounter{sec_name}{num}

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

\setcounter{chapter}{2}

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

### 3.3.4 Appendix

An appendix is introduced with the declaration

\appendix

It has the effect of resetting the section counter for article and the chapter counter for book and 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

\frontmatter
  \preface, table of contents
\mainmatter
  \textit{main body of text}
\backmatter
  \textit{bibliography, index, colophon}

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

**Exercise 3.9:** Insert at the beginning of your exercise text the command \section{Title A} and at some appropriate place near the middle \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:

\pagestyle{myheadings} \markright{Exercises} \pagenumbering{Roman}

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[twoside]{report} and call the page style command \pagestyle{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

\chapter[Short form]{Chapter title}

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

\setcounter{tocdepth}{num}

The value \textit{num} has exactly the same meaning and effect as it does for the counter 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 \subsection for book and report and to level \subsubsection for article.

#### 3.4.2 Printing the table of contents

The table of contents is generated and printed with the command

\tableofcontents
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 \texttt{\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}{section}{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 \texttt{sec.name}, which is the same as one of the sectioning commands without the \texttt{\} character (for example, \texttt{section}). The \texttt{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 \texttt{\addtocontents} command puts any desired command or text into the .toc file. This could be a formatting command, say \texttt{\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 \quad \text{reads and/or produces file .lof}
\listoftables \quad \text{reads and/or produces file .lot}
\end{verbatim}
Chapter 3. Document Layout and Organization

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

\begin{verbatim}
\pagenumbering{roman}
\tableofcontents \newpage
\pagenumbering{arabic}
\end{verbatim}

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 \texttt{10pt} (default), \texttt{11pt}, and \texttt{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.
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 L\TeX{} with the command \texttt{\emph} 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]} 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], second, and \emph[third font switch]}
The \{\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 L\TeX{} 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 \{\ldots\}, 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 larger 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:

\begin{verbatim}
\renewcommand{\baselinestretch}{factor}
\end{verbatim}

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

\begin{verbatim}
\small\texttt{\normalsize}
\end{verbatim}

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 \LaTeXe, 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

\begin{verbatim}
\rmfamily to switch (back) to a Roman font;
\ttfamily to switch to a typewriter font;
\sffamily to select a sans serif font.
\end{verbatim}

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

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

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

\begin{verbatim}
\mdseries to switch (back) to medium weight;
\bfseries to select a bold face font.
\end{verbatim}

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 {...}, such as \{\scshape Romeo and Juliet\} producing ROMEO AND JULIET. For longer sections of text, an environment is preferable:

\begin{verbatim}
\begin{fontstyle} ... text in new font ... \end{fontstyle}
\end{verbatim}

This keeps better track of the beginning and end of the switch-over. For font_style, any of the above font commands may be used, leaving off the initial \ 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 $\textbf{series}$, and then a
slanted shape with $\textit{shape}$, we obtain a bold, slanted font.

\begin{verbatim}
normal and \textbf{series} bold and
\textit{shape} slanted} and back} again.
\end{verbatim}

produces: normal and **bold and slanted** and back again.

Finally, the declaration \texttt{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.

### 4.1.4 Font commands

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

\begin{verbatim}
Family: \texttt{text} \texttt{texttt} \texttt{textsf}
Shape: \texttt{textup} \texttt{textit} \texttt{textsl}
Series: \texttt{textmd} \texttt{textbf}
Default: \texttt{textnormal}
Emphasis: \texttt{em}
\end{verbatim}

Note that the \texttt{em} command is included here, corresponding to the
declaration \texttt{em}. 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{series} bold and \textit{shape} slanted} and back} again.
\end{verbatim}

to make: normal and **bold and slanted** and back again.

As for the \texttt{em} 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{bf} and \texttt{tt}, 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.

### 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 \_ 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 \ \ ... \ line n \end{center}

centers the sections of text that are separated by the \ 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 \ 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 \text{} command.

4.2.2 One-sided justification

The environments

\begin{flushleft} line 1 \ \ line 2 \ \ ... \ line 2 \end{flushleft}
\begin{flushright} line 1 \ \ line 2 \ \ ... \ 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 \[^{len}\] 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[^{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 \textit{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 enumerate

1. The labels consist of sequential numbers.
2. The numbering starts at 1 with every call to the 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 description

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

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

\textbf{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[purpose] This environment is appropriate when a number of words or expressions are to be defined.
\item[example] A keyword is used as the label and the entry contains a clarification or explanation.
\item[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{itemize}
    \item Every list should contain at least two points.
  \end{enumerate}
  \item Blank lines ahead of an \verb+\item+ command ...
\end{itemize}

\textbf{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.

\textbf{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.

\textit{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.
Exercise 4.6: Change the standard labels for the itemize environment into a long dash — (written ---) 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 enumerate environment for the first level to (I), (II), ..., 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:, ..., II–1:, II–2:, ....

An alternative method of customizing the enumeration labels is with the enumerate package in the tools collection (Section B.5.4). Once this package has been loaded, the 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 {}.

For example,
\begin{enumerate}
\item Witness tells the truth
\item Witness is lying
\end{enumerate}
\Rightarrow
Case A Witness tells the truth
Case B Witness is lying

4.4 Generalized lists

Lists such as those in the three environments itemize, enumerate, and 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 list environment:
\begin{list}{\stnd_lbl}{\list_decl} item_list \end{list}

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

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

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

4.4.1 Standard label

The first argument in the list environment defines the stnd_lbl, that is, the label that is produced by the \item command when it appears without an argument. In the case of an unchanging label, such as for the itemize environment, this is simply the desired symbol. If this is to be a mathematical symbol, it must be given as $\text{symbol\_name}$, enclosed in $\text{signs}$. For example, to select $\Rightarrow$ as the label, stnd_lbl must be defined to be $\Rightarrow$.
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 \textit{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{stnd\_lbl} could be any of the commands for printing counters described in Section 4.3.5: for example, \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{stnd\_lbl} 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 \item command. The user can redefine \makelabel with the aid of the \renewcommand in the list declaration:

\begin{verbatim}
\renewcommand{\makelabel}{new definition}
\end{verbatim}

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 \item entry. It may be redefined globally, but only for the first and second levels.
Figure 4.1: The list parameters

\parsep is the vertical spacing between paragraphs of a single \item. Its default value is reset at each level, as for \topsep.

\itemsep is the vertical spacing in addition to \parsep that is inserted between two \item entries. As for \topsep and \parsep, 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
4.4. Generalized lists

normally set to 0 pt so that no indentation occurs. This can only be changed in \texttt{list\_decl}.

\begin{itemize}
\item[\texttt{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.
\item[\texttt{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.
\item[\texttt{itemindent}] is the distance by which the label and the first line of text in an \texttt{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}.
\end{itemize}

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

4.4.3 Example of a user's list

List of Figures:

\begin{itemize}
\item[Figure 1:] Page format with head, body, and foot, showing the meaning of the various elements involved.
\item[Figure 2:] Format of a general list showing its elements.
\item[Figure 3:] A demonstration of some of the possibilities for drawing pictures with \TeX.
\end{itemize}

This list was produced with the following input:

\begin{verbatim}
\newcounter{fig}
\begin{list}{\bfseries\upshape Figure \arabic{fig}:}
  {\usecounter{fig}
   \setlength{\labelwidth}{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} \slshape}
  \item Page format with head, body, and foot, showing the meaning of the various elements involved.
  \item Format of a general list showing its elements.
  \item A demonstration of some of the possibilities for drawing pictures with \LaTeX.
\end{list}
\end{verbatim}
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 stdlbl 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:

\newenvironment{figlist}{\begin{list} {\bfseries\upshape Figure \arabic{fig}:} {\usecounter{fig} ... {0ex plus0.2ex}\slshape}}{\end{list}}

It can then be called with

\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 20mm, and the distance between the label box and the item text is to be 2mm, with a total left margin of 22mm. The right edge of the text is to be moved in 5mm from that of the enclosing text. The extra vertical spacing between two items is to be 1ex plus0.5ex minus0.4ex 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} \centering \item[] \end{trivlist}
\end{verbatim}

in which the arguments stdlbl and 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 \parsep to \parskip.

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

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

The environments flushleft and flushright are similarly defined.

4.4.6 Nested lists

Lists can be nested within one another with the list environments itemize, enumerate, and description, to a maximum depth of six. At each level, the new left margin is indented by the amount \leftmargin relative to that 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 `\leftmargin n`, where \(n\) stands for \(i, ii, iii, iv, v,\) or \(vi\). These values can be changed by the user; for example, by declaring `\setlength{\leftmarginiv}{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 list.decl.

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 `\leftmargin n`, 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

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

or

**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 *Definition, Corollary, Declaration, Lemma*, instead of *Theorem* or *Axiom*. What they have in common is that a keyword and a running number are printed in **bold face** and the corresponding text in *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

```latex
\newtheorem{struct_type}{struct_title}[in_counter]
```

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

The predefined structures are called with the command

```latex
\begin{struct_type}[extra_title] text \end{struct_type}
```
which also increments the necessary counter and generates the proper number. The above examples were produced with
\begin{verbatim}
\newtheorem{theorem}{Theorem} \newtheorem{axiom}{Axiom}[chapter]
...
\begin{theorem}[Balzano--Weierstrass] Every .... \end{theorem}
\begin{axiom} The natural numbers form ........ \end{axiom}
\end{verbatim}

The optional *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{verbatim}
\newtheorem{struct_type}[num_like]{struct_name}
\end{verbatim}

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

For more powerful theorem tools, see the *AMS* `amsthm` package (Section 12.3.1) and the `theorem` package in the tools collection (Section B.5.4).

## 4.6 Tabulator stops

### 4.6.1 Basics

On a typewriter it is possible to set *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 *LaTeX* with the `tabbing` environment:
\begin{verbatim}
\begin{tabbing}
lines \end{tabbing}
\end{verbatim}

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 zeroth tab stop. The stops can be set at any spot within a line with the command `\=`, and a line is terminated by the `\\` command:

Here is the `\=first tab stop, followed by` `\\` sets the first tab stop after the blank following the word *the*, and the second immediately after the word *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 `\>`.

A new line is started with the usual `\\` command.
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:

<table>
<thead>
<tr>
<th>Left border</th>
<th>1st tab stop</th>
<th>2nd tab stop</th>
<th>3rd tab stop</th>
</tr>
</thead>
<tbody>
<tr>
<td>3cm</td>
<td>sample column</td>
<td>4cm</td>
<td></td>
</tr>
</tbody>
</table>

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. Tabulator stops

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
\> Middle col
\= Extra 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 \texttext", ' 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', and \a' instead. For example, to produce ´o, ø, or ø inside a \tabbing environment, one must give \a=o, \a'o, or \a=a. 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

4.6.5 Remarks on tabbing

TeX treats the tabbing environment like a normal paragraph, breaking a page if necessary between two lines within the environment. However, the commands \newpage and \clearpage are not allowed within it, and the command \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, \[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 \> 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 \ command. The text could extend beyond the right margin of the page. The user must take care that this does not happen.

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

Exercise 4.9: Generate the following table with the tabbing environment.

<table>
<thead>
<tr>
<th>Year</th>
<th>Project Total</th>
<th>Deficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>2003</td>
<td>$900,000.00</td>
<td>$100,000.00</td>
</tr>
<tr>
<td>2004</td>
<td>$450,000.00</td>
<td>$150,000.00</td>
</tr>
<tr>
<td>2005</td>
<td>$350,000.00</td>
<td>$150,000.00</td>
</tr>
</tbody>
</table>
Hint: the first line in the tabbing environment should read

Project: \=Total Requirements\= = \$900,000.00 \+\\

What is the effect of the \+ command at the end of this line? How do you arrange, using these tab stops, for the years 2003, 2004, and 2005 in the second to fourth lines all to be positioned before the second tab stop? Which command should be at the end of the second line just before the \\terminator?

Lines 1–4 and 8–12 all use the same set of tab stops, even though there are additional stops set in the eighth line. With \$1\=00,000.00 one can align the entry \$\>50,000.00 in the ninth line to match the decimal places of the lines above.

Lines 5–7 have their own tab stops. Use the save and recall feature to store the preset tab stops and to bring them back. The left border of lines 5–7 correspond to the first stop of the first group. What command is at the end of the fourth line to ensure that the left border is reset to one stop earlier? How is the left border of the second-to-last line reset?

The last line contains 'signed: H. André' right justified. With what tabbing command was this produced? Watch out for the accent è in this entry within the tabbing environment!

4.7 Boxes

A box is a piece of text that T\(\hat{\text{e}}\)X 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, \(\hat{\text{T}}\)\(\hat{\text{e}}\)X cannot break it up again, even 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 \(\hat{\text{T}}\)\(\hat{\text{e}}\)X 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.

\(\hat{\text{T}}\)\(\hat{\text{e}}\)X 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{text} is also framed.

With the two commands at the right, the width is predetermined by
the optional length argument width. The other optional argument pos
specifies how the text is positioned within the box. With no value given,
the text is centered. Otherwise pos 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 ...... 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 pos. 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][1]{/}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):

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

\framebox[6\textwidth]{Text} \hfill 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}{\textwidth}{\textwidth}{\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 \texttt{high} and \texttt{low}, and back again.

The values for \texttt{height} and \texttt{depth} can be totally different from the actual ones of the \texttt{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 \texttt{height} to be the regular character size, the raised box will overprint the line above, and similarly for \texttt{depth} when a box is lowered.

### 4.7.3 Parboxes and minipages

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

\begin{verbatim}
\parbox[pos]{width}{text}
\end{verbatim}

or with the environment

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

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

The optional positioning argument \texttt{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:

\begin{verbatim}
\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.}
\end{verbatim}
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 `\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}

The minipage environment creates a vertical box like the parbox command. The bottom line of this minipage is aligned with the middle of this narrow parbox, which in turn is aligned with 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.

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

The `\parbox` command produces a vertical box containing the text just like the `\minipage` environment. However, the latter is more general. The text in a `\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 `\minipage` environment. That is, a `\minipage` can include centered or indented text, as well as lists and tabbings.

4.7.4 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{minipage}[b]{..}
parbox[t]{..} \hfill parbox[t]{..}
\end{minipage}
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.

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

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 first line.

\mbox{} \mbox{} \mbox{} \mbox{} \\[-\baselineskip\] \mbox{aligns with the very top, or \mbox{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 Paragraph boxes of specific height**

The complete syntax of the \parbox command and minipage environment includes two more optional arguments:

\parbox*[pos][height][inner_pos]{width}{text}
\begin{minipage}[pos][height][inner_pos]{width}
\text
\end{minipage}

In both cases, \textbf{height} is a length specifying the height of the box; the parameters \textbf{height}, \textbf{width}, \textbf{depth}, and \textbf{totalheight} may be employed within the \textbf{height} argument in the same way as in the \textbf{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 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}.

\subsection*{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[\texttt{lift}]{\texttt{width}}{\texttt{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{\textwidth}{\extrarule}. Without the optional argument \texttt{lift}, the rectangle is set on the baseline of the current line of text.
Chapter 4. Displayed 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 \textbf{Text}. In order to print \textbf{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.

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: fontsize 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.

Note: the problems of correctly aligning two side-by-side boxes as a pair on a line of text (Section 4.7.4) arises here once more. It will be necessary to add a dummy line to get the vertical alignment correct.

Exercise 4.11: Produce the framed structure shown below and store it with the command \sbox{\warning}{\text}. 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}{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

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:

\begin{itemize}
  \item \shadowbox{text}
    \begin{itemize}
      \item 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.
    \end{itemize}
  \item \doublebox{text}
    \begin{itemize}
      \item 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.
    \end{itemize}
  \item \ovalbox{text}
    \begin{itemize}
      \item 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.
    \end{itemize}
  \item \ovalbox{text}
    \begin{itemize}
      \item The frame thickness is set by \thicklines, but otherwise is the same as \ovalbox.
    \end{itemize}
\end{itemize}
This package also allows all pages to be boxed as part of the page style. This is done by issuing \fancypage{cmds1}{cmds2}, where cmds1 and cmds2 are commands setting box parameters, terminated by one of the box commands \fbox, \shadowbox, and so on. The first set, 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 tabular, tabular*, and array are the basic tools with which tables and matrices can be constructed. The syntax for these environments is

\begin{array}sim{pos}\{cols\}\{rows\}\end{array}
\begin{tabular}sim{pos}\{cols\}\{rows\}\end{tabular}
\begin{tabular*}sim{width}\{pos}\{cols\}\{rows\}\end{tabular*}

The 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 tabular environment. All three environments actually create a minipage. The meaning of the arguments is as follows:

pos Vertical positioning argument (see also the explanation of this argument for parboxes in Section 4.7.3). It can take on the values
t the top line of the table is aligned with the baseline of the current external line of text;
b the bottom line of the table is aligned with the external baseline;
with no positioning argument given, the table is centered on the external baseline.
width This argument applies only to the \texttt{tabular*} environment and determines its overall width. In this case, the \texttt{cols} argument must contain the \texttt{@-expression} (see below) \texttt{@{\extracolsep\fill}} somewhere after the first entry. For the other two environments, the total width is fixed by the textual content.

\texttt{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 \textit{column formatting symbols} are

\begin{itemize}
\item[l] the column contents are \textit{left} justified;
\item[r] the column contents are \textit{right} justified;
\item[c] the column contents are \textit{centered};
\item[p\{wth\}] the text in this column is set into lines of width \textit{wth}, and the top line is aligned with the other columns. In fact, the text is set in a \texttt{parbox} with the command \texttt{\parbox[t]{wth}{column text}};
\end{itemize}

\texttt{*\{num\}\{cols\}} the \textit{column format} contained in \texttt{cols} is reproduced \texttt{num} times, so that \texttt{*\{5\}\{c\}} is the same as \texttt{|c|c|c|c|c|}.

The available \textit{formatting symbols} for the left and right borders and for the intercolumn spacing are

\begin{itemize}
\item[\texttt{\{}] draws a vertical line;
\item[\texttt{||}] draws two vertical lines next to each other;
\end{itemize}

\texttt{@\{text\}} this entry is referred to as an \textit{@-expression}, and inserts \textit{text} in every line of the table between the two columns where it appears.

An \texttt{@-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 \texttt{\hspace{}} within the \texttt{text} of the \texttt{@-expression}. If the intercolumn spacing between two particular columns is to be something other than the standard, this may be easily achieved by placing \texttt{@\{\hspace{wth}\}} between the appropriate columns in the formatting argument. This replaces the standard intercolumn spacing with the width \textit{wth}.

An \texttt{\extracolsep\{wth\}} within an \texttt{@-expression} will put extra spacing of amount \textit{wth} between all the following columns, until countermanded by another \texttt{\extracolsep} command. In contrast to the standard spacing, this additional spacing is not removed by later \texttt{@-expressions}. In the \texttt{tabular*} environment, there must be a command \texttt{@{\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 \LaTeX 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.
If a p-type column contains \raggedright or \centering, the \ forces a new line within the column entry and not the end of the whole row. If this occurs in the last column, then \ cannot be used to terminate the row; instead one must use \tabularnewline to end such a row.

Since a table is a vertical box of the same sort as parbox and minipage, it may be positioned horizontally with other boxes or text (see examples in Section 4.7.3). In particular, the table must be enclosed within

\begin{center} table \end{center}

in order to center it on the page.

### 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 \texttt{tabular} environment itself.

- \texttt{\tabcolsep} is half the width of the spacing that is inserted between columns in the \texttt{tabular} and \texttt{tabular*} environments;
- \texttt{\arraycolsep} is the corresponding half intercolumn spacing for the \texttt{array} environment;
- \texttt{\arrayrulewidth} is the thickness of the vertical and horizontal lines within a table;
- \texttt{\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 \setlength{\arrayrulewidth}{0.5mm}. 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.
<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>3</td>
<td>Clarkson Chargers</td>
<td>33</td>
<td>17</td>
<td>7</td>
<td>9</td>
<td>70:44</td>
<td>41:25</td>
</tr>
<tr>
<td>4</td>
<td>Daysdon Bombers</td>
<td>33</td>
<td>14</td>
<td>10</td>
<td>9</td>
<td>66:50</td>
<td>38:28</td>
</tr>
<tr>
<td>5</td>
<td>Edgartown Devils</td>
<td>33</td>
<td>16</td>
<td>6</td>
<td>11</td>
<td>63:53</td>
<td>38:28</td>
</tr>
<tr>
<td>6</td>
<td>Freeburg Fighters</td>
<td>33</td>
<td>15</td>
<td>7</td>
<td>11</td>
<td>64:47</td>
<td>37:29</td>
</tr>
<tr>
<td>7</td>
<td>Gadsby Tigers</td>
<td>33</td>
<td>15</td>
<td>7</td>
<td>11</td>
<td>52:37</td>
<td>37:29</td>
</tr>
<tr>
<td>8</td>
<td>Harrisville Hotshots</td>
<td>33</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>62:58</td>
<td>35:31</td>
</tr>
<tr>
<td>9</td>
<td>Idleton Shovers</td>
<td>33</td>
<td>13</td>
<td>9</td>
<td>11</td>
<td>49:51</td>
<td>35:31</td>
</tr>
<tr>
<td>10</td>
<td>Jamestown Hornets</td>
<td>33</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>48:47</td>
<td>33:33</td>
</tr>
<tr>
<td>11</td>
<td>Kingston Cowboys</td>
<td>33</td>
<td>13</td>
<td>6</td>
<td>14</td>
<td>54:45</td>
<td>32:34</td>
</tr>
<tr>
<td>12</td>
<td>Lonsdale Stompers</td>
<td>33</td>
<td>12</td>
<td>8</td>
<td>13</td>
<td>50:57</td>
<td>32:34</td>
</tr>
<tr>
<td>13</td>
<td>Marsdon Heroes</td>
<td>33</td>
<td>9</td>
<td>13</td>
<td>11</td>
<td>50:42</td>
<td>31:35</td>
</tr>
<tr>
<td>14</td>
<td>Norburg Flames</td>
<td>33</td>
<td>10</td>
<td>8</td>
<td>15</td>
<td>50:68</td>
<td>28:38</td>
</tr>
<tr>
<td>15</td>
<td>Ollison Champions</td>
<td>33</td>
<td>8</td>
<td>9</td>
<td>16</td>
<td>42:49</td>
<td>25:41</td>
</tr>
<tr>
<td>16</td>
<td>Petersville Lancers</td>
<td>33</td>
<td>6</td>
<td>8</td>
<td>19</td>
<td>31:77</td>
<td>20:46</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 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 \[0.5ex] 
2 & Borden Comets & 33 & 18 & 9  & 6  & 65:37 & 45:21 \[0.5ex] 
... & ..... & .. & .. & .. & .. & ... & ... \[0.5ex] 
17 & Quincy Giants & 33 & 7  & 5  & 21 & 40:89 & 19:47 \[0.5ex] 
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|l|c|rrr|c|c}
Position & Club & Games & W  & T  & L  & Goals & Points \\
1 & Amesville Rockets & 33 & 19 & 13 & 1  & 66:31 & 51:15 \[0.5ex] 
2 & Borden Comets & 33 & 18 & 9  & 6  & 65:37 & 45:21 \[0.5ex] 
... & ..... & .. & .. & .. & .. & ... & ... \[0.5ex] 
17 & Quincy Giants & 33 & 7  & 5  & 21 & 40:89 & 19:47 \[0.5ex] 
18 & Ralston Regulars & 33 & 3  & 11 & 19 & 37:74 & 17:49
\end{tabular}
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
| 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  |
\hline
\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

\{r|l|c|rrr|c|c|\} \ or \ {l||@{ 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} or
{|r|l||@{ 33 }|rrr|r@{:}l|r@{:}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

\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
column, use the rule that a column ‘owns’ everything up to but excluding the next \( r \), \( l \), or \( c \).

The table of final results for our soccer league 2002/03 is to have the following title:

\[
\begin{tabular}{|r|l||rrr|r|@|:|l|@|:|l||c|}
\hline
\multicolumn{10}{|c|}{\bfseries 1st Regional Soccer League --- Final Results 2002/03}\hline
\end{tabular}
\]

The horizontal lines for positions 3–5, 7–14, and 17 were made with the command \texttt{\cline{1-9}} while all the others used \texttt{\hline}:

\begin{verbatim}
11 & Kingston Cowboys & 13 & 6 & 14 & 54&45 & 32&34 & Medium Teams \cline{1-9}
\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}
18 & Ralston Regulars & 3 & 11 & 19 & 37&74 & 17&49 & \raisebox{1.5ex}[0pt]{Demoted}\hline
\end{verbatim}
The \texttt{\raisebox} command lifts the text ‘Demoted’ by 1.5 ex. If the optional argument \texttt{[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{center}
\begin{tabular}{|c|c|c|c|c|c|}
\hline
\textbf{Club} & \textbf{W} & \textbf{T} & \textbf{L} & \textbf{Goals} & \textbf{Points} & \textbf{Remarks} \\
\hline
\end{tabular}
\end{center}

\begin{center}
\textbf{1st Regional Soccer League — Final Results 2002/03}
\end{center}

This is done by inserting an invisible vertical rule, a \texttt{strut} (Section 4.7.6), into the heading text:

\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.

\begin{center}
\begin{tabular}{|c|c|c|c|c|c|}
\hline
Day & 6.15–7.15 pm & 7.20–8.20 pm & 8.30–9.30 pm \\
\hline
\textbf{Subj.} & \textbf{Teacher} & \textbf{Subj.} & \textbf{Teacher} & \textbf{Subj.} & \textbf{Teacher} \\
& \textbf{Room} & & \textbf{Room} & & \textbf{Room} \\
\hline
Mon. & UNIX & Dr. Smith & Fortran & Ms. Clarke & Math. & Mr. Mills \\
& & Comp. Ctr & & Hall A & & Hall A \\
\hline
Tues. & \LaTeX & Miss Baker & Fortran & Ms. Clarke & Math. & Mr. Mills \\
& & Conf. Room & & Conf. Room & & Hall A \\
\hline
Wed. & UNIX & Dr. Smith & C & Dr. Jones & ComSci. & Dr. Jones \\
& & Comp. Ctr & & Hall B & & Hall B \\
\hline
Fri. & \LaTeX & Miss Baker & C++ & Ms. Clarke & & canceled \\
& & Conf. Room & & Conf. Room & & \\
\hline
\end{tabular}
\end{center}
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, 32x CDRW drive, 64 MB TV output, Windows XP, 15&quot; 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, 17&quot; 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 DVD/CD-RW drive, 2x USB, 15&quot; 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 p{\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 & 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 \[1ex\]
FBD 480 & Desktop DeLuxe: same as FBD 360 but with XP4800+ Processor, 48x CDRW drive, 17" monitor & 1399.00 \[1ex\]
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.

\textbf{Warning:} The line termination command \ is ambiguous within a \texttt{p} column, for it can either start a whole new row, or if \texttt{\raggedright} or \texttt{\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</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>March 14 - 16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Introduction to \LaTeX    | Word processors and formatting programs — text and commands — environments — document and page styles — displayed text — math equations — simple user-defined structures | LSEDIT        |
| March 21 - 25             |                                                                                 |               |

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.

```
\newsavebox{\k}
\newsavebox{\kkk}
\sbox{\k}{\framebox[4mm]{\rule{0mm}{3mm}}}
\sbox{\kkk}{\usebox{\k}\usebox{\k}\usebox{\k}\usebox{\k}}
\begin{tabular} {|l|c|c|c|}
\hline
\multicolumn{4}{|c|}{\rule[-0.3cm]{0mm}{0.8cm}\bfseries
Budget Plan 2003--2004}\hline
| Project Nr. | Name | Year | 2003 | 2004 | 2005 |
\hline
| Investment Costs | | | | | |
| Operating Costs | | | | | |
| Industrial Contracts | | | | | |
| Signature | Authorization | | | | |
\hline
\end{tabular}
```

\texttt{\newsavebox{\k}\newsavebox{\kkk}}
\texttt{\sbox{\k}{\framebox[4mm]{\rule{0mm}{3mm}}}}
\texttt{\sbox{\kkk}{\usebox{\k}\usebox{\k}\usebox{\k}\usebox{\k}}}
\texttt{\begin{tabular} {|l|c|c|c|}}
\hline
\texttt{\multicolumn{4}{|c|}{\rule[-0.3cm]{0mm}{0.8cm}\bfseries
Budget Plan 2003--2004}}\hline
| Project | | | | |
| Signature | Authorization | | | | |
\multicolumn{1}{|r|}{Year} & 2003 & 2004 & 2005 \\
\cline{2-4}
& \euro\ & US \$ & \euro\ & US \$ & \euro\ & US \$ & \euro\ & US \$ & \euro\ & US \$ & \euro\ & US \$ & \euro\ & US \$
Investment & \hspace{2.5cm} & \hspace{2.5cm} & \hspace{2.5cm} & \hspace{2.5cm} & \hspace{2.5cm} & \hspace{2.5cm} & \hspace{2.5cm} & \hspace{2.5cm} & \hspace{2.5cm} & \hspace{2.5cm} & \hspace{2.5cm} & \hspace{2.5cm} & \hspace{2.5cm}
Costs & & & & & & & & & & & & & & \\
Operating Costs & & & & & & & & & & & & & & \\
Industrial Contracts & & & & & & & & & & & & & & \\
\multicolumn{4}{|l|}{\rule[-1.0cm]{0mm}{1.3cm}Signature \hspace{5cm} \vline \hspace{5cm} Authorization} \\
\end{tabular}

The first three lines are only indirectly related to the table construction. They arrange for three empty boxes \hspace{2.5cm} 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 \hspace{5cm} Authorization} merges all four table columns into one, in which the text is set flush with the left margin. This text consists first of a strut \rule[-12mm]{0mm}{15mm} that says the height of the last row begins 12 mm below the baseline and is a total of 15 mm high. Then, beginning at the left margin, comes the word Signature, followed 5 cm later by \vline, a vertical line. The word Authorization is separated from the vertical line by a blank space ( ).

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 inter-row 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.

\section{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 `tabular` and `array` environments by adding several column formatting arguments, and by allowing the user to be able to define his or her own such arguments.

- \m{wth} produces a column of width \textit{wth} which is aligned vertically in the middle. (The standard \p{wth} aligns the text with the top line.)
- \b{wth} is like \p{wth} and \m{wth} but aligns the text on the bottom line.
- \>{\textit{decl}} inserts \textit{decl} before the next column; thus >\{\bfseries\} sets the entire column in bold face without having to type \bfseries in each row.
- \<\{\textit{decl}\} inserts \textit{decl} after the last column; to have a centered column in math mode, give >\{$\}$c<\{$\}$.
- \!\{\textit{decl}\} inserts \textit{decl} between two columns without removing the normal intercolumn spacing, as for @\{\textit{decl}\}.

With \texttt{\newcolumntype{\textit{type}}{\textit{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 `dcolumn` package loads the `array` package and defines a column specifier D to align a column of numbers on a decimal point. Its syntax is

\begin{verbatim}
D\{\texttt{\textit{in\_char}}}\{\texttt{\textit{out\_char}}}\{\texttt{\textit{number}}\}
\end{verbatim}

where \texttt{\textit{in\_char}} is the input character for the decimal point (say \texttt{.}), \texttt{\textit{out\_char}} is the character that is output (say \texttt{\cdotp}), and \texttt{\textit{number}} is the maximum number of decimal places. If \texttt{\textit{number}} is negative, the column is centered on the decimal point, otherwise it is right justified. Later versions allow \texttt{\textit{number}} to specify the number of digits on both sides of the decimal point, for example as 3.2.

The `tabularx` package loads the `array` package and defines an environment `tabularx` which makes a table of a desired total width, like `tabular*`, but in which the columns expand, not the intercolumn spacings. The column specifier X is used to indicate the expandable columns, and is equivalent to \p{wth} where \textit{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 \caption[]\{\textit{continued}\} \hline |
| Left & Center & Right \\ \hline |
| \multicolumn{3}{r}{\textit{continued on next page}} \hline |
| Twenty-two & fifty & A hundred and eighty \\ \hline |

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\textsuperscript{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\textsuperscript{b}</td>
<td>2.7</td>
<td>2.3</td>
<td>3.0</td>
</tr>
</tbody>
</table>

\textsuperscript{a}BCU = Bituminous Coal Unit (1 ton BCU corresponds to the heating equivalent of 1 ton of bituminous coal = 8140 kwh)

\textsuperscript{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}
head_text \hspace{1cm} table \hspace{1cm} foot_text \end{table}

where table stands for the entire table as defined in a tabular environment, head_text for whatever text appears above the table, and 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):


\begin{table}  
\centering  
\begin{tabular*}{118mm}{@{}llllll@{}}
\textbf{Primary Energy Consumption}\\
\end{tabular*}  
\end{table}

There are a number of formatting parameters that may be used in connection with the 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}
\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 "\strut" 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 \verb and \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 `verbatim` 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 `verbatim` 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 \emph{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{|}`, one can print short literal text with `|text|`. The counter command `\DeleteShortVerb{|}` then restores the original meaning to `|`. Any character may be temporarily turned into a literal switch this way.

One problem with the `verbatim` environment is that the entire literal text is input and stored before processing, something that can lead to memory overflows. The `verbatim` 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 `\end{verbatim}`.

The `verbatim` 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 `\verbatiminput{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.,

`\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

`\footnote{footnote_text}`
which comes immediately after the word requiring an explanation in a footnote. The text \textit{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\footnote{1}, which is sequentially numbered. This footnote is produced with:

\footnote{The usual method of marking footnotes in a typewritten ... same page.}, which is ...

The footnote numbering is incremented throughout the whole document for the \texttt{article} class, whereas it is reset to 1 for each new chapter in the \texttt{report} and \texttt{book} classes.

The \texttt{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 \texttt{minipage} environment, in which case the footnote text is printed beneath the minipage and not at the bottom of the actual page.\footnote{2}

The \texttt{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:

... of the actual page.\footnote{With nested ... wrong place.}

\subsection{Non-standard footnotes}

If the user wishes the footnote numbering to be reset to 1 for each \texttt{section} command with the \texttt{article} class, this may be achieved with

\setcounter{footnote}{0}

just before or after a \texttt{section} command.

The internal footnote counter has the name \texttt{footnote}. Each call to \texttt{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{\texttt{\thefootnote}}{\texttt{\numberstyle{\texttt{footnote}}} }

\footnote{1The 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.}

\footnote{2With 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 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 \footnote call.

An optional argument may be added to the \footnote command

\begin{verbatim}
\footnote[\textit{num}]{\textit{footnote_text}}
\end{verbatim}

where \textit{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}}
For example\footnote[7]{The 7th symbol ... marker.}, \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.

\section{Footnotes in forbidden modes}

A footnote marker can be inserted in the text with the command

\begin{verbatim}
\footnotemark[\textit{num}]
\end{verbatim}

even where the \footnote command is normally not allowed, that is, in LR boxes, tables, and math mode. The marker is either the optional argument \textit{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[\textit{num}]{\textit{footnote_text}}
\end{verbatim}

If the optional argument has been used for the footnote marker, the same \textit{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 \textit{num} or with that of the footnote counter.

This counter is incremented by a call to \footnotemark without an optional argument. The corresponding \footnotetext command, on the other hand, does not alter the counter.

\begin{itemize}
  \item **The 7th symbol appears as the footnote marker.
\end{itemize}
4.10. Footnotes and marginal notes

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}{\textit{dif}}

where \textit{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 \textit{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 \texttt{minipage} environment. However, the footnote appears underneath the minipage, not below the main page.

Footnote commands within a minipage\textsuperscript{a} have a different marker style. The footnote comes after the next \texttt{\end{minipage}} command.\textsuperscript{b} 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}

Footnote commands within a minipage\footnote{The marker is a raised lower-case letter.} have a different...

\textsuperscript{a}The marker is a raised lower-case letter.
\textsuperscript{b}Watch out for nested minipages.

Footnotes within a \texttt{tabular} environment can normally only be generated with the commands described above: \texttt{\footnotemark} within the table and

\begin{footnotes}
\footnote{Small insects}
\footnote{Large mammals}
\end{footnotes}
\footnotetext outside the environment. However, if the tabular environment is inside a minipage, normal \footnote commands may also be used inside the table. The footnote appears below the table where the minipage comes to an end.

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.

Exercise 4.18: Redefine the command \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.

Exercise 4.19: Complete Exercise 4.15 so that the footnotes $^a$ and $^b$ appear as in the table on page 109.

4.10.5 Marginal notes

Notes in the page margin are produced with the command

\marginpar{note\_text}

which puts the text note\_text into the margin beginning at the level of the line where the command is given. The marginal note appearing here was generated with

\marginpar{This\ is\ a\ margin-\al note}

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 \ 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

\marginpar{\rule[-17.5mm]{1mm}{20mm}}

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 twoside option has been selected. ‘Outer’ means the right margin for odd pages, and left for even ones. With the 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 turn 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 the command \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{\textwidth}{\hht}\}\vspace{-\hht}

A value of 0 cm for the \hht 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 \setlength command.

\section{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 \%. 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 \%.

As for other single character commands, the percent sign itself is printed with the command \%, as explained in Section 2.5.4.

The comment character \% 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 comment environment from the \texttt{verbatim} package (page 111).

Finally, the \% 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.

\textbf{Exercise 4.20:} Comment out the changes from Exercise 4.18 in your preamble. These commands may be reactivated later by removing the \% character.
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} \quad \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{.text}\$ which is actually the \texttt{\LaTeX} method. All three are essentially the same, and there is no reason not to use the \$ sign.

The contents of the formula, \textit{formula\text{.text}}, consist of math constructs, which are described in the following sections.

Displayed formulas, or equations, are produced in the environments

\begin{equation} \text{formula\text{.text}} \end{equation}
\begin{equation*} \text{formula\text{.text}} \end{equation*}

The difference between these two is that the \texttt{equation} environment automatically adds a sequential equation number. The \texttt{displaymath} environment may be given with the shorthand forms \$\ldots\$ or \$\ldots\$.

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 \texttt{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 \texttt{\setlength{\mathindent}{indent}}, where \texttt{indent} is a length specification. Moreover, the document class option \texttt{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} \text{formula\text{.text}} \end{eqnarray}
\begin{eqnarray*} \text{formula\text{.text}} \end{eqnarray*}

where the standard form adds a sequential equation number for each line and the \texttt{\text{*}}-form is without equation numbers.

\section{Main elements of math mode}

\subsection{Constants and variables}

Numbers that appear within formulas are called \texttt{constants}, whereas \texttt{simple variables} are represented by single letters. The universal practice in mathematical typesetting is to put constants in Roman typeface and variables in \textit{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 \texttt{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 \quad \$M(s) \textless M(t) \textless |M| = m\$
\[ y'' = c \{ f[y', y(x)] + g(x) \} \quad \$y'' = c \{ f[y', y(x)] + g(x) \}$

**Preparation:** Create a new L\TeX\ 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 \[ f''(x)g(x) + 2f'(x)g'(x) + f(x)g''(x) \].’

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).

\[ x \wedge 2 \quad x \wedge {2n} \quad a_n \quad a \wedge {n} \quad x \_ {\wedge {n \_ n}} \]

When exponents and indices occur together, their order is unimportant. The last example above could also have been given as \( x \_ n \ \wedge \_ n \).

If the exponent or index contains more than one character, the group of characters must be enclosed in braces \{ \}:

\[ x \wedge {2n} \quad x \wedge {\{2n\}} \quad x_2y \quad x_{\{2\}} \quad A_{i,j,k}^{-n!2} \quad A_{\{i,j,k\}}^{-n!2} \]

Multiple raisings and lowerings are generated by applying \^ and \_ to the exponents and indices:

\[ x \wedge {y^2} \quad x \wedge {\{y^2\}} \quad x \_ {y^1} \quad x \_ {\{y\_1\}} \quad A_{i,j,k}^{x^2} \quad A^{x_{i,j}^2} \_ {\{j \_ {2n}\} \_ {n,m}} \]

**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

\[
\frac{\text{numerator}}{\text{denominator}}
\]

is employed to write the numerator on top of the denominator with a horizontal fraction line of the right width between them.

\[
\frac{1}{x + y} \]

\[
\frac{a^2 - b^2}{a + b} = a - b
\]

Fractions may be nested to any depth within one another.

\[
\frac{\frac{a}{x-y} + \frac{b}{x+y}}{1 + \frac{a-b}{a+b}}
\]

\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

\[
\sqrt[n]{\text{arg}}
\]

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 \textit{arg}:

$\sqrt{x^2 + y^2 + 2xy} = x+y$  $\sqrt{x^2 + y^2 + 2xy} = x+y$, or

\[
\sqrt[n]{x^n - y^n} \]

\[
\sqrt[n]{1 + u^{2n}}
\]

Roots may be nested inside one another to any depth:

\[
\sqrt[3]{-q + \sqrt{q^2 + p^3}}
\]

\[
\sqrt{-q + \sqrt{q^2 + p^3}}
\]
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 \^ and \_. 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:

$$\sum_{i=1}^n \int_a^b$$

Some authors prefer the limits for the integral to be placed above and below the integral sign, the same as for summation. This is achieved with the command \limits immediately following the integral sign: \int\limits_{x=0}^{x=1}$\int$

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 \texttt{\texttt{mathrm}} command (Section 5.4.2). Thus \texttt{\texttt{int y \, mathrm{d}x}} and \texttt{\texttt{int f(z) \, mathrm{d}z}} produce the desired results shown above, whereas \texttt{\texttt{int y \, d\times}} and \texttt{\texttt{int f(z) \, d\, z}} yield $\int y \, dx$ and $\int f(z) \, dz$.

5.2.6 Continuation dots—ellipsis

Formulas occasionally contain a row of dots \ldots, meaning \textit{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

\texttt{\texttt{\texttt{\texttt{1dots} \ldots low dots \cdots center dots}}}
\texttt{\texttt{\texttt{\texttt{vdots} : vertical dots \ddots diagonal dots}}}

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 $a_0, a_1, \texttt{\texttt{\texttt{1dots}}, a_n}$ for the first and $a_0 + a_1 + \texttt{\texttt{\texttt{cdots}}, a_n}$ 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 \dots 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 include 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

\[
x = \frac{1}{n} \sum_{i=1}^{r} p_ix_i, \quad s = \sqrt{\frac{1}{n} \sum_{i=1}^{r} p_i(x_i - \bar{x})^2}
\]

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 \( \int_{-1}^{8} (dx / \sqrt[3]{x}) = \frac{3}{2} (8^{2/3} + 1^{2/3}) = 15/2 \).

## 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

\[ \alpha \ \theta \ \omega \ \tau \ \tau \]
\[ \beta \ \vartheta \ \varpi \ \phi \ \varphi \]
\[ \gamma \ \iota \ \varsigma \ \chi \]
\[ \delta \ \kappa \ \rho \ \varpi \]
\[ \varepsilon \ \lambda \ \varrho \ \psi \]
\[ \zeta \ \nu \ \varsigma \ \omega \]
\[ \eta \ \xi \]

Upper case letters

\[ \Gamma \ \Lambda \ \Sigma \ \Psi \]
\[ \Delta \ \Xi \ \Upsilon \ \Omega \]
\[ \Theta \ \Pi \ \Phi \]

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 math alphabet command \mathcal: \$\mathcal{\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 calligraphic letters may also be used in math formulas:


These are called with the math alphabet command \mathcal:

\$\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 binary operation. The symbols that are available for use as binary operators are
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

\[
\begin{align*}
\pm & \quad \cap & \quad \circ & \quad \bigcirc \\
\mp & \quad \cup & \quad \bullet & \quad \Box \\
\times & \quad \uplus & \quad \diamond & \quad \Diamond \\
\div & \quad \sqcap & \quad \lhd & \quad \bigtriangledown \\
\cdot & \quad \sqcup & \quad \rhd & \quad △ \\
\ast & \quad \oplus & \quad \oslash & \quad \setminus \\
\star & \quad \ominus & \quad \odot & \quad \wr \\
\dagger & \quad \oplus & \quad \oslash & \quad \setminus \\
\ddagger & \quad \ominus & \quad \odot & \quad \wr \\
\amalg & \quad \otimes \\
\leq & \quad \leq & \quad \geq & \quad \geq & \quad \neq & \quad \sim & \quad \not\sim \\
\ll & \quad \gg & \quad \doteq & \quad \doteq & \quad \simeq & \quad \simeq \not\simeq \\
\subset & \quad \supset & \quad \approx & \quad \approx & \quad \asymp & \quad \asymp \\
\sqsubset & \quad \sqsupset & \quad \cong & \quad \cong & \quad \smile & \quad \frown \\
\sqsubseteq & \quad \sqsupseteq & \quad \equiv & \quad \equiv & \quad \bowtie & \quad \bowtie \\
\in & \quad \ni & \quad \prec & \quad \succ \\
\vdash & \quad \dashv & \quad \preceq & \quad \succeq \\
\models & \quad \perp & \quad \parallel & \quad \mid \\
\end{align*}
\]

A number of the above symbols may be called by more than one name. For example, $\leq$ may be produced with either \texttt{\leq} or \texttt{\le}.

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 \texttt{\not}. Thus \texttt{\not\in} yields $\notin$. The same is true for the keyboard characters: \texttt{\not=}, \texttt{\not>}, and \texttt{\not<} produce $\neq$, $>$, and $\neq$. For \texttt{\not=} there is also the special command \texttt{\neq} producing $\neq$ which is a symbol on its own and not the combined $\neq$.

The following symbols may be negated in this manner. Note that the last two, \texttt{\not\in} and \texttt{\notin}, 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 \longrightarrow \uparrow \Leftarrow \Leftarrow \Longleftarrow \Uparrow
- \rightarrow \to \rightarrow \longrightarrow \downarrow \Rightarrow \Rightarrow \Longrightarrow \Downarrow
- \leftrightarrow \leftrightarrow \leftrightarrow \longleftrightarrow \updownarrow \Leftrightarrow \Leftrightarrow \Longleftrightarrow \Updownarrow
- \mapsto \mapsto \mapsto \longmapsto \nearrow \hookleftarrow \hookrightarrow \searrow
- \leftharpoonup \leftharpoonup \leftharpoonup \rightharpoonup \swarrow \leftharpoondown \leftharpoondown \leftharpoondown \rightharpoondown \nwarrow
- \rightleftharpoons \rightleftharpoons \rightleftharpoons \Join

Here again the symbols → and ← may also be referred to under the names \to and \gets. Furthermore, the command \Longleftrightarrow may be substituted by \iff, although the latter (\iff) has a little more spacing on either side than the former (\Longleftrightarrow).

5.3.6 Various other symbols

The above lists by no means exhaust the complete repertoire of mathematical symbols. However, the following are additional characters that standard \LaTeX does make available. (Even more symbols are possible with the \AMS symbol fonts and \amssymb package, Section 12.4.4.)

- \aleph \prime \forall \forall \Box
- \hbar \emptyset \exists \Diamond \nabla \neg \triangle
- \imath \nabla \flat \clubsuit
- \ell \partial \natural \diamondsuit
- \wp \top \# \heartsuit
- \Re \bot \| \| \spadesuit
- \Im \vdash \angle \Join
- \mho \dashv \\backslash \infty
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:

\[ \sum \sum \ \sum \ \int \int \ \int \ \oint \oint \ \prod \prod \ \prod \ \coprod \coprod \ \coprod \]

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 \( ^\wedge \) 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 \( \mathlimits \) forces the limits to be written above and below the symbol where they would otherwise be placed beside it. Similarly the complementary command \( \mathnolimits \) sets them beside the symbol when the standard positioning is above and below.

\[ \oint^\infty_0 \oint\limits^\infty_0 \prod^\nu_{\nu=0} \prod\limits^\nu_{\nu=0} \]

5.3.8 Function names

The universal standard for mathematical formulas is to set variable names in \textit{italics} but the names of functions in Roman. If one were simply to write the function names \textit{sin} or \textit{log} in math mode, \LaTeX{} would interpret these as variables \texttt{s i n} and \texttt{l o g} and write them as \texttt{sin} and \texttt{log}. In order to tell \LaTeX{} that a function name is wanted, it is necessary to enter a command consisting of the backslash \texttt{\textbackslash} plus the function name. The following names are recognized by \LaTeX{}:

\begin{verbatim}
\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
\end{verbatim}
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: \( \lim_{x \to \infty} \) 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 \(_\):

\[
\det \quad \gcd \quad \inf \quad \lim \quad \liminf \quad \limsup \quad \max \quad \min \quad \Pr \quad \sup
\]

Finally, there are the function commands \( \bmod \) and \( \pmod{\text{arg}} \), both of which produce the function \( \text{mod} \) in one of two forms:

\[
\begin{align*}
\$ a \bmod b \$ & \Rightarrow a \mod b \\
\$ y \pmod{a+b} \$ & \Rightarrow y \ (\text{mod} \ a+b).
\end{align*}
\]

With \( \LaTeX \) (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:

\[
\begin{align*}
\hat{a} & \quad \check{a} & \quad \grave{a} & \quad \bar{a} & \quad \tilde{a} & \quad \vec{a} & \quad \dot{a} & \quad \ddot{a} & \quad \mathring{a}
\end{align*}
\]

The letters \( i \) and \( j \) should be printed without their dots when they are given an accent. To accomplish this, type the symbols \( \text{imath} \) and \( \text{jmath} \) instead of the letters, as in

\[
\$\vec{\text{imath}} + \tilde{\text{jmath}}\$: \quad i+j
\]

There are wider versions of \( \hat \) and \( \tilde \) available with the names \( \text{widehat} \) and \( \text{widetilde} \). In this way, these accents may be placed over parts of a formula:

\[
\begin{align*}
\widehat{1-x} = \tilde{y} & \quad \$\text{widehat}{1-x} = \text{widehat}{-y}\$
\end{align*}
\]

\[
\begin{align*}
\widehat{xyz} & \quad \$\text{widetilde}{xyz}\$
\end{align*}
\]

**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 \( A \cup \emptyset = A \) and \( J \cup A = J \).

**Exercise 5.7:** Applying l'Hôpital's rule, one has

\[
\begin{align*}
\lim_{x \to 0} \frac{\ln \sin \pi x}{\ln \sin x} & = \lim_{x \to 0} \frac{\frac{\pi \cos \pi x}{\sin \pi x}}{\frac{\cos x}{\sin x}} = \lim_{x \to 0} \frac{\pi \tan x}{\tan \pi x} = \lim_{x \to 0} \frac{\pi / \cos^2 x}{\pi / \cos^2 \pi x} = \lim_{x \to 0} \frac{\cos^2 \pi x}{\cos^2 x} = 1
\end{align*}
\]
Exercise 5.8: The gamma function $\Gamma(x)$ is defined as

$$\Gamma(x) \equiv \lim_{n \to \infty} n! n^{x-1} \prod_{\nu=0}^{n-1} \frac{n!}{x+\nu} = \lim_{n \to \infty} n! n^{x-1} \frac{n!}{x(x+1)(x+2) \cdots (x+n-1)} \equiv \int_0^\infty e^{-t} t^{x-1} \, dt$$

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:

$$\alpha \vec{x} = \vec{x} \alpha, \quad \alpha \beta \vec{x} = \beta \alpha \vec{x}, \quad (\alpha + \beta) \vec{x} = \alpha \vec{x} + \beta \vec{x}, \quad \alpha (\vec{x} + \vec{y}) = \alpha \vec{x} + \alpha \vec{y}.$$ $$\vec{x} \vec{y} = \vec{y} \vec{x} \text{ but } \vec{x} \times \vec{y} = -\vec{y} \times \vec{x}, \quad \vec{x} \vec{y} = 0 \text{ for } \vec{x} \perp \vec{y}, \quad \vec{x} \times \vec{y} = 0, \text{ for } \vec{x} \parallel \vec{y}.$$ 

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

$$\lim_{x \to 0} \frac{\sqrt{1+x} - 1}{x} = \lim_{x \to 0} \frac{\sqrt{1+x} - 1}{x} \frac{(\sqrt{1+x} + 1)}{\sqrt{1+x} + 1} = \lim_{x \to 0} \frac{1}{\sqrt{1+x} + 1} = \frac{1}{2} \quad (5.1)$$

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = 0 \implies U_M = \frac{1}{4\pi} \int \frac{1}{r} \frac{\partial U}{\partial n} \, ds - \frac{1}{4\pi} \int \frac{1}{r} \frac{\partial}{\partial n} U \, ds \quad (5.2)$$

$$I(z) = \sin\left(\frac{\pi}{2} z^2\right) \sum_{n=0}^{\infty} \frac{(-1)^n \pi^{2n}}{1 \cdot 3 \cdots (4n+1)} z^{4n+1} - \cos\left(\frac{\pi}{2} z^2\right) \sum_{n=0}^{\infty} \frac{(-1)^n \pi^{2n+1}}{1 \cdot 3 \cdots (4n+3)} z^{4n+3} \quad (5.3)$$

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
article, the equations are numbered sequentially throughout the entire document.

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

```
\left lbrack sub_form \right rbrack
```

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`. 
Chapter 5. Mathematical Formulas

\[ \left[ \int + \int \right]_{x=0}^{x=1} \]

The pair of brackets [ ] is adjusted to the size of the enclosed formula, as are the raised exponent and lowered index as well.

The commands \left and \right must appear as a pair. For every \left command there must be a corresponding \right command somewhere afterwards. The pairs may be nested. The first \left is paired with the last \right; the following \left with the second last \right, and so on. There must be the same number of \right as \left commands in a nesting.

The corresponding bracket symbols \lbrack and \rbrack may be perfectly arbitrary and do not need to be a logical pair.

\[ \vec{x} + \vec{y} + \vec{z} = \begin{pmatrix} a \\ b \end{pmatrix} \]

This set of brackets is admittedly unusual but permissible.

\[
\begin{array}{rcl}
y &=& \begin{cases} 
-1 & : & x < 0 \\
0 & : & x = 0 \\
+1 & : & x > 0 
\end{cases} \\
\end{array}
\]

Sometimes a formula contains only a single opening or closing bracket without a corresponding counterpart. However, the \left...\right commands must still be given as a pair, but with a period '.' as an invisible bracket symbol.

\[
y = \begin{cases} 
-1 & : & x < 0 \\
0 & : & x = 0 \\
+1 & : & x > 0 
\end{cases} \\
\]

The array environment in the above example is described in Section 4.8.1 and produces a table in math mode.

The \left...\right commands may be applied to a total of 22 different symbols. These are

\[
\begin{array}{ll}
( & ) \\
[ & ] \\
\{ & \} \\
\| & \|
\end{array}
\]

For example, \left|...\right| produces two vertical bars adjusted in height to contain the enclosed formula text.

**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 \( \cos \left( \frac{\pi}{2} z^2 \right) \) and \( \sin \left( \frac{\pi}{2} z^2 \right) \).
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 \mbox{normal text} given inside the formula, together with horizontal spacing commands such as \quad or \hspace. For example:

\[
X_n = X_k \quad \text{if and only if} \quad Y_n = Y_k \quad \text{and} \quad Z_n = Z_k
\]

\[
[ \quad X_n = X_k \quad \text{\quad if and only if} \quad \text{\quad and} \quad Z_n = Z_k \quad ]
\]

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:

\[
\text{\mathrm \mathtt \mathbf \mathsf \mathit \mathcal}
\]

We have already met \mathcal in Section 5.3.2 on calligraphic letters. All these commands function the same way: they set their argument in the corresponding font.

\[
B^0(x) \quad T^i_j \quad \text{\quad $\mathbf{B}^0(x)$ \quad $\mathsf{T}^i_j$}
\]

The command \mathnormal in Section 5.3.1 also belongs to this group. The difference between it and \mathit 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.

\[
$\text{\mathnormal{differ} \ne \mathit{differ}}$
\]

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 \mbox.

5.4.3 Matrices and arrays

\[
a_{11} \quad a_{12} \quad \cdots \quad a_{1n} \\
\vdots \quad \vdots \quad \ddots \quad \vdots \\
a_{n1} \quad a_{n2} \quad \cdots \quad a_{nn}
\]

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 \texttt{array} environment, whose syntax and construction are described in Section 4.8.1 on tables. The \texttt{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 \\
  \ddots & \ddots & \ddots & \ddots & \ddots \\
  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@{:+:}}} is an abbreviation for three repetitions of the column definition \texttt{c@{:+:}.} \texttt{c} defines the column to be one of centered text. \texttt{\multicolumn{5}{c}} says that the next five columns are to be merged and replaced by one with centered text. The command \texttt{\dotfill} fills the column with dots.

It is possible to nest \texttt{array} environments:

\begin{align*}
\begin{array}{c}
  x_{11} & x_{12} \\
  x_{21} & x_{22} \\
  y \\
  z
\end{array}
\end{align*}

\begin{align*}
\begin{array}{c}
  a_{q_1} & a_{q_2} & \cdots & a_{q_k} \\
  a_{q_1} & a_{q_2} & \cdots & a_{q_k} \\
  \ddots & \ddots & \ddots & \ddots \\
  a_{q_1} & a_{q_2} & \cdots & a_{q_k}
\end{array}
\end{align*}

The outermost array consists of one column with centered text (\texttt{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 \texttt{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.
5.4. Additional elements

\[ \sum_{p_1 < p_2 < \cdots < p_{n-k}} \left| \begin{array}{ccc} a_{q_1q_1} & a_{q_1q_2} & \cdots & a_{q_1q_k} \\ a_{q_2q_1} & a_{q_2q_2} & \cdots & a_{q_2q_k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{q_kq_1} & a_{q_kq_2} & \cdots & a_{q_kq_k} \end{array} \right| \]

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.

\[ x - \begin{array}{c} a_1 \\ \vdots \\ a_n \end{array} \]

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 \( \left| \ldots \right| \), 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

\[ \pm \begin{vmatrix} x_1 - x_2 & y_1 - y_2 & z_1 - z_2 \\ l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \end{vmatrix} \begin{vmatrix} l_1 & m_1 \\ l_2 & m_2 \end{vmatrix}^2 + \begin{vmatrix} m_1 & n_1 \\ m_2 & n_2 \end{vmatrix}^2 + \begin{vmatrix} n_1 & l_1 \\ n_2 & l_2 \end{vmatrix}^2 \]

If the numerator is zero, the two lines meet somewhere.

Note: we do not recommend using \{ \texttt{\framebox{cc}} \} in the formatting argument of the three determinants in the denominator under the root sign. Here the \texttt{\left\ldots\right\}} pair should be applied. Try out both possibilities for yourself and compare the results.

Exercise 5.15: Laurent expansion: using \( c_n = \frac{1}{2\pi i} \oint (\zeta - a)^{-n-1} f(\zeta) \, d\zeta \), for every function \( f(z) \) the following representation is valid (\( n = 0, \pm 1, \pm 2, \ldots \))

\[
f(x) = \sum_{n=-\infty}^{+\infty} c_n (z-a)^n = \begin{cases} \quad c_0 + c_1 (z-a) + c_2 (z-a)^2 + \cdots + c_n (z-a)^n + \cdots \\ + c_{-1} (z-a)^{-1} + c_{-2} (z-a)^{-2} + \cdots \\ + c_{-n} (z-a)^{-n} + \cdots \end{cases}
\]

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}} \quad and \quad \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:

\[
\overline{a^2 + xy + z} \quad [ \quad \overline{\underbrace{\overline{a^2} + \underline{xy}}} + \overline{\overbrace{\overline{z}}} \quad ]
\]

The command \texttt{\underline{sub_form}} may also be employed in normal text mode to underline text, whereas \texttt{\overline{sub_form}} is allowed only in math mode.

Exactly analogous to these are the two commands

\texttt{\overbrace{sub_form}} \quad and \quad \texttt{\underbrace{sub_form}}

which put horizontal curly braces above or below the sub-formula.

\begin{align*}
\overbrace{a + b + c + d} & \quad \text{overbrace}\{a + \underbrace{b+c} + d\} \\
\end{align*}

In displayed formulas, these commands may have exponents or indices attached to them. The (raised) exponent is set above the \texttt{\overbrace{sub_form}} while the (lowered) index is placed below the \texttt{\underbrace{sub_form}}.
\[
\underbrace{a + b + \cdots + y + z}_{\alpha \beta \gamma}
\]

\[
\overbrace{123} + z \overbrace{\{\alpha \beta \gamma\}} \]

**Exercise 5.16:** The total number of permutations of \( n \) elements taken \( m \) at a time (symbol \( P^m_n \)) is

\[
P^m_n = \prod_{i=0}^{m-1} (n - i) = \frac{n(n-1)(n-2) \ldots (n-m+1)}{\text{total of } m \text{ factors}} = \frac{n!}{(n-m)!}
\]

### 5.4.5 Stacked symbols

The command

\[
\stackrel{\text{upper sym}}{\text{lower sym}}
\]

places the symbol \( \text{upper sym} \) centered on top of \( \text{lower sym} \), whereby the symbol on top is set in a smaller typeface.

\[
\vec{x} \overset{\text{def}}{=} (x_1, \ldots, x_n) \quad \text{vec}\{x\} \stackrel{\text{def}}{=} (x_1, \ldots)
\]

\[
A \overset{\alpha'}{\longrightarrow} B \overset{\beta'}{\longleftarrow} C \quad \text{A} \stackrel{\alpha'}{\longrightarrow} \text{B} \ldots
\]

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 \( \preceq \) instead of \( \leq \). This is achieved by combining \( < \) and \( = \) with \$\text{\textstyle<}\{=\}$\$. If the command \text{\textstyle} were omitted, the symbol would be printed as \( \preceq \).

### 5.4.6 Additional \TeX{} commands for math

The \TeX{} math commands \texttt{\atop} and \texttt{\choose} are useful additions to the set of commands and may be applied within any \LaTeX{} document. (In fact, all \TeX{} math commands except \texttt{\eqalign}, \texttt{\eqalignno}, and \texttt{\leqalignno} may be used in a \LaTeX{} manuscript.) Their syntax is

\[
\{\text{top} \ \atop \ \text{bottom}\}
\]

\[
\{\text{top} \ \choose \ \text{bottom}\}
\]

Both commands produce a structure that looks like a fraction without the dividing line. With the \texttt{\choose} command, this structure is also enclosed within round brackets (in mathematics this is called a binomial coefficient).

\[
\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}
\]
\[ \prod_{j \geq 0} \left( \sum_{k \geq 0} a_{jk} z^k \right) = \sum_{n \geq 0} z^n \left( \sum_{k_0, k_1, \ldots \geq 0 \atop k_0 + k_1 + \cdots = 0} a_{0k_0} a_{1k_1} \ldots \right) \]

Similar structures can be generated with the \LaTeX environments
\begin{array}{c} \text{upper line} \\ \text{lower line} \end{array} \quad (\text{atop})
\left(\begin{array}{c} \text{upper} \\ \text{lower} \end{array}\right) \quad (\text{choose})

The difference between these array structures and those of the \TeX commands is that the former are always printed in the size and style of normal text formulas, whereas the latter will have varying sizes depending on where they appear within the formula.

For comparison
\[ \Delta p_1 p_2 \cdots p_{n-k} \]
\[ \begin{array}{c} p_1 p_2 \cdots p_{n-k} \\ p_1 p_2 \cdots p_{n-k} \end{array} \]

The above \TeX commands may also be employed to produce small matrices within text formulas, such as \( \begin{pmatrix} 1 & 0 \end{pmatrix} \) or \( \begin{pmatrix} a & b & c \\ l & m & n \end{pmatrix} \). Here the first matrix was typed in with
\$\{1\atop0\choose0\atop1\}$
and the second with
\$\left(\begin{array}{c} a\atop b\atop c \end{array}\right)$

The syntax of these Plain \TeX 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{verbatim}
\begin{eqnarray} line_1 \& \cdots \& line_n \end{eqnarray}
\begin{eqnarray*} line_1 \& \cdots \& line_n \end{eqnarray*}
\end{verbatim}
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
When printed, all the \textit{left} formulas appear right justified in a left column, the \textit{right} formulas left justified in a right column, and the \textit{mid} formulas centered in between. The column separation character \& designates the various parts of the formula. Normally the \textit{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} \{rcl\} \ldots \end{array} environment.

The difference between the \texttt{array} and \texttt{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 \texttt{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 \texttt{eqnarray} environment adds an automatic sequential equation number, which is missing in the \texttt{*}-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{\textit{name}} once the keyword \textit{name} has been assigned to the equation number with the \texttt{\label{\textit{name}}} command somewhere in the line of the equation. See Section 9.2.1 for more details.

Examples:

\begin{eqnarray}
(x+y)(x-y) & = & x^2 - xy + xy - y^2 \\
& = & x^2 - y^2 
\end{eqnarray}

\begin{eqnarray}
(x+y)^2 & = & x^2 + 2xy + y^2
\end{eqnarray}
Chapter 5. Mathematical Formulas

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 \( \backslash \backslash \) 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 \( \left( \ldots \right. \backslash \ldots \backslash \left. \right) \). 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 \texttt{eqnarray*} example on the previous page.

Occasionally it is better to break long multiline equations as follows:

\[
\begin{eqnarray*}
& & w + x + y + z = \\
& & a + b + c + d + e + f + g + h + i + j + k + l
\end{eqnarray*}
\]

that is, the second and subsequent lines are not aligned with the equals sign but are left justified with a certain indentation from the beginning of the first line.

The command \texttt{\left\{eqnarray*\}} \texttt{\left\{w+x+y+z = \right. \backslash \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{depth}} between the \texttt{\left\{eqnarray*\}} \texttt{\left\{w+x+y+z = \right. \backslash \backslash} and the \texttt{\backslash \right. \}} line termination. A positive value for \texttt{depth} increases the indentation, a negative value decreases it.
Exercise 5.17: The following equations are to be broken as shown:

\[
\arcsin x = - \arcsin(-x) = \frac{\pi}{2} - \arccos x = \left[ \arccos \sqrt{1 - x^2} \right] \\
= \arctan \frac{x}{\sqrt{1 - x^2}} = \left[ \arccot \frac{\sqrt{1 - x^2}}{x} \right]
\] (5.6)

\[
f(x + h, y + k) = f(x, y) + \left\{ \frac{\partial f(x, y)}{\partial x} h + \frac{\partial f(x, y)}{\partial y} k \right\} \\
+ \frac{1}{2} \left\{ \frac{\partial^2 f(x, y)}{\partial x^2} h^2 + 2 \frac{\partial^2 f(x, y)}{\partial x \partial y} kh + \frac{\partial^2 f(x, y)}{\partial y^2} k^2 \right\} \\
+ \frac{1}{6} \{ \cdots \} + \cdots + \frac{1}{n!} \{ \cdots \} + R_n
\] (5.7)

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 \( y = f(x) = ax + bx^2 + cx^3 + dx^4 + ex^5 + fx^6 + \cdots \quad (a \neq 0) \) begins with the elements

\[
x = \phi(y) = \frac{1}{a} y - \frac{b}{a^3} y^2 + \frac{1}{a^5} (2b^2 - ac) y^3 \\
+ \frac{1}{a^7} (5abc - z^2 d -.fb^3) y^4 \\
+ \frac{1}{a^9} (6a^2 bd + 3a^2 e^2 + 14b^4 - a^3 e - 21ab^2 c) y^5 \\
+ \frac{1}{a^{11}} (7a^3 be + 7a^3 cd + 84ab^3 c - a^4 f - 28a^2 b^2 d - 28a^2 bc^2 - 43b^5) y^6 + \cdots
\]

Select a value for the declaration \texttt{\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.
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.

\begin{equation}
\begin{align*}
\alpha &= f(z) \\
\beta &= f(z^2) \\
y &= f(z^3)
\end{align*}
\end{equation}

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.

Vertical boxes can also be useful when equation numbers are placed in an unconventional manner. The eqnarray environment generates an equation number for every line, which may be suppressed with \nonumber. To add a vertically centered equation number to a set of equations, for example,

\begin{equation}
P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n
\end{equation}

the following text may be given:

\begin{equation}
P(-x) = a_0 - a_1 x + a_2 x^2 - \cdots + (-1)^n a_n x^n
\end{equation}

Emphasizing formulas by framing requires no new construction elements. It is sufficient to put them into an \fbox (Section 4.7.7). Text formulas \fbox{$a+b$} are simply framed with \fbox{$a+b$}. For displayed formulas, \texttt{\texttt{displaystyle}} (Section 5.5.2) must be issued, else they are set as text formulas.

\[
\int_0^\infty f(x) \, dx \approx \sum_{i=1}^n w_i e^{x_i} f(x_i)
\]

is produced with

\[
[\fbox{$\texttt{displaystyle} \int^n \texttt{infty}_0 f(x), \texttt{mathrm}{d}x ..$}]
\]

An alternative method to frame displayed equations is with the \texttt{\texttt{AMSTeX boxed}} command, page 270.
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{} that we met in Section 5.4.2:

\[ S^{-1}\mathbf{T}S = dg(\omega_1, \ldots, \omega_n) = \mathbf{\Lambda} \]

produces \( S^{-1}\mathbf{T}S = dg(\omega_1, \ldots, \omega_n) = \mathbf{\Lambda} \).

In this example, the entire formula has been set as the argument of \textbf{} 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 bold Roman with \textbf{}. Lower case Greek letters and other math symbols appear in the normal 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{} command.

\[ \mathbf{2\sqrt{x}/y} = z \]

The math font style command \textbf{\textbf} 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{\textbf} 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{\textbf} resets the math fonts back to the normal ones.

\[ \oint_C V \, d\tau = \oint_\Sigma \nabla \times V \, d\sigma \]

If \textbf{\textbf} has been turned on outside the math mode, it may be temporarily turned off inside with \textbf{\textbf}: 

\[ W_r = \int M \, d\varphi = r^2 m \omega^2 / 2 \]

An alternative method of printing single symbols in bold face is provided by the \textbf{\textbf} 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 \textit{}:

\[ \textit{Fe}_2^{2+}\text{Cr}_2\text{O}_4 \]
5. 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 $B \mathbf{v} \omega$.
3. Tensors of 2nd order and matrices may appear in a sans serif font, as $M D I$.
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 \char`\textbackslash \textbackslash 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 \textbackslash \vec command, which produces $\vec{B}$. Nor does \textbackslash \mathbf help, for this yields $B$, in an upright font. The best solution is to use the \textbackslash \texttt{boldsymbol} command from the \texttt{AMS} package \texttt{amsbsy} (Section 12.2.1) or the \textbackslash \texttt{bm} command from the tools package \texttt{bm} (Section B.5.4). Otherwise one must resort to defining

\begin{verbatim}
\renewcommand{\vec}[1]{\mbox{\textbf{#1}}}
\end{verbatim}

for a revised \texttt{vec} command.

Similarly point 3 can be met with the math alphabet command \texttt{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 \texttt{mathrm} math alphabet command. However, to simplify the application, it is recommended to create some user-defined commands, such as

\begin{verbatim}
\newcommand{\me}{\texttt{mathrm{e}}} for math \, e
\newcommand{\mi}{\texttt{mathrm{i}}} for math \, i
\newcommand{\dif}{\texttt{mathrm{d}}} for 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_{\text{\texttt{\textbackslash infty}}}^{\text{\texttt{\textbackslash infty}}} f(x) \, \texttt{\textbackslash dif} x \approx \texttt{\textbackslash sum}_{i=1}^{n} w_i \me^{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 \texttt{\textbackslash dif} or \texttt{\textbackslash mathrm{d}} of the previous section.)

Small amounts of horizontal spacing can be added in math mode with the commands

\begin{itemize}
  \item \texttt{\textbackslash ,} small space = \( 3/18 \) of a quad
  \item \texttt{\textbackslash ;} medium space = \( 4/18 \) of a quad
  \item \texttt{\textbackslash ;} large space = \( 5/18 \) of a quad
  \item \texttt{\textbackslash !} negative space = \( -3/18 \) of a quad
\end{itemize}

In the following examples, the third column contains the results without the additional horizontal spacing command.

\[
\begin{array}{lll}
\sqrt{2}x & \sqrt{x} & \sqrt{2}x \\
\sqrt{\log x} & \sqrt{\log x} & \\
o \left(1/\sqrt{n}\right) & O\left(1/\sqrt{n}\right) & \\
[0,1) & (0,1) & \\
\log n & \log n & \log n \\
x^2/2 & x^2/2 & \\
n/\log n & n/\log n & \\
\Gamma_2 + \Delta^2 & \Gamma_2 + \Delta^2 & \\
R_{i}^{\ j} & R_{i}^{\ j} & \\
\int_{0}^{x} \! \int_{0}^{y} dF(u, v) & \int_{0}^{x} \! \int_{0}^{y} dF(u, v) & \\
& \int \! \int_{D} dxdy & \\
& \int \! \int_{D} dxdy & \\
\end{array}
\]

Note: In the third last example \( R_{\ j}^{\ i} \) 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 \( R_{\ j}^{\ i} \), instead of \( R_{i}^{\ j} \) which is produced by \( R_{\ i}^{\ j} \).

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 \TeX selects for the various parts of the formula. First we must explain what sizes are available in math mode and what \TeX'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} \textit{D} Normal size for displayed formulas
\item \texttt{\textbackslash textstyle} \textit{T} Normal size for text formulas
\item \texttt{\textbackslash scriptstyle} \textit{S} Normal size for first sub-, superscript
\item \texttt{\textbackslash scriptscriptstyle} \textit{SS} Normal size for later sub-, superscripts
\end{itemize}

From now on we shall make use of the symbolic abbreviations \textit{D}, \textit{T}, \textit{S}, and \textit{SS}. When math mode is switched on, the active font size becomes \textit{D} for displayed and \textit{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 \textit{D}, the smaller to \textit{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>\textit{D}</td>
<td>\textit{T}</td>
<td>\textit{T}</td>
<td>\textit{S}</td>
</tr>
<tr>
<td>\textit{T}</td>
<td>\textit{S}</td>
<td>\textit{S}</td>
<td>\textit{S}</td>
</tr>
<tr>
<td>\textit{S}</td>
<td>\textit{SS}</td>
<td>\textit{SS}</td>
<td>\textit{SS}</td>
</tr>
<tr>
<td>\textit{SS}</td>
<td>\textit{SS}</td>
<td>\textit{SS}</td>
<td>\textit{SS}</td>
</tr>
</tbody>
</table>

If the active size is \textit{D}, size \textit{T} is selected for both parts of the fraction. That is, for both numerator and denominator \textit{T} becomes the active size. If they in turn contain further fractions, these will be set in \textit{S}. If the active size is \textit{D} or \textit{T}, superscripts and subscripts (exponents and indices) appear in \textit{S}; within them \textit{S} is the active size and any fractions or shiftings inside them will be set in \textit{SS}.

The \TeX elements \texttt{\{ \textbackslash atop \}} and \texttt{\{ \textbackslash choose \}} are treated as fractions.

The active font size inside an array environment is \textit{T}.

The smallest available math font size is \textit{SS}. Once it has been reached, no further reduction is possible, so that all subsequent superscripts and subscripts appear in \textit{SS} as well.

From the table one easily sees that

\[
\left[ \frac{a_0}{\frac{1}{a_1 + \frac{1}{\frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4}}}}} \right]
\]

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{\textstyle} 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.

\[
\frac{a}{x-y} + \frac{b}{x+y}
\]

\[
1 + \frac{a-b}{a+b}
\]

\[
e^{-\frac{x_i-x_j}{n^i+n^j}}
\]

\[
\begin{pmatrix}
ab & e+f \\
(ch) & g-h \\
0 & ij
\end{pmatrix}
\]

\[
\left(\begin{array}{cc}
0 & \frac{e+f}{g-h} \\
ij & kl
\end{array}\right)
\]

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

\texttt{\newcommand{\D}{\textstyle}\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 `\big`, `\Big`, `\bigg`, or `\Bigg` before the bracket symbol.

- **none**: \( () [ ] \{ \} \langle \rangle \slash \| \uparrow \uparrow \downarrow \downarrow \$)
- **\big**: \( () [ ] \{ \} \langle \rangle \slash \| \uparrow \uparrow \downarrow \downarrow \$)
- **\Big**: \( () [ ] \{ \} \langle \rangle \slash \| \uparrow \uparrow \downarrow \downarrow \$)
- **\bigg**: \( () [ ] \{ \} \langle \rangle \slash \| \uparrow \uparrow \downarrow \downarrow \$)
- **\Bigg**: \( () [ ] \{ \} \langle \rangle \slash \| \uparrow \uparrow \downarrow \downarrow \$)

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 \TeX. The user may alter them at any time with the `\setlength` command in the usual manner.
5.5. Fine-tuning mathematics

\texttt{\arraycolsep}
Half the width of the intercolumn spacing for the \texttt{array} environment (see also Section 4.8.2).

\texttt{\jot}
Extra vertical spacing that appears between the rows of multiline equations in the \texttt{eqnarray} and \texttt{eqnarray*} environments.

\texttt{\mathindent}
The amount of indentation for math formulas when the document class option \texttt{fleqn} has been selected.

\texttt{\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 \textit{long}.

\texttt{\belowdisplayskip}
The vertical spacing inserted below a \textit{long} displayed formula.

\texttt{\abovedisplayshortskip}
Vertical spacing added above a \textit{short} displayed formula. This is one in which the left edge is to the right of the end of the preceding line of text.

\texttt{\belowdisplayshortskip}
Vertical spacing inserted after a \textit{short} displayed formula.

\texttt{\topsep}
The above four spacings are not used with the document class option \texttt{fleqn} where instead \texttt{\topsep} is inserted above and below displayed formulas (see Section 4.4.2).

All the above parameters, except \texttt{\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 \texttt{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.

\textbf{Exercise 5.19:} Generate this continued fraction. \[
a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4}}}}
\]

Note: in contrast to the example on page 146, the 1 here in the numerator appears left justified.

Hint: do you remember the command \texttt{\hfill}?

\textbf{Exercise 5.20:} Produce the following set of equations with the \texttt{array} environment.

\[
\begin{align*}
\sin 2\alpha &= 2 \sin \alpha \cos \alpha, & \cos 2\alpha &= \cos^2 \alpha - \sin^2 \alpha \\
\sin 3\alpha &= 3 \sin \alpha - 4 \sin^3 \alpha, & \cos 3\alpha &= 3 \cos^2 \alpha - 3 \cos \alpha \\
\sin 4\alpha &= 8 \cos^3 \alpha \sin \alpha - 4 \cos \alpha \sin \alpha, & \cos 4\alpha &= 8 \cos^4 \alpha - 8 \cos^2 \alpha + 1
\end{align*}
\]
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.

<table>
<thead>
<tr>
<th>Equations for the tangential plane and surface normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equation for the surface</td>
</tr>
<tr>
<td>( F(x, y, z) = 0 )</td>
</tr>
<tr>
<td>( z = f(x, y) )</td>
</tr>
<tr>
<td>( x = x(u, v) )</td>
</tr>
<tr>
<td>( y = y(u, v) )</td>
</tr>
<tr>
<td>( z = z(u, v) )</td>
</tr>
<tr>
<td>( r = r(u, v) )</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 \( r_1 = \frac{\partial r}{\partial u}, r_2 = \frac{\partial r}{\partial v} \).

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{\textbackslash displaystyle} and \( \texttt{bm} \) for \texttt{\textbackslash boldmath} and possibly even \( \texttt{ba} \) and \( \texttt{ea} \) for \texttt{\textbackslash begin\{array\}} and \texttt{\textbackslash 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{\textbackslash 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{[t]} positioning argument. Do not forget to activate the size \texttt{\textbackslash 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{\textbackslash [\ldots]}. 
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 \texttt{array} environments. The third column contains three fractions whose denominators may be produced either with \texttt{array} or with the \TeX\ \{\ldots\atop\ldots\} command; the fractions themselves are placed in two rows with another \texttt{array} environment.

6. The second and third columns of the last mathematical row again contain \texttt{array} environments. Some of the sub-formulas in this row appear in \texttt{\textbf{boldmath}}. Recall that this command may only be invoked within text mode, that is, within an \texttt{\textbf{mbox}}.

7. In the last row, all three columns of the outer \texttt{array} environment are merged together and the text is set within a \texttt{parbox} of appropriate width:

\begin{verbatim}
\multicolumn{3}{|c|}{\parbox{..}{......}}
\end{verbatim}

\section*{5.6 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, AMS, which has supported \TeX\ from its beginning, has developed a set of packages to assist authors in composing even more complicated formulas with \LaTeX. Known as \texttt{AMS-LaTeX}, it includes many extra commands, further environments to supplement \texttt{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 \textdollar\ 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 \verb|\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 \LaTeXe, 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 \texttt{graphics} and \texttt{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 \texttt{encapsulated} PostScript. Image format conversion may be required.

6.1 The \texttt{graphics} packages

Strictly speaking, these packages, referred to as the \texttt{graphics collection}, are extensions to the basic \LaTeX installation. We describe them here
Chapter 6. Graphics Inclusion and Color

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 grfguide.tex, which may already be preprocessed to .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 The \LaTeX Graphics Companion (Goossens 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 graphics and 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 .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<>\dagger pctexhp<>\dagger tcidvi\dagger
dvipdfm  emtex<>\dagger pctexps  textures
dvips    oztex   pctexwin<>\dagger truetex\dagger
dvipsone pctex32 pdftex  vtex
\end{verbatim}

(Limited functionality: < no color support; = no scaling; \dagger no rotation.)

The xdvi previewer (and its Windows equivalent windvi) work with PostScript graphics; although there is also an xdvi option, it is only an alias for dvips.

A driver option must be specified when loading the graphics and 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 graphics package and the more extended 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 `graphics` or `graphicx` are loaded:

- **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;

- **final** counteracts `draft`; needed when the `draft` option has been issued globally in `\documentclass`;

- **hidescale** leaves blank space where scaled text should be;

- **hiderotate** leaves blank space where rotated text should be; this and the option `hidescale` are useful if the previewer cannot handle scaling or rotation;

- **hiresbb** look for bounding box values in `%%HiResBoundingBox` instead of the normal `%%BoundingBox` line.

### 6.1.2 Importing with the `graphics` package

The basic importing command with the `graphics` package is

```latex
\includegraphics[llx,lly][urx,ury]{file_name}
```

where `llx, lly` are the coordinates of the lower left corner, and `urx, 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 `[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 `[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 encapsulated PostScript files with extension `.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

```latex
\includegraphics{clock}
```

(It is not necessary, nor recommended, to include the file extension `.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:

\begin{verbatim}
\scalebox{h_scale}[v_scale]{text}
\end{verbatim}

which applies horizontal and vertical scale factors to the contents text; if v_scale is omitted, it is the same as h_scale;

\begin{verbatim}
\resizebox{h_length}{v_length}{text}
\end{verbatim}

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 v_length to refer to the height + depth of the box, rather than just to the height. In both cases, the contents 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

\begin{verbatim}
\reflectbox{text}
\end{verbatim}

**Rotation**

Rotation of a box about the left-hand end of its baseline is done with

\begin{verbatim}
\rotatebox{angle}{text}
\end{verbatim}

where 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

\begin{verbatim}
\rotatebox{30}\{\resizebox{!}{2cm}{\includegraphics{clock}}
\end{verbatim}

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
6.1. The graphics packages

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 \TeX document, with some normal text above and below it. The result should appear as:

```
%!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 pdf\TeX ;

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 pdf\TeX , this option must be used rather than bb;

trim = dl lx dl ly dur x 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 \%\%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° 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 (\LaTeX 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

\begin{verbatim}
\epsfysize=y\textwidth \epsfxsize=x\textwidth
\epsf[llx lly urx ury]{file name}
\end{verbatim}

The `epsfig` package also defines an importing command that makes use of the regular keys and values to enter its parameters:

\begin{verbatim}
\epsfig{file=file name, key=value, ...}
\end{verbatim}

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: landscape

Another extra package in the `graphics` bundle is `landscape`, by David Carlisle. This defines a `landscape` environment that prints its contents rotated 90° 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.

#### Package: rotating

The `rotating` package by Sebastian Rahtz and Leonor Barroca tries to make the interface for rotation somewhat simpler. It defines

\begin{verbatim}
\begin{sideways} text \end{sideways}
\begin{turn}{angle} text \end{turn}
\begin{rotate}{angle} text \end{rotate}
\end{verbatim}

where `sideways` rotates `text` by 90°, `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{verbatim}
\setlength{\unitlength}{0.01\linewidth}
\setlength{\fboxsep}{0pt}
\setlength{\fboxrule}{1.5pt}
\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}
\end{verbatim}

Figure 6.1: An embellished image file
6.1. The graphics packages

\drawline(71.2,26)(20.5,36.7)
\drawline(71.2,40.6)(20.5,67.2)
\drawline(57.40.6)(-10,67.2)
\put(-10,37){\fbox{\includegraphics[width=.3\linewidth, viewport=390 180 490 280,clip]{texlive}}}
\put(-9,19){\vector(1,4){4}}
\put(-9,18){\makebox(0,0)[t]{\large Blow-Up}}
\put(43,2){\hyperlink{info}{\makebox(30,7){}}}
\end{picture}

Here the unit length for the picture placement has been set to $1/100$ of the current line width \texttt{\linewidth}, so we can conveniently use 100 units for the full width. The picture environment reserves a space of $80 \times 80$ units; the image file \texttt{texlive} is placed at the lower left corner (location $0,0$) and set to a width of $80\%$ of \texttt{\linewidth}. We then draw a framed box around a portion of this image, and then add that portion as a second image (selected with \texttt{viewport} and clipped) as a blow-up. The \texttt{drawline} commands (from the \texttt{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 (\texttt{vector}) and the text 'Blow-up' (with \texttt{makebox}). Finally, the \texttt{hyperlink} command from the \texttt{hyperref} package turns the $30 \times 7$ unit area around the text 'TEXLive!' into an active link to the target named \texttt{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 \texttt{\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 \texttt{.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 \texttt{.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

```
%%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 pdf\TeX\ program (Section 10.2.3) is the most convenient way to produce it with \LaTeX. However, pdf\TeX\ 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 pdf\TeX. 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 pdf\TeX, 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: pdf\TeX

The single most common error when importing a graphics file with pdf\TeX 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 pdf\TeX and are astonished to get the error message:

```
! LaTeX Error: Unknown graphics extension: .eps.
```

As explained above, pdf\TeX cannot handle PostScript files, so the graphics must be converted to, or regenerated in, another format.
6.1. The graphics packages

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 pdfTeX, 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 \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 \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 \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 `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 `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 `dvips` is the standard graphics driver, you may add
to \texttt{graphics.cfg} and you do not need to give the option \texttt{dvips} when loading \texttt{graphics} or \texttt{graphicx}. You may still specify any other graphics driver option if you wish.

Many supplied configuration files go further and determine whether \texttt{pdftex} is activated or not; if so, the default graphics option is set to \texttt{pdftex}, otherwise to \texttt{dvips}.

The rest of the configuring commands in this section can be issued either in the \texttt{graphics.cfg} file, or in the document.

\section*{Search path for graphics files}

One can specify the directories where \TeX{} is to look for graphics files with

\begin{verbatim}
\graphicspath{\{dir\list\}}
\end{verbatim}

\noindent where \texttt{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, \TeX{} 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}

\textbf{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.

\section*{Default extensions}

A list of default extensions for the graphics files can be defined with

\begin{verbatim}
\DeclareGraphicsExtensions{\texttt{\{ext\list\}}}
\end{verbatim}

\noindent This means that only the root name of the file must be given and \TeX{} will attempt to find it by attaching all the possible extensions. For PostScript drivers, the \texttt{ext\list} is usually set to \texttt{.eps}, \texttt{.ps}. At our installation, we also include the non-standard extension \texttt{.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.

\section*{Graphics types}

Defining the extensions is only part of the task: one must also associate each extension with a graphics type so that \TeX{} knows how to process it. PostScript recognizes only one type, \texttt{eps}, encapsulated PostScript, but there do exist other types such as \texttt{bmp} and \texttt{pcx} for other drivers. For the non-standard \texttt{.psc} extension above, we must also give

\begin{verbatim}
\DeclareGraphicsRule{\texttt{.psc}}{\texttt{eps}}{}{}
\end{verbatim}

\noindent to inform \TeX{} that this extension belongs to type \texttt{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,

\begin{verbatim}
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{'gunzip -c #1}
\end{verbatim}

which establishes the graphics type as eps, with the bounding box information in the file of the same name and extension .eps.bb, and that the operating system command gunzip -c must be applied to the file (represented as #1). The single quote ' is required to indicate a system command. The \%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

\begin{verbatim}
\DeclareGraphicsRule{.eps-gz}{eps}{.bb}{'gzip -d -c #1}
\end{verbatim}

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{nodvipsnames}

- 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}}
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 \([\text{rgb}]\{1,0,0\}\) defines red, \([\text{cmyk}]\{0,0,1,0\}\) yellow. The *gray* model takes only one number. The *named* model accesses colors by internal names that were originally built into the \texttt{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

\[
\texttt{\definecolor{name}{model}{specs}}
\]

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 \([\text{model}]\{\text{spec}\}\), like \([\text{rgb}]\{0,1,0\}\).

\texttt{\pagecolor{col_spec} sets the background color for the current and following pages;}

\texttt{\color{col_spec} is a declaration to switch to setting text in the given color;}

\texttt{\textcolor{col_spec}{text} sets the text of its argument in the given color;}

\texttt{\colorbox{col_spec}{text} sets its argument in a box with the given color as background;}

\texttt{\fcolorbox{col_spec1}{col_spec2}{text} like \colorbox, with a frame of \texttt{col_spec1} around a box of background color \texttt{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;}

\texttt{\normalcolor switches to the color that was active at the end of the preamble. Thus placing a \texttt{color} command in the preamble can change the standard color for the whole document. This is the equivalent to \texttt{\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 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{\textheight}{22cm}
\setlength{\textwidth}{16cm}
\setlength{\textwidth}{16cm}
\setlength{\textwidth}{16cm}
\setlength{\textwidth}{16cm}
\setlength{\textwidth}{16cm}
\setlength{\textwidth}{16cm}
\setlength{\textwidth}{16cm}
\setlength{\textwidth}{16cm}
\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 palette.tex or copy it from the enclosed CD from \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{verbatim}
\begin{figure}[where] figure \end{figure}
\begin{figure*}[where] figure \end{figure*}
\begin{table}[where] table \end{table}
\begin{table*}[where] table \end{table*}
\end{verbatim}

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.
7.3. Style parameters for floats

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

```
\suppressfloats[loc]
```

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 `loc` 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

172

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>15 000</td>
<td>145 000</td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>Network costs</td>
<td>5 000</td>
<td>23 000</td>
<td>28 000</td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td>Repairs</td>
<td>25 000</td>
<td>6 000</td>
<td>31 000</td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>Expendables</td>
<td>68 000</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

\addcontentsline and \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 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,

\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:

\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.)
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></td>
<td>3 000</td>
<td>83 000</td>
</tr>
<tr>
<td>1.5</td>
<td>Expendables</td>
<td></td>
<td></td>
<td>42 000</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 \texttt{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{\superscript{\texttt{floats}}} (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 \( t \) 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{\bfseries Results and Seat Distribution of the...}
\mbox{\small}
\begin{minipage}[b]{7.7cm}

\textbf{Results and Seat Distribution of the 2003 General Election}

\begin{tabular}{|c|c|c|}
\hline
\textbf{Pros} & \textbf{Cons} & \textbf{Seat Distribution in the Assembly of 2004–2006} \\
\hline
140 & 186 & \textbf{Results: 2003 General Election} \\
\hline
60 & 42 & \\
\hline
\end{tabular}

\begin{tabular}{|c|c|c|}
\hline
\textbf{Party} & \textbf{Votes} & \textbf{%} \\
\hline
Pros & 15 031 287 & 37.7 \\
Cons & 12 637 418 & 31.7 \\
Ups & 6 499 210 & 16.3 \\
Downs & 4 486 514 & 11.3 \\
Others & 1 201 078 & 3.0 \\
\hline
\end{tabular}

\end{minipage}
\end{table}
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
Chapter 7. Floating tables and figures

\caption{\label{budget04} Computer Center ... }

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’.

7.7 Some float packages

7.7.1 Placing a float right here

Package: 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 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 \pagebreak command is first issued. This will most likely lead to an ugly page break.

7.7.2 Flowing text around a float

Package: floatflt

The package floatflt by Mats Dahlgren (based on the earlier and now obsolete 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 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 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 \centering command at the start of the figure material.

The material in the floatingfigure environment need not actually be a figure. In the above example, some boxed text in a minipage was used for demonstration purposes. However, if the \caption command is included in the environment, then a figure number will be printed, in sequence with those of the regular figure environment.

Rather than giving the pos parameter each time, one can specify a default value as an option when loading the package with \usepackage:
Some float packages

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{wth} 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

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

\texttt{\listof{plate}{List of Plates}}

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
The float package also defines 4 float styles:

- **plain**  much like standard \LaTeX\ except the caption is always \textit{below} the float contents;
- **plaintop**  like plain but with the caption always \textit{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 \texttt{\floatstyle{style}}, where \texttt{style} is one of the above possibilities. The style then applies to all new float environments defined with \texttt{\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 \texttt{figure} and \texttt{table} floats, issue

\begin{verbatim}
  \floatstyle{ruled}
  \restylefloat{figure}
  \restylefloat{table}
\end{verbatim}

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:

- part
- chapter
- paragraph
- figure
- enumi
- section
- subparagraph
- table
- enumii
- subsection
- page
- footnote
- enumiii
- subsubsection
- equation
- mpfootnote
- enumiv

The meanings of most of these counters are obvious from their names and need no further explanation. The counters enumi... enumiv refer to the four levels of the \texttt{enumerate} environment (Sections 4.3.4 and 4.3.5), while the counter 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 \textit{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 \textit{counter_name} nor any name of a previously defined
user counter.

The optional argument \textit{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 \textit{in_counter} is incremented by one
of the commands \texttt{\stepcounter} or \texttt{\refstepcounter} (see below). For
example, the \texttt{subsection} counter is reset to zero whenever the \texttt{section}
counter is incremented.

The \texttt{\newcounter} command may not appear in any file that is read in
with the \texttt{\include} command (Section 9.1.2). It is therefore best to put all
\texttt{\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 \texttt{counter} is as-
signed the integer value \texttt{num}.

\addtocounter{counter}{num}
With this command, the value of the \texttt{counter} is increased by the
integer \texttt{num}, which may be positive or negative.

\stepcounter{counter}
The value of the \texttt{counter} is increased by one, and at the same
time all its sub-counters (that is, those that have this counter as
their \texttt{in_counter}) are reset to zero (see above).

\refstepcounter{counter}
This command has the same effect as \texttt{\stepcounter} but also
makes \texttt{counter} the current counter for the cross-referencing com-
mand \texttt{\label} (see Section 9.2.1).

This last command may be applied, for example, within the \texttt{figure}
or \texttt{table} environments when the \texttt{\caption} command is missing and yet
there is to be a reference in the text to the figure or table number using the
\texttt{\ref} command. Then \texttt{\refstepcounter{figure}} or \texttt{\refstepcounter}
\texttt{\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
\texttt{\textbackslash value\{counter\}}
which does not change the value at all. This command is mostly used in connection with \texttt{\textbackslash setcounter} or \texttt{\textbackslash 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 \texttt{\textbackslash setcounter\{mypage\}\{\textbackslash value\{page\}\}}.

8.1.4 Printing counter values

The numerical value in a counter can be printed with the commands
\begin{itemize}
  \item \texttt{\textbackslash arabic\{counter\}} as an Arabic number,
  \item \texttt{\textbackslash Roman\{counter\}} as a capital Roman numeral,
  \item \texttt{\textbackslash roman\{counter\}} as a lower case Roman numeral,
  \item \texttt{\textbackslash alpha\{counter\}} as a lower case letter,
  \item \texttt{\textbackslash Alph\{counter\}} as a capital letter,
  \item \texttt{\textbackslash fnsymbol\{counter\}} as a footnote symbol.
\end{itemize}

For the commands \texttt{\textbackslash alpha} and \texttt{\textbackslash 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 \texttt{\textbackslash 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
\begin{itemize}
  \item \texttt{\textbackslash the\{counter\}}
\end{itemize}
is also available, consisting of \texttt{\textbackslash the} immediately followed by the name of that counter, such as \texttt{\textbackslash the\{page\}}. This type of command is initially identical to \texttt{\textbackslash arabic\{counter\}}, but may redefined to be composed of several counter commands. In the document classes book and report, for example, the command \texttt{\textbackslash thesection} is defined in terms of both the chapter and section numbers: \texttt{\textbackslash arabic\{chapter\}.\textbackslash arabic\{section\}}. Here is the result of printing \texttt{\textbackslash 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 \texttt{\textbackslash the\{counter\}} commands. If a different format for some automatic numbering is desired, say alphabetical equation numbers, the definition of the corresponding \texttt{\textbackslash the\{counter\}} command can be altered using the methods described in Section 8.3.

**Exercise 8.1:** Take your standard exercise.tex file and print out the final values of the \LaTeX counters with \texttt{\textbackslash arabic\{counter\}} commands at the end. Change some the values with the \texttt{\textbackslash setcounter} and \texttt{\textbackslash 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{\length_cmd}{\length_spec}

where \texttt{length_spec} may be a length specification (with units) or another length parameter. In the latter case, \texttt{length_cmd} takes on the current value of that other parameter. Thus with \texttt{\setlength{\rightmargin}{\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

\addtolength{\length_cmd}{\length_spec}

which adds \texttt{length_spec} to the value of the length parameter \texttt{length_cmd}. A negative value for \texttt{length_spec} decreases \texttt{length_cmd} by that amount. Once again, another length parameter may be used for \texttt{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: \texttt{0.5\textwidth} means half the width of the text column and \texttt{2\parskip} twice the inter-paragraph spacing.

With the command

\settowidth{\length_cmd}{\text}

the length parameter \texttt{length_cmd} is set equal to the natural length of a piece of text.

Similarly the commands

\settoheight{\length_cmd}{\text}
\settodepth{\length_cmd}{\text}

set the \texttt{length_cmd} equal to the height and depth of the \texttt{text} above and below the baseline, respectively.

Finally, the command

\stretch{\texttt{decimal_num}}

yields a rubber length that is \texttt{decimal_num} times as stretchable as \texttt{\fill} (Section 2.4.2).

A user-defined length parameter is created with
\newlength{\new_len_cmd}

which establishes \new_len_cmd as a length with a value of 0pt. All the above commands may be used to manipulate its value further.

The command

\addvspace{length_spec}

inserts extra vertical spacing of the amount length_spec at that point where it appears. If more than one such command is given, the total inserted spacing will be that of the largest argument and not the sum of them all. This command may only be given between paragraphs. Its application for user-defined commands and environments lies in the generation of structures that should behave like paragraphs.

8.3 User-defined commands

New commands may be defined or redefined under \LaTeX with the commands

\newcommand{\com_name}[narg][opt]{def}
\renewcommand{\com_name}[narg][opt]{def}

The first version is used to define a command \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 \com_name. In both cases, an error message is printed if the incorrect variant is called. The first optional argument narg is a number between 1 and 9 specifying how many arguments the new or altered command is to have. A second optional argument 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 def.

8.3.1 Commands without arguments

We will first illustrate the use of the \newcommand without the optional argument [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

\newcommand{\xvec}{x_1, \ldots, x_n}

creates a new command named \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 \xvec is called, it is its definition that goes into the L\TEX processing.

Since the new command \xvec contains a math command (the subscript command _), it may only be called within math mode. Thus $\xvec$ 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 \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{x_1, \ldots, x_n}}
\end{verbatim}

Now both \xvec and $\xvec$ are allowed, both with the same result.

The above text mode example was actually written as \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 \xvec and ... produces ‘$x_1, \ldots, x_n$ and ...’ without any spacing between. This problem is solved by adding a space command \_ or the empty structure {} after the command name, in this case as \xvec\_ or \xvec{}.

It would also have been possible to include the blank in the definition of \xvec, as {\ensuremath{x_1, \ldots, 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 ‘programmed’ blank will always be present, even when some punctuation or other symbol directly follows the command.

A better solution is provided by the xspace package in the tools collection (Section B.5.4). With this package, one adds the command \xspace at the end of a definition where a space may appear:

\begin{verbatim}
\newcommand{\xvec}{\ensuremath{x_1, \ldots, x_n}\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 \newcommand, although in fact this command may only be used once to initiate a user-defined command that does not already exist. Once \xvec has been created, a revised definition can only be given with the command \renewcommand. This was indeed done with the second and subsequent definitions of \xvec.

Following this example, the user may employ \newcommand (or alternatively \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 pos-
sible 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
Chapter 8. User Customizations

#1 in the command definition is a dummy argument representing the text of *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 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

\newcommand{anvec}[2]{\text{ensuremath}{#1_1, \ldots, #1_{#2}}} \text{and invoked as } \text{anvec}{u}{n} \text{ for } u_1, \ldots, u_n \text{ and as } \text{anvec}{v}{m} \text{ for } u_1, \ldots, u_m. \text{ The optional argument } [2] \text{ for } \text{newcommand} \text{ says that the command being defined contains two arguments; in the definition part, } #1 \text{ is replaced by the first argument and } #2 \text{ by the second. By imagining } u \text{ or } v \text{ in place of } #1 \text{ and } n \text{ or } m \text{ where } #2 \text{ stands, one can see how the command } \text{anvec}{arg1}{arg2} \text{ operates.}

This pattern may be carried on for even more arguments. With

\newcommand{subvec}[3]{\text{ensuremath}{#1_{#2}, \ldots, #1_{#3}}} \text{ a command } \text{subvec} \text{ is defined with three arguments. It should be clear from the definition that calling } \text{subvec}{a}{i}{j} \text{ produces } a_i, \ldots, a_j. \text{ A command argument that consists of only a single character need not be put into curly brackets } \{ \} \text{ 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 } \text{subvec aik} \text{ is the same as } \text{subvec}{a}{i}{k}, \text{ and } \text{subvec xin} \text{ produces the same structure } x_1, \ldots, x_n \text{ as our first user-defined example } \text{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 } \text{subvec}{A}{ij}{lk} \text{ prints out } A_{ij}, \ldots, A_{lk}. \text{ The three replacement arguments are } A \text{ for } #1, \text{ } i \text{ for } #2, \text{ } \text{and } k \text{ for } #3.

Why does \text{subvec}{A}{ij}{lk} \text{ produce } A_{ij}, \ldots, A_{lk} \text{ and not the expected result } A_{ij}, \ldots, A_{ik}? \text{ 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 } \text{ensuremath}{A_{ij}, \ldots, A_{lk}} \text{ so that only the first characters following the subscript symbols } _{ } \text{ 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}. \text{ This may be achieved with the } \text{subvec} \text{ command by including an extra set of brackets in the arguments: } \text{subvec}{A}{ij}{lk}. \text{ A better solution, however, is to put the brackets in the definition to begin with:}

\renewcommand{\subvec}[3]{\text{ensuremath}{#1_{#2}, \ldots, #1_{#3}}} \text{ which will always produce the desired result with only a single set of brackets per argument: } \text{subvec}{A}{ij}{lk} \text{ 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 \texttt{\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 \texttt{\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 \texttt{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_{\ldots,#1_{#3}}}}
\end{verbatim}

The difference between this and the previous definition is the addition of \texttt{[x]} after the \texttt{[3]} argument. This states that the first of the three arguments is to be optional, and its standard value is \texttt{x}. Now \texttt{\subvec{i}{j}} prints \texttt{x_i,...,x_j} while \texttt{\subvec[a]{1}{n}} produces \texttt{a_1,...,a_n}.

There may only be one optional argument in the user-defined command, and it will always be the first one, the \texttt{#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.

\textbf{Example 1:} In Section 5.4.6, the \TeX commands \texttt{\atop} and \texttt{\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 \texttt{\frac}. However,

\begin{verbatim}
\newcommand{\latop}[2]{{#1\atop#2}}
\newcommand{\lchoose}[2]{{#1\choose#2}}
\end{verbatim}

define two commands \texttt{\latop} and \texttt{\lchoose} that yield the same results with a syntax like that of \LaTeX: \texttt{\latop{upper}{lower}}.

\textbf{Example 2:} The command \texttt{\defbox{sample_text}} is to set a box width equal to the length of the text \texttt{sample_text}. A subsequent call to the command \texttt{\textbox{text}} centers \texttt{text} within a frame with the same width as \texttt{sample_text}.

\begin{verbatim}
\newlength{\width}
\newcommand{\defbox}[1]{\settowidth{\width}{#1}}
\newcommand{\textbox}[1]{\framebox[\width]{#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.)

\textbox{as wide as this text}
\defbox{as wide as this text}\textbox{}
\textbox{text}
\textbox{longer text}

**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 \footnote{* † ‡ § ¶ ∥ ∗∗ †† ‡‡} 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}\[page\]

making a user-defined counter myfn that is set to zero every time the page counter is incremented.* Next the command

\renewcommand{\thefootnote}{\fnsymbol{footnote}}

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.

**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

---

*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 \texttt{\reseteqn} restores the numbering scheme to its original style. Thus a sequence of equation numbers could be 4, 5, 6-\texttt{a}, 6-\texttt{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 \texttt{equation} is saved in the counter \texttt{saveeqn} and then incremented, while equation itself is set to zero. The form of the equation marker, \texttt{\theequation}, is redefined using these two counters. The equation numbering routines will operate on \texttt{equation} as usual, leaving \texttt{saveeqn} unchanged. The resetting command \texttt{\reseteqn} puts the value of \texttt{saveeqn} back into \texttt{equation} and restores the definition of \texttt{\theequation}.

This example is only appropriate for the document class \texttt{article}. For \texttt{report} and \texttt{book}, the definition of \texttt{\theequation} is

\[\arabic{chapter}.\arabic{equation}\]

The necessary modifications are left as an exercise for the user.

The \texttt{\mbox} command in the first \texttt{\renewcommand{\theequation}} defining the combined equation number is necessary because the result will be printed in \textit{math mode}, where the hyphen ‘-’ is interpreted as a binary operator (minus sign) with extra spacing between it and its two ‘operands’, \texttt{\arabic{saveeqn}} and \texttt{\alph{equation}}. Thus 6 - $a$ would be output instead of 6-\texttt{a}. The \texttt{\mbox} command causes a temporary switch out of math into text mode.

\textbf{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 \texttt{\mathsf} accomplishes this only for an upright font. The solution is to create a new math alphabet \texttt{\mathsfsl} for use in a \texttt{\tensor} command.

\begin{verbatim}
\DeclareMathAlphabet{\mathsfsl}{OT1}{cmss}{m}{sl}
\newcommand{\tensor}[1]{{\mathsfsl{#1}}}
\end{verbatim}

The NFSS system of font attributes is presented in Section A.1, and the \texttt{\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}{\mathbf{#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 theAMS\textit{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 \texttt{LaTeX} commands \( \Lbrack \) and \( \Lbrace \) in the same manner as in Example 1 for \texttt{latopt} and \texttt{lchoose} corresponding to the \texttt{LaTeX} commands \( \lbrack \) and \( \lbrace \). These \texttt{LaTeX} commands behave as \texttt{lchoose} from Section 5.4.6 except that they enclose their contents in square brackets \( \Lbrack \) or curly braces \( \Lbrace \). (Note that the names \( \lbrack \) and \( \lbrace \) 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{\Alph{}}...\texttt{\Roman{}} as given by the argument type. For example, 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 \( \iint{(D)} \), \( \iiint{V} \), and \( \idotsint{G} \) should produce:

\[
\begin{align*}
\iint{(D)} & \quad \iiint{V} & \quad \idotsint{G}
\end{align*}
\]

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 subscripted symbol is needed using \texttt{\hspace{-...}}.

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 \texttt{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

\begin{verbatim}
\usepackage{ifthen}
\end{verbatim}

in the preamble. It then makes available the two commands \texttt{\ifthenelse} and \texttt{\whiledo}, which have the following syntaxes:

\begin{verbatim}
\ifthenelse{test}{then_text}{else_text}
\whiledo{test}{do_text}
\end{verbatim}

In both cases, \texttt{test} is a logical statement (explained below); for the first command, \texttt{then_text} or \texttt{else_text} is inserted into the text depending on whether \texttt{test} is \texttt{⟨true⟩} or \texttt{⟨false⟩}. For the second command, the \texttt{do_text} is inserted (executed) as long as \texttt{test} evaluates to \texttt{⟨true⟩}. (The \texttt{do_text} must alter the inputs to \texttt{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 \texttt{<}, \texttt{=}, or \texttt{>} between them, which stand for \textit{less than}, \textit{equals}, and \textit{greater than}, respectively. The value of a counter may be tested by putting its name as the argument of the \texttt{\value} command. Examples:

\begin{verbatim}
\newcommand{\three}{3}
\ifthenelse{\three = 3}{O.K.}{What?}
\ifthenelse{\value{page} < 100}{Page xx}{Page xxx}
\end{verbatim}

The first case prints \texttt{O.K.} since \texttt{\three} is equal to 3; in the second case \texttt{Page xx} is printed if the current page number is less than 100, else \texttt{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 \texttt{\isodd}

\begin{verbatim}
\ifthenelse{\isodd{\value{page}}}{odd}{even}
\end{verbatim}
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}}{Frederick}{??}

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 \LaTeX 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

\texttt{\textbackslash and \textbackslash or \textbackslash not \textbackslash ( \textbackslash )}

which should be straightforward to anyone familiar with boolean logic. For example, to set \texttt{\textwidth} to 10 cm if two-column mode is active or if \texttt{\paperwidth} is greater than 15 cm and the page counter is below 100,

\begin{verbatim}
\ifthenelse {\lengthtest{\textwidth > 10cm} \or
\( \lengthtest{\paperwidth > 15cm} \and
\value{page} < 100 \) }
{\setlength{\textwidth}{10cm}} {}
\end{verbatim}

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.

\begin{verbatim}
\newboolean{US}
\setboolean{US}{true} %For American spelling
%\setboolean{US}{false} %For British spelling
\newcommand{\USUK}[2]{\ifthenelse{\boolean{US}}{#1}{#2}}
\end{verbatim}

Now \texttt{\USUK} is a command that prints its first or second argument according to the setting of the flag \texttt{US}. Thus in the text one may write

... the \texttt{\USUK{color}{colour}} of music ...

which will yield American spelling if \texttt{\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 \texttt{\whiledo}: one wishes to write some text $n$ times, where both the text and $n$ are to be variable.

\begin{verbatim}
\newcounter{mycount}
\newcommand{\replicate}[2]{\setcounter{mycount}{#1} \whiledo{\value{mycount}>0} \setcounter{mycount}{-1}}
\end{verbatim}

Now \texttt{\replicate{30}{?}} will print 30 question marks.

\section{User-defined environments}

Environments may be created or changed with the commands
where the arguments have the following meanings:

- **env_name**: the name of the environment; for \newenvironment, it may not be the same as any existing environment or command name, whether \LaTeX{} or user-defined. For \renewenvironment, on the other hand, there must already be an environment bearing this name. Any changes to \LaTeX{} environments should only be undertaken if the user knows what he or she is doing.

- **narg**: a number between 1 and 9 that states how many arguments the environment is to have; if the optional argument *narg* is omitted, the environment is to have no arguments.

- **opt**: the default text for the first argument (#1) if it is to be optional; this behaves the same as for \(\text{\texttt{re}new\texttt{command}}\) (page 185).

- **beg_def**: the *initial* text to be inserted when \begin{env_name} is called; if this text contains entries of the form #n, with n = 1, ..., narg, then when the environment is started with the call
  \begin{env_name}{arg_1}...{arg_n}...
  each occurrence of #n within \texttt{beg_def} is replaced by the text of the argument arg_n.

- **end_def**: the *final* text that is inserted when \end{env_name} is called; here the dummy arguments #n are not allowed since they are only to appear in the \texttt{beg_def} text.

### 8.4.1 Environments without arguments

Just as for user-defined commands, environments without the optional argument *narg* will be illustrated first. A user-defined environment named sitquote is created with

\begin{verbatim}
\newenvironment{sitquote}{\begin{quote}\small\itshape}{\end{quote}}
\end{verbatim}

which sets the text appearing between \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 \begin{quote} \texttt{\small\itshape} while \texttt{end_def} is simply \end{quote}. Now the call

\begin{verbatim}
\begin{sitquote} text \end{sitquote}
\end{verbatim}

is the same as

\begin{verbatim}
\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{\small \textit{at the beginning of the \texttt{quote} environment. However, it is more consistent with the concept of \textit{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:

\begin{verbatim}
\newcounter{com}
\newenvironment{comment}
{\noindent\slshape Comment:\texttt{\begin{quote}\small\textit{}}}
{\stepcounter{com}\hfill(\arabic{com})\texttt{\end{quote}}}
\end{verbatim}

where now \texttt{beg_def} contains the text and commands
\begin{verbatim}
\noindent\slshape Comment:\texttt{\begin{quote}\small\textit{}}
\end{verbatim}

and \texttt{end_def} is
\begin{verbatim}
\stepcounter{com}\hfill(\arabic{com})\texttt{\end{quote}}
\end{verbatim}

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{verbatim}
\begin{comment}
  This is a comment. Comments should ...
  ... in round parentheses.
\end{comment}
\end{verbatim}

will generate the following type of structure:

Comment:

\begin{quote}
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. \hfill (1)
\end{quote}

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.

\begin{verbatim}
\renewenvironment{comment}[1]{
\begin{sloppypar}\noindent\slshape Comment: #1
\begin{quote}\small\itshape
}{\stepcounter{com}\hspace*{\fill}(\arabic{com})\end{quote}\end{sloppypar}}
\end{verbatim}

The text \begin{comment}{Helmut Kopka} This is a modified ... \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.*

(2)

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}{\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{verbatim}
\begin{comment}[Patrick W. Daly]More than ...
... appropriate.\end{comment}
\end{verbatim}

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
\begin{varlist}{Sample}{\Alph}
\end{varlist}
to be equal to the width of the item word, and then with
\begin{varlist}{Sample}{\Alph}
\end{varlist}
to be 1 cm larger. Length assignments for \texttt{labelwidth} and \texttt{labelsep} 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:
8.5. Some comments on user-defined structures

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 \texttt{\newcommand} or \texttt{\newenvironment} that must be used and not the \texttt{\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 \texttt{\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 \texttt{\renewcommand} or \texttt{\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 \texttt{\newsavebox} or with \texttt{\newcounter} are globally defined. If they are given within an environment, their definitions remain effective even outside that environment. Similarly \texttt{\setcounter} functions globally, although \texttt{\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

\begin{verbatim}
\newcommand{\A}{defa}
\newcommand{\B}{defb}
\newcommand{\C}{\A \B}
\end{verbatim}

where \texttt{\C} is defined in terms of \texttt{\A} and \texttt{\B}. However, it is also permissible to write

\begin{verbatim}
\newcommand{\C}{\A \B}
\textit{normal text, but without calling \C}
\newcommand{\B}{defb}
\newcommand{\A}{defa}
\textit{further text with any number of calls to \texttt{\A}, \texttt{\B} and \texttt{\C}}
\end{verbatim}

### 8.5.6 Nested definitions

User definitions may be nested inside one another. A structure such as

\begin{verbatim}
\newcommand{\outer}{\{\newcommand{\inner}...\}}
\end{verbatim}

is permissible. The command that is defined with the name \texttt{\inner} is valid and known only within the command \texttt{\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 \textit{second} call to the outer command, since the inner definition still exists as a leftover from the first call. Nevertheless, here is an example:

\begin{verbatim}
\newcommand{\twentylove}
{\{\newcommand{\fivelove}
{\{\newcommand{\onelove}
{I love \LaTeX!}\%
\onelove\onelove\onelove\onelove\onelove}\}}
\fivelove\fivelove\fivelove\fivelove\fivelove}
\end{verbatim}
Chapter 8. User Customizations

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! 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! 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 \ldots ##9, while those for the outer
one are the normal #1 \ldots #9. For example:

\begin{verbatim}
\newcommand{\thing}[1]{\color{#1}{red} \color{#1}{green} \color{#1}{blue}}
\end{verbatim}

The entry The colors of the objects are\thing{dress}\thing{book}\thing{car}
produces

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 \ldots ######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

\begin{verbatim}
\listfiles
\end{verbatim}

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.

\textbf{Exercise 9.1:} 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
9.1. Processing parts of a document

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

\texttt{\textbackslash include\{filename\}}

which is only allowed within the text part of the document, together with

\texttt{\textbackslash includeonly\{file\ list\}}

in the preamble, containing a list \texttt{file\ list} of those files that are to be read in. The file names are separated by commas and the extension \texttt{.tex} is left off.

If \texttt{filename} is in the \texttt{file\ list}, or if \texttt{\includeonly} is missing from the preamble, the command \texttt{\include\{filename\}} is identical to

\texttt{\textbackslash clearpage \textbackslash input\{filename\} \textbackslash clearpage}

However, if \texttt{filename} is not contained within the \texttt{file\ list}, \texttt{\include} is equivalent to \texttt{\clearpage \input\{filename\} \clearpage} and the file contents are not read in. However, an auxiliary \texttt{.aux} file is read in that sets all the counters and cross-referencing information for that file.

Since \texttt{\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, \texttt{\include} commands may not be nested: they may only appear in the main processing file. However, an \texttt{\input} command may be given within a file that is \texttt{\included}.

The great advantage of the \texttt{\include} command is that the additional information about page, section, and equation numbers will be supplied by the \texttt{.aux} files that are read in in place of the \texttt{.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 \texttt{\ref} and \texttt{\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 \texttt{file\_3} ends on page 17, but after the selective processing it now extends to page 22. The following \texttt{file\_4} still begins on page 18, and all further files also have their original starting page numbers. If \texttt{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 \texttt{file\_3}. So far so good. However, if instead \texttt{file\_6} were to be selectively processed right after \texttt{file\_3}, it would receive its starting page number from \texttt{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 \texttt{\include} command is extremely useful for large documents, saving considerable computation time. Longer documents are normally written and edited in many stages. The \texttt{\include} command permits one to reprocess selective alterations in a short time, even if the numbering systems temporarily go awry. This can be repaired later on with a complete reprocessing of the entire document, by deactivating the \texttt{\includeonly} command in the preamble.

A file that is read in with \texttt{\include} may not contain any \texttt{\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 \texttt{chap1.tex}, \texttt{chap2.tex}, \ldots. The processing file itself contains the text

\begin{verbatim}
\documentclass{book}
\begin{document}
\frontmatter
\include{toc}
\mainmatter
\include{chap1}
\include{chap2}
\include{chap3}
\end{document}
\end{verbatim}

where the file \texttt{toc.tex} contains only commands for the title page, table of contents, and other listings:

\begin{verbatim}
\begin{titlepage} ... \end{titlepage}
\tableofcontents \listoftables \listoffigures
\end{verbatim}

and \texttt{back.tex} those for bibliography and index:

\begin{verbatim}
\bibliography{..}
\printindex
\end{verbatim}

The chapters to be processed are selected by adding them to the argument of \texttt{\includeonly}, not by changing the \texttt{\include} commands, which must always be present to ensure that all the .aux are input every time. For example, by giving \texttt{\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 \TeX write a message to the computer monitor during processing. This can be achieved with the command

\typeout{message}

where *message* stands for the text that is to appear on the monitor. This text is printed when the \TeX processing reaches this command. It is also written to the transcript file, but does not appear in the processed output.

If *message* contains a user-defined command, it will be interpreted and its translation appears on the monitor. The same thing applies to \TeX commands. This could have dire consequences if the commands, either user or \TeX, are not really printable. To print the command name literally, precede it with the command \protect \typeout{message}

The command

\typein[\com_name]{message}

also writes the *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{(return)} key. If the optional argument \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 \typein{Name:}\ ...
\end{verbatim}

then what appears on the monitor is:

\begin{verbatim}
Name:
\@typein=
\end{verbatim}

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 \typein command contains the optional argument \com_name, this is treated the same as

\typeout{message} \newcommand{\com_name}{entered definition}

In this case the definition is stored under the command name \com_name interactively and may be invoked and executed in the rest of the document like any other \TeX command.

With some experience in \TeX methodology, it soon becomes obvious that interactive processing with the \typein command can be very practical. For example, if the preamble contains

\begin{verbatim}
\typein[\files]{Which files?}
\includeonly[\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{\textbackslash 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 \texttt{exer1.tex}, \texttt{exer2.tex}, and \texttt{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: \texttt{\textbackslash typein=} Finger Wrestling
Unit: \texttt{unit=} Points
Gold: \texttt{\textbackslash typein=} A. T. Glitter
Country: \texttt{\textbackslash typein=} AUR
Value: \texttt{\textbackslash typein=} 7999.9
Silver: \texttt{\textbackslash typein=} S. Lining
. . . = . . .
```

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

\begin{verbatim}
\label{marker}
\end{verbatim}

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

\begin{verbatim}
\pageref{marker}
\end{verbatim}

at any point earlier or later in the document.

If the \texttt{\label} command is given after a sectioning command, or within an equation, eqnarray, or enumerate environment, or inside the argument of a \texttt{\caption} within the figure or table environments, the command
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}[Balzano--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{\textbackslash 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 document, 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 circumvent 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 tools package \texttt{varioref} 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, \texttt{Fig. \vref\{f1\}} automatically produces one of

\begin{verbatim}
Fig. 5 when \label\{key\} on same page,
Fig. 5 on the preceding page on previous page,
Fig. 5 on the next page on following page,
Fig. 5 on the facing page on opposite page with twoside option,
Fig. 5 on page 24 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
Chapter 9. Document Management

the example above when \lref{example} on same page,
the example on the previous page on following page,
the example on page 24 when two of more pages away,

while \lref{above example}[example]{f1} outputs one of:
the above example when \lref{example} on same page,
the example on the next page on the following page,
the example on page 24 when two of more pages away.

A range of numbers and pages can be produced with the command \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 \vrefrange only is printed with \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 \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 \texttt{in questa pagina} for ‘on this page’ and \texttt{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:

\texttt{\reftextbefore} previous but non-facing page
\texttt{\reftextfacebefore} previous but facing page
\texttt{\reftextafter} next but non-facing page
\texttt{\reftextfaceafter} next but facing page
\texttt{\reftextfaraway{key}} more than one page away
\texttt{\reftextpagerange{key1}{key2}} page range of 2 labels
\texttt{\reftextlabelrange{key1}{key2}} range of 2 different labels.

The last three must be redefined with 1 or 2 arguments, as
\texttt{\renewcommand{\reftextfaraway}[1]{on page \pageref{#1}}}  \quad \texttt{--\pageref{#1}}
\texttt{\renewcommand{\reftextpagerange}[2]{on pages \pageref{#1}%\pageref{#2}}}  \quad \texttt{\ref{#1} to \ref{#2}}

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.

9.3 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 \LaTeX, 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 \LaTeX 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 \BibTeX program that should be part of the \LaTeX 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 \BibTeX.

\section{Bibliography by hand}

The actual bibliography, whether created by \BibTeX or by hand, is placed inside the environment

\begin{verbatim}
\begin{thebibliography}{sample_label}
  entries
\end{thebibliography}
\end{verbatim}

The individual \textit{entries} in the bibliography each begin with the command

\texttt{\bibitem[label]{key} entry_text}

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 \textit{sample_label}, so this should be as large as the longest label in the bibliography. For the standard application with running numbers, \textit{sample_label} should be a dummy number with as many digits
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 \texttt{thebibliography} are the same, except that the optional \texttt{label} must be present, taking a special form that will transfer the author and year texts to the citation commands.

A sample (numerical) \texttt{thebibliography} environment could look as follows:

\begin{thebibliography}{99}
\bibitem{lamport} Leslie Lamport. \textsl{\LaTeX\ -- A Document Preparation System}, 2nd edition. Addison-Wesley, Reading, MA, 1994
\bibitem{knuth} Donald E. Knuth. \textsl{Computers and Typesetting Vol. A--E}. Addison-Wesley, Reading, MA, 1986
\bibitem[6a]{knuth:a} Vol A: \textsl{The \TeX\book}, 1986
\bibitem[6e]{knuth:e} Vol E: \textsl{Computer Modern Typefaces}, 1986
\end{thebibliography}

Here \texttt{lamport}, \texttt{knuth}, and \texttt{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 \texttt{bibitem}. The entry with the key \texttt{knuth} is the sixth in the list and thus it automatically receives the label [6]; in order for its sub-entries \texttt{knuth:a} ... \texttt{knuth:e} to be printed as [6a] ... [6e], it is necessary to set their optional \texttt{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 \texttt{thebibliography} environment at the desired location. For document classes \texttt{book} and \texttt{report}, the word \texttt{Bibliography} appears as an unnumbered chapter title at the beginning, whereas for \texttt{article} the word \texttt{References} is written as an unnumbered section heading. The above sample bibliography appears as:


[6a] Vol A: \textsl{The \TeX\book}, 1986

\textbf{Exercise 9.4:} Produce a bibliography with the \texttt{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 \textsc{Bib}{\TeX}

Bibliographic databases and the \textsc{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 bibliography is to appear; database is a list of all databases to be searched.

- Process the source file with \LaTeX{}; run \textsc{Bib}{\TeX} and then \LaTeX{} twice.

\textsc{Bib}{\TeX} writes a bibliography following the formatting specifications of \textit{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[\textit{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{\LaTeX\ 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{\LaTeX\ itself is explained in [1, pages 74, 75].}

9.3.4 \textbf{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 \cite{Jones et al., 1999} or textual, as shown by \textit{Jones et al.} \cite{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. Each has its own bibliography style (\texttt{.bst}) file to allow \texttt{\LaTeX\ 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 \texttt{.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 short is the short author list (e.g. Jones et al.) and 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 \TeX when a bibliography style is invoked with the \bibliographystyle command that is compatible with natbib. Three such .bst files are provided with natbib, as substitutes for 3 of the 4 basic \TeX styles: plainnat, unsrtnat, and abbrvnat (Section 14.1). However, many others also exist as contributions for specific publishers and journals.

Author-year citations

There is a flexible citation syntax permitting parenthetical (\citep) and textual (\citet) citations, with the abbreviated or full author list, with optional notes both before and after the citations. For example,

\citet{JON99} \Rightarrow Jones et al., (1999)
\citet[pg. 22]{JON99} \Rightarrow Jones et al., (1999, pg. 22)
\citep{JON99} \Rightarrow Jones et al., (1999)
\citep[pg. 22]{JON99} \Rightarrow Jones et al., (1999, pg. 22)
\citep[e.g.]{JON99} \Rightarrow (e.g. Jones et al., 1999)
\citep[e.g.][pg. 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)

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:

\citet{JON99,JAM01} \Rightarrow Jones et al. (1999); James et al. (2001)
\citep{JON99,JAM01} \Rightarrow (Jones et al., 1999; James et al., 2001)
\citep{JON99,JON00} \Rightarrow (Jones et al., 1999, 2000a)
\citep{JON00,JON00b} \Rightarrow (Jones et al., 2000a,b)
In numerical citation mode, the command \citet 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.

\citet{JON99} ⇒ Jones et al. [21]
\citet[pg. 22]{JON99} ⇒ Jones et al. [21, pg. 22]
\citet{JON99} ⇒ [21]
\citet[pg. 22]{JON99} ⇒ [21, pg. 22]
\citet[e.g.]{JON99} ⇒ [e.g. 21]
\citet[e.g.][pg. 22]{JON99} ⇒ [e.g. 21, pg. 22]
\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
put 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 \bibliography 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 \texttt{.bbl} files. Then the document is reprocessed with \LaTeX, at least twice; the \texttt{.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 \texttt{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 \texttt{\cbinput\{file\}} command. Set the package option \texttt{draft} when loading \texttt{chapterbib}, process with \LaTeX, process each file with \BibTeX, remove the \texttt{draft} option, process with \LaTeX at least twice.

With classes \texttt{book} and \texttt{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 \texttt{sectionbib} to change the bibliography to an unnumbered section instead.

The chapterbib package is compatible with natbib except for the \texttt{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\_entries \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 \texttt{\indexname} which may be redefined for other languages.) The individual entries are made with commands

\item \subitem \subsubitem and \texttt{\indexspace}

followed by the keywords and their page numbers. For example,

\begin{verbatim}
commands, 18
arguments, 19, 101
multiple, 103, 104
replacement symbol, 20
as environments, 42
used as arguments for sectioning commands, 41, 42
\end{verbatim}

\begin{verbatim}
displayed text, 21--32
\end{verbatim}

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 \texttt{\indexspace} leaves a blank line in the index.

9.4.2 Preparing the index entries

The \texttt{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

\begin{verbatim}
\index{index\_entry}
\end{verbatim}
where \textit{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 \textit{index\_entry}, such as \texttt{\LaTeX\ logo}. Even the one command that may otherwise never be used as an argument, \texttt{\verb}, may be included. However, if \textit{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

\begin{verbatim}
\makeindex
\end{verbatim}

in which case, a file with the document's root name plus the extension .idx is opened. Now the \texttt{\index} commands write \textit{index\_entry} and the current page number to this file in the form

\begin{verbatim}
\indexentry{index\_entry}{pagenumber}
\end{verbatim}

It is this .idx file that is the input to the MakeIndex program. There are special symbols that may be used in the \textit{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:

\begin{verbatim}
\index{main\_entry}
\index{main\_entry!sub\_entry}
\index{main\_entry!sub\_entry!sub\_sub\_entry}
\end{verbatim}

The individual main and sub-entries may contain any combination of characters with the exceptions of !, @, and _. 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 \textit{sub\_entry} will be assigned to a \texttt{\subitem} command underneath the corresponding \textit{main\_entry}. The texts for the \texttt{\subitem} will also be in alphabetical order. Similarly the text of a \textit{sub\_sub\_entry} will appear following a \texttt{\subsubitem} command, alphabetically ordered, below the appropriate \textit{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 \texttt{\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
eexample,

\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 read-
ability 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 MakIndex program can be run to produce the theindex environment from the data in the .idx file just as \TeX produces thebibliography environments. This tool is described in the next section.

The \LaTeX 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 MakIndex

The tiresome drudgery of making up the theindex environment for a keyword index can be dispensed with if the program MakIndex 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 MakIndex 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 or simply makeindex root_name
```

A subsequent \LaTeX 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 MakIndex 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 \"Ä, \ö, \ü, 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 MakelIndex 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 MakelIndex 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

\begin{verbatim}
\makeglossary
\glossary{glossary_entry}
\end{verbatim}

in the preamble and

\begin{verbatim}
\glossary{glossary_entry}
\end{verbatim}

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

\begin{verbatim}
\glossaryentry{glossary_entry}{pagenumber}
\end{verbatim}

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 pdftex 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.
Chapter 10. PostScript and PDF

Its two most significant features are the inclusion of graphics as encap-
sulated PostScript images, as described in Chapter 6, and the capability of
making use of PostScript fonts in place of the Computer Modern ones (Sec-
tion 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}

to process \texttt{myfile.dvi} and generate \texttt{myfile.ps}. (The \LaTeX source text
was of course in \texttt{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 \texttt{dvips} and the file name. For example, to output five
pages only starting at the 10th, sending them to a file named \texttt{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 \texttt{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 \texttt{dvips --help}.

One useful option is \texttt{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, \texttt{dvips} has another
way of accomplishing this: add the driver-specific command

\texttt{\special{landscape}}

in the source text itself and \texttt{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 \texttt{-o filename} is given. To send
the output directly to a printer, specify that printer as the output 'file', as
\texttt{-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.
10.1. LaTeX and PostScript

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 \texttt{-t a4} or \texttt{-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 \texttt{-P} plus extension. For example, config.cx contains lines for a generic PostScript printer with resolution of 300 dpi; with \texttt{-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{\textbackslash{}rmdefault}, \texttt{\textbackslash{}sfdefault}, \texttt{\textbackslash{}ttdefault} and the bold face default attribute \texttt{\textbackslash{}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\TeX\ to generate PDF directly. Other
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:

- \texttt{.tfm} the font metric files for both the virtual and raw fonts;

- \texttt{.fd} the NFSS font definition files associating the font attributes to precise virtual font names; these and the \texttt{.tfm} files are the only ones read by \LaTeX{} itself;

- \texttt{.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;

- \texttt{.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 pdftex and other programs that make use of PostScript fonts. It translates the \LaTeX name for the fonts into the internal names used by PostScript itself. For example, the line

\begin{verbatim}
ptmr8r Times-Roman "TeXBase1Encoding ReEncodeFont" <8r.enc
\end{verbatim}

saves that the font known to \LaTeX as ptm8r is really the \emph{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

\begin{verbatim}
hlhr8r LucidaBright "TeXBase1Encoding ReEncodeFont"
<8r.enc <hlhr8a.pfb
\end{verbatim}

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

\begin{verbatim}
p psfonts.map
\end{verbatim}

in config.ps, or individual mapping files are listed, as

\begin{verbatim}
p psnfss.map
p +pazo.map
p + ....
\end{verbatim}

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):

\begin{verbatim}
Encoding: OT1 Family: ptm Series: b Shape: sl
\end{verbatim}
Chapter 10. PostScript and PDF

The encoding and family indicate the font definition file that associates these attributes with real font names, in this case ot1ptm.fd. In that file, one finds this set of attributes to be assigned to font ptmbo7t; this is the Berry name for Times-BoldOblique, 7-bit-text (Tables G.2 and G.3 on pages 504 and 505).

Our sample input code is \AE; according to file ot1enc.def, this corresponds to character 29 in the OT1 encoding. \LaTeX thus writes instructions to the .dvi file to output the symbol in position 29 of font ptmbo7t. And with that, \LaTeX is finished with its part of the operation.

The next step is to process the .dvi file with the dvips driver. When this program searches for a font, it first looks in the map files, then it seeks the .vf virtual font files, and finally it tries to find it as a .pk pixel file. In our case, it discovers the virtual font ptmbo7t.vf, which tells the driver that character 29 is to be symbol 198 of font ptmbo8r. This is the same basic font but with 8-bit-raw encoding. Again dvips searches for this next font, and finds it in a mapping file, containing the line

\begin{verbatim}
ptmbo8r Times-Bold ".167 SlantFont TeXBase1Encoding
   ReEncodeFont" <8r.enc
\end{verbatim}

This tells the driver that it is to use the font with the internal name Times-Bold slanted by a factor 0.167 and encoded according to the TeXBase1Encoding encoding vector. The file 8r.enc must also be copied into the output.

Finally, the output .ps file is sent to a PostScript printer or other PostScript interpreter. The specified encoding vector indicates that character 198 is the symbol with the internal name AE, which is then printed from the Times-Bold font, distorted as required. The long road has come full circle.

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 `.tex \rightarrow .dvi \rightarrow .ps \rightarrow .pdf` using `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 `pdf\TeX` 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 `hyperref` package described in Section 10.2.4.

### 10.2.2 The `dvipdfm` driver

The driver program `dvipdfm` by Mark A. Wicks is the equivalent of `dvips` for PDF output. As its name implies, it converts DVI to PDF output. (The `m` suffix is to distinguish it from a previously announced `dvipdf` program that never came to fruition.)

Graphics input can be included with the `graphics` or `graphicx` packages (Section 6.1) by specifying the option `dvipdfm`. The allowed graphics formats are JPEG, PNG, and PDF.

The program is called from a command line with

```
dvipdfm myfile
```

to convert `myfile.dvi` to `myfile.pdf`. As for `dvips`, there are many options that may be included. It is best to read the provided manual in `dvipdfm.pdf` or `dvipdfm.dvi`, or produce it by processing `dvipdfm.tex` with \TeX.

The configuration is set up by means of a file named `config` in the `dvipdfm` directory. This is set up similarly to the `config.ps` file for `dvips`, including the list of mapping files, although these have a different syntax to those of `dvips`.

Hypertext features and other PDF additions can be added with the `hyperref` package (Section 10.2.4).
10.2.3 The pdfT\TeX program

The most direct method of getting PDF output is to process the L\TeX file with pdf\TeX, a variation on the T\TeX program developed by H\`an Th\`e Th\`anh. The following points should be noted:

- pdf\TeX behaves just like the regular T\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 T\TeX-like commands for including PDF features.

- Since pdf\TeX combines the functionality of both the T\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 T\TeX name for the font to various characteristics, encoding scheme, true name, and specify the file containing the character drawing instructions.

- New T\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 pdf\TeX by means of a \texttt{pdftex} graphics option.

To run pdf\TeX with L\TeX (often referred to as pdfL\TeX), it is necessary to generate a L\TeX format (Section B.1.3) using pdf\TeX instead of T\TeX, with the \texttt{-ini} option, renaming the resulting \texttt{latex.fmt} to \texttt{pdflatex.fmt}; it is then called with the command

\begin{verbatim}
pdftex &pdflatex
\end{verbatim}

Most installations contain a command \texttt{pdflatex} that translates to this combination. The editor shell should also have an icon that invokes this command.
0.2. Portable Document Format

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

Package: hyperref

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
Chapter 10. PostScript and PDF

Figure 10.1: Output produced by pdf\TeX\ with the hyperref package for automatic links, with an outline list at the left.

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
10.2. Portable Document Format

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:

\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:

\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:

dvipdf dvipdfm dvips dvipsone dviwindo hypertex latex2html pdftex tex4ht textures vtex
The option \texttt{hypertext} 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  (false) use Acrobat 4 features;

\texttt{anchorcolor}
  (black) set color of anchors (targets of internal links);

\texttt{backref}
  (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}
  determines the prefix for URLs given with \texttt{\href};

\texttt{bookmarks}
  (true) make bookmarks out of sectioning commands;

\texttt{bookmarksnumbered}
  (false) include section numbers in bookmark text;

\texttt{bookmarksopen}
  (false) start document with bookmarks tree opened;

\texttt{bookmarksopenlevel}
  (max) the level to which bookmarks are opened;

\texttt{bookmarkstype}
  (toc) specifies which 'contents' file used for bookmarks;

\texttt{breaklinks}
  (false) allows link text to break over lines;
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;

filecolor
(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;

hypertexnames
(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;
Chapter 10. PostScript and PDF

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
   \( (0 \ 0 \ 1, \text{but } 0 \ 0 \ 0 \text{ with colorlinks}) \) 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;

urlcolor
(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 \texttt{baseurl} option, but can be changed within the document;

\href{url}{text}
makes \textit{text} into a link to the full URL given by \textit{url} prefixed with the current base URL; special characters like \# and \& can be type in \textit{url} directly;

\hyperimage{url}{image.url}
inserts the image referenced by \textit{image.url};

\hyperdef{category}{name}{text}
establishes \textit{text} as a target (anchor) with name \textit{category.name};

\hyperref{url}{category}{name}{text}
makes \textit{text} a link to \texttt{url\#category.name};

an alternate syntax is \hyperref[\textit{key}]{text} to make \textit{text} an internal link to the location marked with \texttt{\label{\textit{key}}};

\hypertarget{name}{text}
establishes \textit{text} as an internal target (anchor) with the name \#\textit{name};

\hyperlink{name}{text}
makes \textit{text} a link to the internal target \#\textit{name}.

The cross-referencing commands \texttt{\ref{\textit{key}}} and \texttt{\pageref{\textit{key}}} (Section 9.2.1) automatically create links to the corresponding \texttt{\label{\textit{key}}} location. As pointed out above, one can also create a manual link with \hyperref[\textit{key}]{text}.

The \hyperref package also provides starred versions \texttt{\ref*} and \texttt{\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 \Acrobatmenu</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).

```latex
\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 • 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{IL\LaTeX}, 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 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{babe1}.
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 babel files**

The babel installation provides the following files for its implementation:

- **babel.def** contains some of the basic macros for running babel, and must be loaded by the first language file read in; the remaining macros are either already in the format itself (loaded from *hyphen.cfg*) or read in from *switch.def*; *babel.def* determines which is the case and reacts accordingly;

- **switch.def** contains the additional babel macros, to be read in if these macros are not already stored in the \LaTeX{} format;

- **hyphen.cfg** contains the same macro definitions as in *switch.def*, plus some more that are to be run by 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 *switch.def* is not necessary at runtime;

- **babel.sty** is a master package that loads the language files specified as options;

- **language.dat** contains a list of languages and the file names of their hyphenation patterns; this file is read in by *hyphen.cfg* during the initex run; it is the only file that may (must!) be edited for the particular installation (Section 11.2);

- **esperant.ldf**... the language definition files.

- **esperant.sty**... compatibility files that read in the .ldf files.

**Invoking babel**

The babel package is loaded with the desired languages as options:

```
\usepackage[english,esperanto]{babel}
```

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

```
\documentclass[english,esperanto]{article}
\usepackage{babel,varioref}
```

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:
Language switching commands

The normal user needs to know very little about the \texttt{babel} internal operations. The new high-level user commands are:

```latex
\selectlanguage{language}
\begin{otherlanguage}{language} text \end{otherlanguage}
\begin{otherlanguage*}{language} text \end{otherlanguage*}
\foreignlanguage{language}{text}
\iflanguage{language}{yes_text}{no_text}
```

The \texttt{\selectlanguage} command and the \texttt{otherlanguage} environment are two ways to switch to another language, with all its features and translations. The \texttt{\foreignlanguage} command and the \texttt{otherlanguage*} environment also switch to \texttt{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 \texttt{\iflanguage} executes \texttt{yes_text} if \texttt{language} is current otherwise \texttt{no_text}.

In addition, \texttt{\language} 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,
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 special 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 \1@language; for example, \1@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 hyphenation patterns. For example, in english.ldf there is the command \adddialect{\1@american}{\1@english}.

A language definition file must provide four commands:

\captionslang to redefine the naming commands like \tablename (see Section D.4.1);
\date{lang} 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 \1@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\1@lang to select the set of hyphenation patterns,
• invoking \originalTeX in order to remove any existing language-specific commands,
• activating \captionslang, \date{lang}, 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 lang.

11.2 Contents of the 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 \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 babel system invokes a more reliable procedure.

During the production of the format by initex (Section B.1.3), the file hyphen.cfg is input, which in turn reads in 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 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 ushyphen.tex, ukhyphen.tex, frhyphen.tex, dehyphn.tex, and dehyphn.tex are loaded under the language numbers 0 to 4 respectively, and \l@american, \l@english, \l@french, \l@german, and \l@ngerman are defined as numbers 0 through 4 for use with the \selectlanguage command. Languages \l@USenglish and \l@UKenglish are synonyms for the current language, in this case 0 and 1, respectively. Similarly \l@british is identical to \l@UKenglish.
12

Math Extensions with \AMS-LATEX

The American Mathematical Society, \AMS, has supported the development and usage of \TeX since its first release. It in fact owns the \TeX logo as a registered trademark.

Shortly after \TeX 82 became available, the \AMS produced a macro package for generating a special format \amstex, described in \textit{The Joy of \TeX} by Spivak (1986). The macros in this \AMS-\TeX complement the mathematical typesetting features of Plain \TeX by adding additional ones and simplifying others.

However, \AMS-\TeX is not a documentation preparation system like \LaTeX, describing the logical layout of a document by means of markup commands, but rather is simply an extension to Plain \TeX.

The great popularity of \TeX as a text formatting program is due primarily to the availability of \LaTeX as a user-friendly interface to the underlying \TeX machinery. Many authors therefore asked the \AMS to provide \LaTeX with the same mathematical features as in \AMS-\TeX. The \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 \AMS. \AMS-\LaTeX has been fully converted to \LaTeX 2\epsilon with version 1.2 in 1996.

The \AMS-\LaTeX collection consists of three parts: packages for extending mathematical typesetting, extra classes for articles and books published by the \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 \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 \AMS-supplied manual found in the file \texttt{amsfndoc.tex}. The extra classes will not be described in this book; \AMS authors should refer to the instructions in the document \texttt{instr-1.tex} provided with \AMS-\LaTeX.
12.1 Invoking \textit{AMSL\-TEX}

If the \texttt{\documentclass} statement at the beginning of the L\-TEX document selects one of the AM\-S classes amsbook, amsart, or amsproc, then most of the AM\-S-L\-TEX features are loaded automatically. These features may still be employed with other classes by including the main extension package 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 amsmath package defines many of the new math typesetting features itself, but it also loads a number of other packages from the AM\-S-L\-TEX collection that contain further extensions, such as amsopn, amstext, and amsbsy. These packages could be loaded separately without amsmath if only their limited features are wanted. On the other hand, the packages amscd and amsthm are not included in 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 amsmath and its associated packages. We call these the standard features of AM\-S-L\-TEX. Further extensions added by other packages are explained afterwards.

The examples in this chapter employ the user-defined commands \texttt{\\textbackslash mi}, \texttt{\textbackslash me}, and \texttt{\textbackslash dif} defined in Section 5.4.10 for printing upright i, e, and d in math mode.

12.2 Standard features of \textit{AMSL\-TEX}

This relatively long section presents those standard features of AM\-S-L\-TEX that are activated by loading the amsmath package, or by selecting one of the AM\-S classes amsart, amsbook, or amsproc.

12.2.1 Additional font switching commands

Standard L\-TEX provides the math alphabet commands \texttt{\textbackslash mathcal}, \texttt{\textbackslash mathrm}, \texttt{\textbackslash mathbf}, \texttt{\textbackslash mathsf}, \texttt{\textbackslash mathit}, \texttt{\textbackslash mathbbt} for changing fonts within math mode (Section 5.4.2). With AM\-S-L\-TEX, one may also use the commands

\begin{verbatim}
\boldsymbol{symbol} \quad \texttt{\textbackslash pmb{symbol}}
\end{verbatim}

to print \textit{symbol} in a bold face, provided there is an appropriate bold font for it. Whereas the command \texttt{\textbackslash 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 $\mathbf{\nabla \times V d\sigma}$ with that
12.2. Standard features of $\textit{AMS-LATeX}$

of $\texttt{\textbackslash boldsymbol{\nabla \times V \, d\sigma}}$. With the standard $\texttt{\LaTeX}$ command $\texttt{\textbackslash mathbf}$, 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 $\texttt{\textbackslash boldsymbol}$, all symbols are bold and italic.

Those symbols for which no bold face font exists will remain in normal weight with the $\texttt{\textbackslash boldsymbol}$ command. Such symbols are, for example, those that come in two sizes (Section 5.3.7) like $\sum$, $\int$, $\bigcup$, and so on. The command $\texttt{\textbackslash pmb}$ (poor man’s bold) simulates a bold face even for these symbols by printing them several times slightly displaced. The result of $\texttt{\textbackslash pmb{\sum \int \bigcup}}$ 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{\text{short\_text}}

In contrast to $\texttt{\mbox{short\_text}}$ for standard $\texttt{\LaTeX}$, the $\texttt{\text}$ command switches font size correctly when used as superscripts and subscripts. Thus $\ldots\{\text{Word}\}$ sets $\text{Word}$ lower and changes to $\text{\scriptstyle Word}$. To achieve the same effect in standard $\texttt{\LaTeX}$, the font size must be given explicitly, as $\ldots\{\texttt{\mbox{\scriptstyle Word}}\}$. 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{insert\_text}}

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:

$$(x + iy)(x - iy) = x^2 - ixy + ixy - i^2 y^2 = y^2 + y^2 \text{ since } i^2 = -1 \text{ is true.}$$

On the other hand

$$(x + iy)^2 = x^2 + 2ixy + i^2 y^2 = x^2 + 2ixy - y^2$$

$$(x - iy)^2 = x^2 - 2ixy + i^2 y^2 = x^2 - 2ixy - y^2$$

\begin{align*}
(x + \mi y)(x - \mi y) &= x^2 - \mi xy + \mi xy - \mi \cdot 2y^2 \quad \\
&= y^2 + y^2 \quad \text{since} \quad \mi \cdot 2 = -1 \\
\end{align*}
\intertext{On the other hand}
\begin{align*}
(x + \mi y)^2 & = x^2 + 2\mi xy + \mi^2 y^2 = x^2 + 2\mi xy - y^2 \\
(x - \mi y)^2 & = x^2 - 2\mi xy + \mi^2 y^2 = x^2 - 2\mi 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 \texttt{align} (Section 12.2.6) is one of several new ones provided by \texttt{AMS-\LaTeX} to replace the standard \LaTeX environment \texttt{eqnarray}. Note that \texttt{\intertext} may only be issued immediately after the $\\$ for starting a new line.

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 \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.

Multiple integrals

With \texttt{AMS-\LaTeX}, the commands $\iint$, $\iiint$, $\iiiint$, and $\idotsint$ output multiple integrals, with upper and lower limits being added in the usual manner. In text formulas, they are printed as $\int\int$, $\int\int\int$, $\int\int\int\int$, $\int\cdots\int$, while in displayed formulas, they appear as

\begin{align*}
\iint_{S} f(x, y) \, dS & \quad \%[ \iint\limits_{S} f(x, y) \, dS ] \\
\iiint_{V} f(x, y, z) \, dV & \quad \%[ \iiint\limits_{V} f(x, y, z) \, dV ] \\
\iiiint_{G} f(x, y, z, t) \, dG & \quad \%[ \iiiint\limits_{G} f(x, y, z, t) \, dG ] \\
\idotsint_{U} f(x_1, \ldots, x_k) \, dU & \quad \%[ \idotsint\limits_{U} f(x_1, \ldots, x_k) \, dU ]
\end{align*}

whereby the command $\texttt{\limits}$ may be left off if the option $\texttt{intslimits}$ has been specified when $\texttt{amsmath}$ was loaded (see Section 12.2.8).
Multiline limits

One often needs multiline limits or indices for summations, as in the examples below.

\[
\Delta_{p_1 p_2 \cdots p_{n-k}} \sum_{k_0, k_1, \ldots \ge 0 \atop k_0 + k_1 + \cdots = 0} a_{0k_0} a_{1k_1} \cdots
\]

\( A_{\text{AMS-\LaTeX}} \) provides a command \texttt{\substack} for this purpose, with syntax

\texttt{\substack{1st line \backslash 2nd line \backslash \ldots \backslash last line}}

where the command must immediately follow the \^ or \_ shifting commands, and be entirely enclosed in curly braces \{ \}. The index for the left-hand example above was generated with

\[
\Delta_{p_1 p_2 \cdots p_{n-k}} \quad \text{\texttt{\Delta_\{\substack{p_1 p_2 \cdots p_{n-k} \\cdots q_{n-k}}}}
\]

The lines of text printed by \texttt{\substack} are centered horizontally, as is apparent in the right-hand example above. This was produced with

\[
\sum_{\begin{subarray}{l} k_0, k_1, \ldots \ge 0 \\atop k_0 + k_1 + \cdots = 0 \end{subarray}} a_{0k_0} a_{1k_1} \cdots
\]

On the other hand, the environment \texttt{\subarray} offers more control over the horizontal alignment:

\[
\begin{subarray}{pos} \text{1st line} \backslash \text{2nd line} \backslash \ldots \backslash \text{last line}
\end{subarray}
\]

The argument \texttt{pos} may be \texttt{c} for centered, or \texttt{l} for left-justified lines.

\[
\sum_{i \in \Lambda \atop i < j < n} P(i, j)
\]

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

\[
\sum' E_n
\]

However, if there to be an upper limit as well, the prime can only be added with much fiddling. The \texttt{\sideset} command simplifies this task considerably.

\[
\sum'_{n=0} (2n+1)E_{2n+1} \quad \text{\texttt{\sum\{n=0\}'infty (2n+1) E_{2n+1}}}
\]

The complete syntax for this command is:
\sideset{\text{pre}}{\text{post}}\symbol

where \text{pre} and \text{post} are the superscripts and subscript commands to be added before and after the \symbol, respectively. They must contain raising and lowering operators \^ and _.

The product symbol \( \prod \) is given daggers above and asterisks below with \sideset{\text{\_dag}^*}{\text{\_dag}^*}\prod, as shown \( *\prod^* \) at the right.

Additional shifting commands are

\overset{\text{char}}{\symbol} \quad \text{and} \quad \underset{\text{char}}{\symbol}

which place the arbitrary \text{char} above or below the \symbol in the size appropriate for superscripts and subscripts. Thus $\overset{*}{X}$ produces \( X \) and $\underset{*}{X}$ yields \( \_X \).

Extended arrows

The \texttt{amsmath} package provides a number of commands to produce extra long arrows for combining with mathematical expressions. The commands

\begin{verbatim}
\overleftarrow{expr} \hspace{1cm} \underleftarrow{expr} \\
\overrightarrow{expr} \hspace{1cm} \underrightarrow{expr} \\
\overleftrightarrow{expr} \hspace{1cm} \underleftrightarrow{expr}
\end{verbatim}

produce lengthened arrows pointing left and right, as well as double arrows above and below the mathematical expression \texttt{expr}.

\begin{verbatim}
\begin{eqnarray*}
\overrightarrow{ABCD} &=& AB + BC + CD \\
\underleftarrow{ABCD} &=& AB + \underleftarrow{BC} + \underleftarrow{CD}
\end{eqnarray*}
\end{verbatim}

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:

\begin{verbatim}
\underrightarrow{\rule[-2pt]{0pt}{2pt}AB} \ldots \text{ and } \\
\underleftarrow{\rule[-2pt]{0pt}{2pt}DC} \ldots
\end{verbatim}

A smaller font size will be used for the arrows when they appear in exponents, indices, superscripts, and subscripts:
\int_{0.2\pi} r \, dq = 2\pi r

There are two more commands for horizontal arrows of variable length:
\xleftarrow{below}{above} \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 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 $\check{A}$, $\breve{A}$, $\acute{A}$, $\grave{A}$, $\tilde{A}$, $\bar{A}$, $\vec{A}$, $\dot{A}$, and $\ddot{A}$ (Section 5.3.9). The amsmath package provides a set of math accent commands with the same names but capitalized,

$\Hat{\Hat{A}}$, $\Breve{\Bar{B}}$, 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

\[ \dddot{sym} \quad \text{and} \quad \ddddot{sym} \]
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. \AmS-\LaTeX 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}.
264 Chapter 12. Math Extensions with $A_M S-L_A T_E X$

\[ a_0 + a_1 + \ldots + a_n \Rightarrow a_0 + a_1 + \cdots + a_n \]
\[ a_0, a_1, \ldots, a_n \Rightarrow a_0, a_1, \ldots, a_n \]

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 \dotsc (comma), \dotsb (binary), \dotsm (multiplication), or \dotsi (integral).

The \dotsc is to be used with commas, so $A_1, A_2, \dotsc$ produces $A_1, A_2, \ldots$; the \dotsb sets them for a binary operator, thus $A_1 + A_2 + \dotsb$ yields $A_1 + A_2 + \cdots$; the command \dotsm is actually identical to \dotsb, but it is logically meant to be applied to multiplication, $A_1 A_2 \dotsm$ makes $A_1 A_2 \cdots$; finally \dotsi places the dots at the mean height of an adjacent integral sign.

\[
\int_{A_1} \int_{A_2} \ldots
\]

12.2.3 Fractions

The $T_X E$ fraction commands \atop, \choose, and others may be allowed in standard $L_A T_E X$, but not in $A_M S-L_A T_E X$. With the amsmath package, only those fraction commands described here may be used.

Basic fraction commands

In addition to the regular $L_A T_E X$ command \frac{over}{under}, $A_M S-L_A T_E X$ provides the commands \tfrac and \dfrac with the same syntax. These are effectively the same as \frac but with the font set to \textstyle or \displaystyle, respectively (Section 5.5.2). We demonstrate their effects with some examples from the $A_M S-L_A T_E X$ user's manual.

\[
\frac{1}{k} \log_2 c(f) \quad \tfrac{1}{k} \log_2 c(f) \quad \sqrt{\frac{1}{k} \log_2 c(f)} \quad \sqrt{\dfrac{1}{k} \log_2 c(f)}
\]

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 amsmath package is

\[ \binom{over}{under} \]
12.2. Standard features of \textsc{AMS-L\TeX}

which functions in the same way as \texttt{\textbackslash frac} and the other fraction commands.

\[
\[ \binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1} \]

Similarly there are the commands \texttt{\textbackslash tbinom} and \texttt{\textbackslash dbinom} analogous to \texttt{\textbackslash frac} and \texttt{\textbackslash dfrac}.

**User-defined fractions**

The \textit{amsmath} package provides a powerful tool for defining fraction-like structures:

\[
\genfrac{\text{left_brk}}{\text{right_brk}}{\text{thickness}}{\text{mathsize}}{\text{over}}{\text{under}}
\]

where \texttt{left_brk} and \texttt{right_brk} 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{\textbackslash displaystyle}, \texttt{\textbackslash textstyle}, \texttt{\textbackslash scriptstyle}, and \texttt{\textbackslash 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{\textbackslash frac} and \texttt{\textbackslash 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}:

\[
\newcommand{\frac}[2]{\genfrac{}{}{}{}{#1}{#2}}
\newcommand{\dfrac}[2]{\genfrac{}{}{}{0}{#1}{#2}}
\newcommand{\tfrac}[2]{\genfrac{}{}{}{1}{#1}{#2}}
\newcommand{\binom}[2]{\genfrac{(}{)}{0pt}{}{#1}{#2}}
\]

As a further example, consider the redefinition of the command \texttt{\frac}

\[
\renewcommand{\frac}[3][3]{\genfrac{}{}{}{}{#1}{#2}{#3}}
\]

in which the line thickness is now an optional first argument; without this optional argument, the command behaves as normal. Thus

\[
\[ \binom{n}{m} = \frac{\! n!}{M!(n-m)!} \]
\]

yields

\[ \binom{n}{m} = \frac{n!}{M!(n-m)!} \]

**Continued fractions**

Continued fractions can be made in \textsc{AMS-L\TeX} with the command

\[
\cfrac{\text{pos}}{\text{over}}{\text{under}}
\]

whereby the denominator \texttt{under} may contain further \texttt{\cfrac} commands.
produces

\[
a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4 + \cdots}}}}
\]

If the optional argument \texttt{pos} is missing, the numerator \texttt{over} is centered on the horizontal rule; otherwise it may take values of \texttt{l} or \texttt{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.

\[
\sum_{p_1 < p_2 < \ldots < p_{n-k}} \Delta_{(1,2,\ldots,n)} \sum_{q_1 < q_2 < \ldots < q_k} \begin{vmatrix}
a_{q_1q_1} & a_{q_1q_2} & \ldots & a_{q_1q_k} \\
a_{q_2q_1} & a_{q_2q_2} & \ldots & a_{q_2q_k} \\
\ldots & \ldots & \ldots & \ldots \\
a_{q_kq_1} & a_{q_kq_2} & \ldots & a_{q_kq_k}
\end{vmatrix}
\]

\[
\sum_{p_1 < p_2 < \ldots < p_{n-k}} \Delta_{(1,2,\ldots,n)} \sum_{q_1 < q_2 < \ldots < q_k}
\begin{vmatrix}
a_{q_1q_1} & a_{q_1q_2} & \ldots & a_{q_1q_k} \\
a_{q_2q_1} & a_{q_2q_2} & \ldots & a_{q_2q_k} \\
\ldots & \ldots & \ldots & \ldots \\
a_{q_kq_1} & a_{q_kq_2} & \ldots & a_{q_kq_k}
\end{vmatrix}
\]
\hdotsfor[2.0]{4}\
a_{q_kq_1} & a_{q_kq_2} & \ldots & a_{q_kq_k} \\
\end{vmatrix} \\
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[stretch]{n}
and which prints a continuous line of dots through \( n \) columns. The optional argument \texttt{stretch} is a multiplicative number to increase the dot density, being 1.0 by default.
\begin{matrix} a & b & c & d & e \\
x & \hdotsfor{3} & z \\
\end{matrix}
Compare the standard dot spacing above with that from \hdotsfor[2.0] in the previous example.
The initial letter of each of the \texttt{xmatrix} environments indicates the type of braces that enclose it: \texttt{pmatrix} for (round) parentheses, \texttt{bmatrix} for (square) brackets, \texttt{Bmatrix} for (curly) braces, \texttt{vmatrix} for vertical lines, and \texttt{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{cases} r & s & t \\ u & v & w \\ x & y & z \end{cases}
where \texttt{xmatrix} is set to \texttt{matrix}, \texttt{pmatrix}, \texttt{bmatrix}, \texttt{Bmatrix}, \texttt{vmatrix}, and \texttt{Vmatrix} one after the other.
To generate a small array within a text formula, one can apply the \texttt{smallmatrix} environment. In this way \((a \ b \ c\ e \ m \ r)\) 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

\textbf{Package: 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, arcsin, 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.

\textit{AMS-\LaTeX} provides some more function names, as variations on the standard \textbackslash lim name:

\begin{verbatim}
\varlimsup \lim \varliminf \lim
\varinjlim \lim \varprojlim \lim
\end{verbatim}

These functions may take on limits with the raising and lowering operators ^ and _; for example \textbackslash 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 \textbackslash begin\{document\}, defines a command \textbackslash cmd that prints the function name \textit{name}. For example, to define a function name \textbackslash doit, give

\begin{verbatim}
\DeclareMathOperator{\doit}{doit}
\end{verbatim}

and then $A=3\doit^2(B)$ yields $A = 3 \doit^2(B)$. 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}

The \textit{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 \textbackslash bmod and \textbackslash pmod commands, and are complemented in \textit{AMS-\LaTeX} by \textbackslash mod and \textbackslash pod. The possibilities are:

\begin{verbatim}
z \equiv x + y \bmod n^2
z \equiv x + y \pmod{n^2}
z \equiv x + y \mod n^2
z \equiv x + y \pod{n^2}
\end{verbatim}

The automatic parentheses are missing with \textbackslash mod, while with \textbackslash pod the name 'mod' is omitted. Furthermore, \textbackslash pmod is redefined for text formulas to reduce the preceding space: $y \pmod{a+b}$: $y \pmod{a+b}$.

The \textbackslash DeclareMathOperator command and additional function names are defined in the \textit{amsopn} package, which may be loaded on its own without amsmath.
Fine-tuning roots

The positioning of an index to a root sign is not always ideal under standard \LaTeX. In $\sqrt{k}$, for example, the $\beta$ could be somewhat higher and shifted slightly to the right. The $\texttt{AMS-\LaTeX}$ commands

\begin{verbatim}
\leftroot{shift} \uproot{shift}
\end{verbatim}

cause such manual displacements, where \texttt{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 $\sqrt[\leftroot{-1}\uproot{3}\beta]{k}$:

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 \texttt{\smash} which places its argument in a box of zero height and depth. The $\texttt{AMS-\LaTeX}$ version of this command allows an optional argument \texttt{b} or \texttt{t} to zero only the depth or height, respectively. The above example is produced with $\sqrt{\smash[b]{y}}$. The option \texttt{b} is taken because we want only the depth to be ignored, not the height of the letter \texttt{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 $\backslash$, $\backslash$; $\backslash$quad and $\backslash$qquad for increasing amounts of positive spacing, and $\backslash$! for negative spacing. With the amsmath package, the first three still exist, but may also be called with the more obvious names $\backslash$thinspace, $\backslash$medspace, and $\backslash$thickspace. There is also $\backslash$negthinspace as an alias for $\backslash$!.

The complete set of spacing commands are summarized in the table below taken from the $\texttt{AMS-\LaTeX}$ 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>$\backslash$</td>
<td>$\texttt{thinspace}$</td>
<td>$___$</td>
<td>$!$</td>
<td>$\texttt{negthinspace}$</td>
<td>$__$</td>
</tr>
<tr>
<td>$\backslash$</td>
<td>$\texttt{medspace}$</td>
<td>$____$</td>
<td></td>
<td>$\texttt{negmedspace}$</td>
<td>$____$</td>
</tr>
<tr>
<td>$\backslash$;</td>
<td>$\texttt{thickspace}$</td>
<td>$_____$</td>
<td></td>
<td>$\texttt{negthickspace}$</td>
<td>$_____$</td>
</tr>
<tr>
<td>$\backslash$quad</td>
<td></td>
<td>$___$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\backslash$qquad</td>
<td></td>
<td>$____$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The general math spacing command is

$\backslash$mspace{$\mu$}

which inserts space in mathematical spacing units ‘mu’ (=1/18 em). For example, $\backslash$mspace{$-9\mu$} puts in negative spacing of 1/2 em.
Vertical bars

In standard \LaTeX, the commands | and \| are used for single and double vertical bars, | and ||. 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 ||\nu||. The \LaTeX commands are only appropriate for single appearances, like p|q or f(t,x)|_{t=0}.

The amsmath package defines the delimiter commands \lvert, \rvert for a single bar, and \lVert, \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 $\norm{\nu}$ ||\nu||.

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 ⟨x⟩, rather than $<x>$ which produces <x>.

Boxed formulas

A formula may be placed in a box with the command

\boxed{formula}

For example,

\begin{verbatim}
\boxed{\int_0^{\infty} f(x) \approx \sum_{i=1}^{n} w_i e^{x_i} f(x_i)}
\end{verbatim}

produces

\[
\int_0^{\infty} f(x) \, dx \approx \sum_{i=1}^{n} w_i e^{x_i} f(x_i)
\]

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 eqnarray and eqnarray* environments (Section 5.4.7). Many authors consider these to be far too limited for publications with complicated multiline equations. \AMS-L\LaTeX therefore provides a range of further alignment environments for formulas extending over a single line:

align gather falign multline alignat split
With the exception of \texttt{split}, all exist in a standard and a *-form. As for \texttt{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-LaTeX in a *-form. It may be used in combination with multiline environments.

\section*{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 \texttt{split} which must be called in math mode. A new line is forced in the formula with the \texttt{\\} command, as usual; an optional argument \texttt{\[\text{len}\]} can be added to increase the line spacing by \texttt{len}, again as usual.

The automatic numbering with the standard forms can be suppressed for single lines by adding \texttt{\notag} before the \texttt{\\} line break. Alternatively, the line can be given a desired marker with \texttt{\tag{\textit{mark}}}. For example, with \texttt{\tag{$\dag$}}, the marker is (†). 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

\begin{verbatim}
\raisetag{\textit{len}}
\end{verbatim}

which moves the marker upwards by \textit{len} for that line only. A negative value moves it downwards.

\section*{The \texttt{multline} environment}

The \texttt{multline} environment is a variant of the \texttt{equation} environment for \textit{single} formulas that are too long for one line. The line breaks occur where the user forces them with the \texttt{\\} command. The first line is left justified, the last right justified, and lines in between are centered. However, if the option \texttt{fleqn} has been given, all the lines appear left justified.

The equation number, if present, appears at the right of the \textit{last} line by default or if the option \texttt{reqno} has been selected; if the option \texttt{leqno} has been chosen, the number is placed at the left of the \textit{first} line. (See Section 12.2.8 for the \texttt{amsmath} options.)

It is possible to shift individual lines fully to the left or right with the commands \texttt{\shoveleft{\textit{formula}}} and \texttt{\shoveright{\textit{formula}}}. The entire formula text for that line, except the terminating \texttt{\\}, is placed in their arguments.

The left and right margins for the formula are set by the length parameter \texttt{\multlinegap} which is initially 10 pt. This may be altered by the user with the \texttt{\setlength} or \texttt{\addtolength} commands.
An equation with five lines could be broken to look as follows:

\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{multline}, the \texttt{split} environment is meant for a single equation that does not fit on one line. Line breaks are again forced with the \texttt{\textbackslash \textbackslash} 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 \texttt{multline} 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.)

\[ H_c = \frac{1}{2n} \sum_{l=0}^{n} (-1)^l (k-l)^{p-2} \sum_{l_1+\ldots+l_p=l} \prod_{i=1}^{p} \binom{n_i}{l_i} \times \left[ (k-l) - (k_i-l_i) \right]^{k_i-l_i} \times \left[ (k-l)^2 - \sum_{j=1}^{p} (k_i - l_i)^2 \right] \] (12.2)
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{equation}
\begin{split}
H_c&=\frac{1}{2n}\sum_{l=0}^{n} (-1)^l (k-l)^{p-2} \\
&\quad \prod_{i=1}^{p} \binom{n_i}{l_i} \times \left[(k-l) - (k_i-l_i)\right]^{k_i-l_i} \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.

\begin{align}
\frac{1}{2} + \left(\frac{2}{3}\right)^4 + \left(\frac{3}{4}\right)^9 + \cdots + \left(\frac{n}{n+1}\right)^{n^2} + \cdots &= \sum_{n=1}^{\infty} \left(\frac{n}{n+1}\right)^{n^2} \tag{12.3} \\
\text{converges since} \quad \lim_{n\to\infty} n\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 \\
2 + \frac{3}{4} + \frac{4}{9} + \cdots + \frac{n+1}{n^2} + \cdots &= \sum_{n=1}^{\infty} \frac{n+1}{n^2} \tag{12.4} \\
\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{align}
\dif x = \left[ \ln x - \frac{1}{x} \right]_c^\infty = \infty

Note the use of \tag and \notag (page 271) to add text as markers to the second and fourth lines.

The align environment

The 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 array environment with column specification \{rl rl rl \ldots\}.

\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 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
12.2. Standard features of \LaTeX

double && so that the left-hand sides of these column pairs are empty: the equals signs are always on an odd-numbered alignment marker.

\begin{align}
V &= \int_V \dif v &= \iiint \dif x \, \dif y \, \dif z & = \iiint \rho \, \dif x \, \dif \rho \, \dif \phi \\
&= \iiint r^2 \sin \theta \, \dif r \, \dif \theta \, \dif \phi & (12.5) \\
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 \\
&= \iiint r^4 \sin^3 \theta \, \dif r \, \dif \theta \, \dif \phi & (12.6) \\
M &= \int_V \delta \, \dif v &= \iiint \delta \, \dif x \, \dif y \, \dif z \\
&= \iiint \delta \rho \, \dif z \, \dif \rho \, \dif \phi \\
&= \iiint \delta r^2 \sin \theta \, \dif r \, \dif \theta \, \dif \phi & (12.7)
\end{align}

There are two variations on the align environment, falign and 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 &= \iiint \dif x \, \dif y \, \dif z \\
I_z &= \int \rho^2 \, \dif v &= \iiint (x^2 + y^2) \, \dif x \, \dif y \, \dif z \\
M &= \int \delta \, \dif v &= \iiint \delta \, \dif x \, \dif y \, \dif z
\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 \texttt{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 \texttt{split} environment is to be placed inside an \texttt{equation} environment. The same is true for the environments \texttt{aligned} and \texttt{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 \textit{pos}

\begin{verbatim}
\begin{aligned}
\end{aligned}
\begin{gathered}
\end{gathered}
\end{verbatim}

which takes values of \texttt{t} or \texttt{b} to determine the vertical alignment (top or bottom) when they appear beside other elements. When no \textit{pos} is given, they are centered. In this way

\[
\begin{aligned}
\alpha &= aa \\
\beta &= bbbbb \\
\gamma &= g \\
\delta &= dd \\
\eta &= eeeeee \\
\varphi &= f
\end{aligned}
\]

versus

\[
\begin{aligned}
\delta &= dd \\
\eta &= eeeeee \\
\varphi &= f
\end{aligned}
\]

is produced with

\[
\begin{equation*}
\begin{aligned}
\alpha &= aa \\
\beta &= bbbbb \\
\gamma &= g \\
\delta &= dd \\
\eta &= eeeeee \\
\varphi &= f
\end{aligned}
\end{equation*}
\]

**The cases environment**

Although it is possible with standard \LaTeX{} to produce structures of the form

\[
P_{r-j} = \begin{cases} 
0 & \text{if } r - j \text{ is odd,} \\
\frac{r!}{(r-j)!} (-1)^{r-j}/2 & \text{if } r - j \text{ is even.}
\end{cases}
\]  

(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!( -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:

\begin{equation}
y = \begin{cases}
-1 & : x < 0 \\
0 & : x = 0 \\
+1 & : x > 0
\end{cases}
\end{equation}

### 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

\begin{verbatim}
\numberwithin{equation}{section}
\end{verbatim}

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{\thesection} 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{verbatim}
\begin{subequations} ...
\end{subequations}
\end{verbatim}
Chapter 12. Math Extensions with $\text{AMS-}\LaTeX$

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 $\text{\label{\{marker\}}}$ is issued in a mathematical formula that receives an automatic equation number, that number can be printed anywhere in the text with $\text{\ref{\{marker\}}}$, where \textit{marker} is arbitrary text to identify that equation.

The amsmath package adds a command $\text{\eqref{\{marker\}}}$ 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 $\ref$, or as equation (12.8) with $\eqref$.

If $\text{\label}$ is given immediately after the start of a subequations environment, the corresponding $\ref$ commands will print the main equation number without the extra letter. In this way one can refer to the entire group of equations. Later $\text{\label}$ commands are associated with individual equation lines and reference them with the letters.

Page breaks within multiline formulas

Unlike the standard $\LaTeX$ 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

\begin{displaybreak}[num]
\end{displaybreak}

just before the line breaking command $. \text{Here the optional } num \text{ has the same meaning as for the standard } \text{\pagebreak} \text{ command (page 33): without it, a new page is forced, but it may take values of 0–4 to allow a break with increasing degree of encouragement, whereby 4 also forces the page break.}$

Alternatively, one can issue $\text{allowdisplaybreaks}$ in the document preamble to allow $\LaTeX$ to break pages automatically within multiline
12.2. Standard features of $\text{AMSL}$-\TeX

formulas as necessary. This command too takes an optional argument \texttt{num} with possible values between 0 and 4 which make it progressively easier for automatic page breaks to occur.

Once \texttt{\allowdisplaybreaks} has been given in the preamble, it is still possible to suppress page breaks within a formula by ending an equation line with \texttt{\*} instead of with \texttt{\}.

12.2.8 Package options for \texttt{amsmath}

The main \texttt{amsmath} package for $\text{AMS}\text{-}\LaTeX$ recognizes a number of options that may be given when it is loaded with

\texttt{\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.

\texttt{centertags} | \texttt{tbtags} The equation number for a \texttt{split} environment (page 272) is centered vertically by default. With \texttt{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.

\texttt{sumlimits} | \texttt{nosumlimits} In displayed formulas, initial and final limits appear below and above the $\sum$ sign with the \texttt{sumlimits} option. With \texttt{nosumlimits}, they are placed beside the sign, raised and lowered with the usual \texttt{\_} and \texttt{\^} characters.

Other symbols that are affected by these options are $\sum \prod \bigcup \bigvee \bigcap \bigvee \bigwedge \bigcirc \bigotimes \bigoplus$. On the other hand, integral signs are not influenced by them.

\texttt{sumlimits}

\begin{align*}
\sum_{n=0}^{\infty} \frac{1}{2^n} &= 2; & \prod_{i=0}^{m-1} n - i &= \frac{n!}{(n - m)!} \\
\text{nosumlimits}
\sum_{n=0}^{\infty} \frac{1}{2^n} &= 2; & \prod_{i=0}^{m-1} n - i &= \frac{n!}{(n - m)!}
\end{align*}

The input text is the same for both of the above cases.

\texttt{intlimits} | \texttt{nointlimits} Integral signs normally have their limits at the side; these options allow them to be placed above and below as for summations.

\texttt{intlimits}

\begin{align*}
\int_{0}^{a} \sqrt{a^2 - x^2} \, dx &= \frac{1}{2} a^2 \sqrt{1 - \sin^2 t} \, dt = a^2 \int_{0}^{\pi/2} \cos^2 t \, dt = \frac{\pi a^2}{4}
\end{align*}
\nointlimits
\[\int_{0}^{a} \sqrt{a^2 - x^2} \, dx = \int_{0}^{1} a^2 \sqrt{1 - \sin^2 t} \, d\sin t = a^2 \int_{0}^{\pi/2} \cos^2 t \, dt = \frac{\pi a^2}{4}\]

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 \nointlimits.

\namelimits | \nonamelimits The function names $\text{\textbackslash det}$, $\text{\textbackslash gcd}$, $\text{\textbackslash inf}$, $\text{\textbackslash lim}$, $\text{\textbackslash liminf}$, $\text{\textbackslash limsup}$ $\text{\textbackslash max}$, $\text{\textbackslash min}$, $\text{\textbackslash Pr}$, and $\text{\textbackslash sup}$ frequently take lower limits which normally appear below the name in displayed formulas. With \nonamelimits, they are placed at the lower right.

\namelimits \quad \nonamelimits
\[\lim_{x \to \infty} \left(1 + \frac{1}{x}\right)^x = e = 2.7182\ldots \quad \lim_{x \to \infty} \left(1 + \frac{1}{x}\right)^x = e = 2.7182\ldots\]

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 $\text{\textbackslash limits}$ and $\text{\textbackslash nolimits}$ commands, as in normal \LaTeX.

The remaining package options select the side for equation numbers and the horizontal positioning of equations.

\lqno | \reqno The standard location for equation numbers is on the right side, at the margin; with \lqno, they are placed on the left side of the equation.

\fleqn With this option, all displayed equations are printed flush left, set off from the left margin by an amount $\text{\textbackslash mathindent}$. Without this option, equations are centered. (This is like the \fleqn class option for the standard classes, page 39.)

12.3 Further $\mathcal{AM}_S$-\LaTeX packages

The $\mathcal{AM}_S$-\LaTeX packages described in this section must be loaded explicitly if their features are to be exploited. Unlike the \amsbsy and \amstext packages, they are not loaded automatically with \amsmath. They may, however, be used on their own, independently of the main package.

12.3.1 Extended theorem declarations

Package: \amsthm The \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
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{\textbackslash newtheorem} command may have one of two optional ones. The complete syntax is

\texttt{\textbackslash newtheorem\{type\}\{num\}_like\{title\}}

or

\texttt{\textbackslash newtheorem\{type\}\{title\}\{in\}_counter}

where \texttt{num\_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\_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.

- A \texttt{\textbackslash newtheorem\*} is provided that defines an unnumbered theorem structure.

- 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}

The desired style is activated by first issuing \texttt{\textbackslash theoremstyle\{style\}}; all subsequent \texttt{\textbackslash newtheorem} statements will have this style until a new one is activated.

- A \texttt{\textbackslash swapnumbers} can be issued to cause all following new theorem types to have the numbers appear before the title, as \texttt{1 Comment}.

- New theorem styles may be defined by the user by means of a \texttt{\textbackslash newtheoremmstyle} 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}.
A proof environment is available for presenting short proofs. It is an unnumbered structure with the title \textit{Proof}. The text is terminated automatically with the Q.E.D. symbol $\Box$. This symbol may be altered by redefining the command \texttt{\qedsymbol}; it may be printed at any time by issuing \texttt{\qed}.

The \texttt{amsthm} package has much in common with Frank Mittelbach's \texttt{theorem} package in the tools collection of Section B.5.4.

12.3.2 Commutative diagrams

The extra \texttt{AMS-L\LaTeX} package \texttt{amscd} makes it easier to generate commutative diagrams like the one here at the right.

These diagrams are created within the \texttt{CD} environment using some additional arrow commands. These bear the rather unusual names \texttt{@>>>} \texttt{@<<<} \texttt{@AAA} and \texttt{@VVV} for arrows pointing right, left, upwards, and downwards, respectively. The command \texttt{@=} draws a horizontal double rule, a lengthened equals sign.

Any text or symbols between the first and second \texttt{>} or \texttt{<} characters will appear above the horizontal arrow in \texttt{\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 \texttt{A} or \texttt{V} are placed to the left; those between the second and third to the right, again in \texttt{\scriptstyle}.

The above example diagram, which is taken from the \texttt{AMS-L\LaTeX} manual \texttt{amsldoc.tex}, was produced with

\begin{verbatim}
\[ \begin{CD}
S^{\mathcal{W}_\Lambda}\otimes T @>j>> T \\
@VVV \@VV{\End P}V \\
(S\otimes T)/I @= (Z\otimes T)/J
\end{CD} \]
\end{verbatim}

The command \texttt{\End} to print the function name 'End' is not standard. It must be previously defined with \texttt{\DeclareMathOperator{\End}{End}} (see page 268).

12.3.3 References with upref package

Normally the numbers printed with the \texttt{\ref} and \texttt{\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 \texttt{AMS-L\LaTeX} package \texttt{upref}. 
12.4 The $\mathcal{AMS}$ fonts

The $\mathcal{AMS}$ 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 $\text{cmmib10}$, the bold symbols $\text{cmbsy10}$, and math extensions $\text{cmex10}$ fonts only in 10 pt size, as indicated by the suffix 10 to their names. The $\mathcal{AMS}$ 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 $\text{cmcsc10}$ is also given companions $\text{cmcsc8}$ and $\text{cmcsc9}$.

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 $\mathcal{AMS}$ 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 $\text{wncy}$, followed by the style designation $r$ (upright), $b$ (bold), $i$ (italic), $sc$ (small caps), or $ss$ (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 $\text{wncyr10}$ has all the same attributes as $\text{cmr10}$, except for the encoding: family $\text{cmr}$, shape $n$, series $m$. (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 81 Ч just as fi is for fi in a Latin font. This means that multi-letter transliterations are simply typed in. The input for ‘Хрущев’ is \cyr Khrushchev, where Kh → Х and shch → щ. The transliteration scheme is that for English; other languages have their own systems to reproduce the original pronunciation. For example, ‘Горбачёв’ is Gorbatschow in German, Gorbaciov in Italian, and Gorbachev in English. These other schemes do not work with these fonts.

Not all letters can be produced so automatically (for example, the ё in Горбачёв) and for this reason the \AMS provides a file 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 Euler collection, named after the eighteenth-century mathematician Leonhard Euler, have been collected from various sources by the \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 amsfonts (which is included automatically with the amssymb package) and then employ the math alphabet command \mathbb. Thus $\mathbb{A B C ..}$ produces

\begin{verbatim}
    ABCDEFGHIJKLMNOPQRSTUVWXYZ
\end{verbatim}

The package eucal redefines the \mathcal command to use the Euler script characters in place of the CM calligraphic letters. If this package is loaded with the option mathscr, the command \mathcal is left unchanged and instead \mathscr is defined to invoke these letters. In this case, $\mathscr{A B C ..}$ produces

\begin{verbatim}
    ABCDEFGHIJKLMNOPQRSTUVWXYZ
\end{verbatim}
12.4. The \textsc{AMS} fonts

Finally, the package eufrak defines the math alphabet command \texttt{\textbackslash mathfrak}, with which $\texttt{\textbackslash mathfrak\{A B C ..\}}$ yields

\[
A \quad B \quad C \quad D \quad E \quad F \quad G \quad H \quad I \quad J \quad K \quad L \quad N \quad O \quad P \quad Q \quad R \quad S \quad T \quad U \quad V \quad W \quad X \quad Y \quad Z
\]

This math alphabet is also enabled with the 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 \textsc{AMS} has produced two fonts, msam10 and 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{\textbackslash 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 amsfonts and then defines names for all the symbols in the two fonts.

For example,

\[
\left[ \texttt{\textbackslash circlearrowright \textbackslash Cup \textbackslash lessapprox \textbackslash lll \textbackslash varpropto \because} \texttt{\textbackslash circeq \textbackslash vDash \textbackslash blacktriangle \textbackslash sphericalangle} \right]
\]

\[
\bigcirc \quad \psi \quad \lesssim \quad \lll \quad \propto \quad \because \quad \ni \quad \Rightarrow \quad \blacktriangle \quad \sphericalangle
\]

All the symbols and their associated names from the 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 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

\begin{verbatim}
\setlength{\unitlength}{length}
\end{verbatim}

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{verbatim}
\begin{picture}(x_dimen,y_dimen)
picture_commands
\end{picture}
\end{verbatim}
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 \textit{x-direction} (horizontal) and \textit{y-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) ... ... ... \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 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 picture environment are enclosed within another environment, such as \texttt{\begin{center} ... \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\((x,y)\)(\textit{\Delta x, \Delta y})\{\textit{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 \textit{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

\((x, y), (x + \Delta x, y + \Delta y), (x + 2\Delta x, y + 2\Delta y), \ldots \) up to
\((x + [\textit{num} - 1]\Delta x, y + [\textit{num} - 1]\Delta y)\)
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,-0.6){5}\{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 \textit{coordinate} and \textit{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.

\textbf{Warning:} Since the comma separates the two numbers in a coordinate pair, it may not be used in place of a decimal point. \textit{For coordinate entries, decimal numbers must be written with a period, not a comma.}

\subsection*{13.1.4 Picture element commands}

\textbf{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.

![An arrow](1.8,1.2)

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:

\begin{itemize}
  \item \texttt{\parbox[b]{..}{..}} \quad Reference point is the lower left corner of the last line in the parbox.
  \item \texttt{\parbox[t]{..}{..}} \quad Reference point is the lower left corner of the top line in the parbox.
  \item \texttt{\parbox[t]{32mm}{...}} \quad For a standard parbox, the reference point is the vertical center of the left edge.
\end{itemize}

\textbf{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 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.'

\textbf{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 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\_dimen, y\_dimen) [*pos*]{text}  
*\framebox*(x\_dimen, y\_dimen) [*pos*]{text}  
*\dashbox*{dash\_len}(x\_dimen, y\_dimen) [*pos*]{text}

The *dimensional* pair *(x\_dimen, y\_dimen)* defines the width and height of the rectangular box in units of `\unitlength`. The positioning argument *pos* determines how the *text* is located within the box. It may take on values:

- **[t]** *top*  The input text appears—centered horizontally—below the upper edge of the box.

- **[b]** *bottom*  The input text appears—centered horizontally—above the lower edge of the box.

- **[l]** *left*  The input text begins—centered vertically—at the left edge of the box.

- **[r]** *right*  The input text ends—centered vertically—at the right edge of the box.

- **[s]** *stretch*  The input text is stretched horizontally to fill up the box, and centered vertically.

*Without* the optional argument *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*  The text appears at the upper left.

- **[tr]** *top right*  The text appears at the upper right.

- **[bl]** *bottom left*  The text appears at the lower left.

- **[br]** *bottom right*  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 `pic elem` in the placement commands `\put` and `\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){center}}
```
The effect of the text positioning argument is made clear with the following examples (UL = 1 cm):

\begin{itemize}
  \item \texttt{top center}
  \item \texttt{bot. left}
  \item \texttt{center right}
  \item \texttt{center stretch}
\end{itemize}

The picture element \texttt{\makebox} is exactly the same as the \texttt{\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.)

The combination \texttt{[lb]} positions the text in exactly the same way as simply typing the text in as \texttt{pic.elem} without a box, as shown on page 290.

The picture element \texttt{\dashbox} also produces a framed box, but with a \textit{dashed} line around it. The argument \texttt{dash.len} specifies the \textit{dash length}.

\begin{itemize}
  \item A dashed frame looks best when both the height and width are a multiple of the dash length.
\end{itemize}

Even in the above picture box commands, the entered \textit{text} may be put into a vertical box (\texttt{\parbox} or \texttt{\minipage}). Since vertical boxes themselves possess an optional positioning argument \texttt{b} or \texttt{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 \texttt{b} or \texttt{t}, then the same value must be applied to the enclosed vertical box. If the picture box command has no positioning argument or only \texttt{r} or \texttt{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

\[
\texttt{\textbackslash line}(\Delta x, \Delta y)\{\textit{length}\}
\]

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 `\put` or `\multiput` command.

\[
\texttt{\textbackslash thicklines} \texttt{\textbackslash put}(0,0)\{\texttt{\textbackslash line}(1,0)\{6}\}\texttt{\textbackslash put}(0,0)\{\texttt{\textbackslash line}(0,1)\{1}\}\texttt{\textbackslash put}(6,0)\{\texttt{\textbackslash line}(0,1)\{0.5}\}
\]

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, ... , 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{\put(2,3){\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{\put(1.0,2.75){\line(2,-1){3.5}}} draws a line starting at point $(1.0, 2.75)$ and length 3.5 going 2.5 units vertically downwards.

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
\texttt{\textbackslash vector}(\Delta x,\Delta y)\{\textit{length}\}

which functions exactly the same as the \texttt{\textbackslash 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{\textit{circle}} picture element is produced with the commands

\begin{verbatim}
\texttt{\textbackslash circle}\{\textit{diameter}\}
\texttt{\textbackslash circle}^\texttt{*}\{\textit{diameter}\}
\end{verbatim}

With the \texttt{*}-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 \textit{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)\thicklines
\put(1,1){\texttt{\textbackslash circle}^\texttt{*}\{0.2\}}
\put(1,1){\texttt{\textbackslash circle}\{1.2\}}
\put(1,1){\texttt{\textbackslash vector}(0,1){0.6}}
\put(2.5,1){\texttt{\textbackslash circle}^\texttt{*}\{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

\[\text{\texttt{\textbackslash oval}(x\_dimen,y\_dimen)[part]}\]

The placement location in the corresponding \texttt{\textbackslash put} command refers to the center of the oval.

\texttt{\textbackslash put(3.0,0.75)\{\texttt{\textbackslash oval}(4.0,1.5)\}}

Here we have set the values \texttt{x\_dimen}=4.0 UL and \texttt{y\_dimen}=1.5 UL, whereby the unit length 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 \texttt{t, b, l, r}, for making half ovals.

\texttt{\textbackslash put(1.75,4.2)\{\texttt{\textbackslash oval}(3.5,1.2)[b]\}}
\texttt{\textbackslash put(1.75,2.6)\{\texttt{\textbackslash oval}(3.5,1.2)[t]\}}
\texttt{\textbackslash put(1.75,1.0)\{\texttt{\textbackslash oval}(3.5,1.2)[l]\}}
\texttt{\textbackslash put(3.25,1.2)\{\texttt{\textbackslash oval}(3.5,1.2)[r]\}}

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 \texttt{tl, tr, bl, or br} to generate a quarter oval. The order of the two letters is unimportant, so that \texttt{tl, tr, bl, and rb} are equally valid.

\texttt{\textbackslash put(2.0,2.5)\{\texttt{\textbackslash oval}(3.0,1.0)[tl]\}}
\texttt{\textbackslash put(2.5,2.5)\{\texttt{\textbackslash oval}(3.0,1.0)[tr]\}}
\texttt{\textbackslash put(1.0,1.5)\{\texttt{\textbackslash oval}(1.0,2.0)[bl]\}}
\texttt{\textbackslash put(3.5,1.5)\{\texttt{\textbackslash oval}(1.0,2.0)[br]\}}

Once again the size specifications refer to the entire oval and not just to the part that is drawn, and the placement coordinates in the \texttt{\textbackslash put} command refer to the center of the complete oval.

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
demonstrate that sections of circles are possible up to a size of about 1.5 cm.

\begin{verbatim}
\put(2.0,1.0){\oval(4.0,4.0)[t]}
\put(2.0,1.0){\oval(1.5,1.5)[t]}
\put(0.75,0.75){\oval(1.5,1.5)[bl]}
\put(1.75,0.0){\oval(1.5,1.5)[tl]}
\put(2.25,0.0){\oval(1.5,1.5)[tr]}
\put(3.25,0.75){\oval(1.5,1.5)[br]}
\end{verbatim}

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.

\begin{verbatim}
\put(0.5,2.5){\oval(1.0,1.0)[t]}
\put(0.5,2.5){\vector(0,-1){2.5}}
\put(1.0,2.5){\vector(0,-1){1.5}}
\put(0.5,2.5){\circle*{0.1}}
\put(2.5,0.5){\line(0,1){2.5}}
\put(2.75,0.5){\oval(0.5,0.5)[bl]}
\put(2.75,0.25){\vector(1,0){1.25}}
\end{verbatim}

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.

\textbf{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.

\textbf{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

\begin{verbatim}
\shortstack[\text{pos}]{\text{col}}
\end{verbatim}

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 \text{col}, each row being separated from the next by \\

The \shortstack command is most frequently implemented for placing short lines of text inside a framed box, or for stacking single letters
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 h and y) will have larger apparent gaps between them than rows without such letters. Adding \strut to each line, as in the third example, equalizes the vertical spacing.

This spacing leaves some room for help
Not really the best for words at all is better

(1.0,0.5) (3.0,0.5) (5.0,0.5) (8.0,0.8) (9.0,1.2)

The placement coordinates of the corresponding \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

\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[r]{A\ 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}}

The \shortstack command may also be used outside of the picture environment within normal text. One possible application is for marginal notes, as in Section 4.10.5.

**Framed text**

The \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 \fbox for drawing a frame around text that fits it exactly (Section 4.7.1). This command is also available in the picture environment.

The amount of spacing between the box frame and the enclosed text is given by the parameter \fboxsep. The placement of an \fbox by means of the \put command occurs in an unexpected manner, as shown below:

\begin{picture}(5,2)
\setlength{\fboxsep}{0.25cm}
\put(0,0){\framebox(5,2){}}
\put(1,1){\fbox{fitted frame}}
\end{picture}

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 com-
mands

\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 \circle, \oval, and \vector, as well as for sloping lines. These may be selected with

\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, \thinlines is in effect.

The thickness of the horizontal and vertical lines may be set with the declaration

\linethickness{thickness}

to any desired size. The argument thickness is a positive length specification. With \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 \linethickness also affects the \framebox and \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

\newsavebox{\sub.pic.name}

which creates a box with the name \sub.pic.name for storing the picture. Afterwards, the sub-picture is saved with

\savebox{\sub.pic.name} (x_dimen,y_dimen) [pos] {sub_pic}

where the arguments (x_dimen,y_dimen) and pos have the same meaning as for \makebox on page 291.

If the picture commands sub_pic are simply a piece of text, this command is exactly the same as the \makebox command except that the text is not printed but stored under the name \sub.pic.name. The sub_pic may be set down anywhere within the main picture as a separate picture element.

\usebox{\sub.pic.name}
This example may not seem very practical, since the \texttt{savebox} and \texttt{usebox} commands could have been replaced by a \texttt{framebox} together with \texttt{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 \texttt{savebox} command may be given outside of the \texttt{picture} environment, and even within the preamble. Such a sub-picture is then available in all \texttt{picture} environments throughout the document. However, if a \texttt{savebox} is defined within an environment, it keeps its contents only until that environment comes to an end.

The picture elements inside a \texttt{savebox} will be sized according to the value of \texttt{unitlength} in effect at the time that the box is constructed. It will not be rescaled by a later change in \texttt{unitlength}.

### 13.1.5 Making graph paper

**Package:** \texttt{graphpap}

The package \texttt{graphpap} adds a new command for drawing gridded paper:

\begin{verbatim}
\graphpaper[\textit{num}](x,y)(lx,ly)
\end{verbatim}

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 \textit{num} units, with every fifth one thicker and labeled. If \textit{num} is not specified, it is assumed to be 10. All arguments must be integers, not decimal fractions. For example, \texttt{\graphpaper(50,50)(200,100)} produces

\begin{center}
\includegraphics[width=0.5\textwidth]{graphpaper.png}
\end{center}

**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 \texttt{picture} environment contains a further coordinate pair as an optional argument
\begin{picture}(x_{\text{dimen}},y_{\text{dimen}})(x_{\text{offset}},y_{\text{offset}})
\text{picture\_commands} \end{picture}

In this form, \((x_{\text{offset}},y_{\text{offset}})\) specifies the coordinates of the lower left corner. This means that for all \(\text{\textbackslash put}\) commands in the environment, the amounts \(x_{\text{offset}}\) and \(y_{\text{offset}}\) are effectively subtracted from the placement coordinates, so that the entire picture is shifted by \(x_{\text{offset}}\) to the left and by \(y_{\text{offset}}\) downwards.

13.2 Extended pictures

13.2.1 The \texttt{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 \TeX{} 2.09, but functions just as well as a \TeX{} 2\epsilon package. Thus it is implemented with

\texttt{\usepackage{epic}}

It makes the following new commands available to the user:

\begin{verbatim}
\multiputlist \matrixput \grid
\dottedline \dashline \drawline
\jput \picsquare \putfile
\end{verbatim}

as well as the environments:

\begin{verbatim}
dottedjoin dashjoin drawjoin
\end{verbatim}

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, \ldots, ObjN}}

places the \(N\) picture elements \(Obj1, \ldots, ObjN\) at \((x,y), (x+\Delta x, y+\Delta y)\), and so on. They are actually put into a series of \texttt{\makebox(0,0)[pos]{}}\}, so that the optional \texttt{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(3,5){3}(10,0)5{\circle{1.5}} produces

```
  o  o  o  o  o
  o  o  o  o  o
  o  o  o  o  o
```

The \grid command is similar to the \graphpaper command, but it will also optionally label the axes.

\grid(width, height) (Δwidth, Δheight) [X₀, Y₀]
draws a grid of size width×height, with grid lines at intervals Δwidth and Δheight. If the optional argument is given, the grid lines are labeled starting at X₀, Y₀, which must be integers, incremented by Δwidth and Δheight, 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} (x₁, y₁) (x₂, y₂) ... (xₙ, yₙ)

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} (x₁, y₁) (x₂, y₂) ... (xₙ, yₙ)

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] (x₁, y₁) (x₂, y₂) ... (xₙ, yₙ)

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){\line(1,1){3}}
    \jput(20,8){\makebox(0,0){\Large$\star$}}
    \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{verbatim}
\begin{picture}(100,13)
  \begin{dottedjoin}{2}
    \jput(0,0){\circle{3}}
    \jput(20,8){\makebox(0,0){\Large$\star$}}
    \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:

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 \% as usual. The \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 \putfile if one wishes. It is also possible to specify the dot character for \texttt{dottedline}, but the default is \texttt{picsquare}. 

\begin{verbatim}
\begin{picture}(100,13)
  \dottedline[.]{3}(0,0)(10,10)(20,0)
  \dashline{3}(30,0)(40,10)(50,0)
  \drawline(60,0)(70,10)(80,0)
\end{picture}
\end{verbatim}

produces

\begin{verbatim}
\begin{picture}(100,13)
  \dottedline[.]{3}(0,0)(10,10)(20,0)
  \dashline{3}(30,0)(40,10)(50,0)
  \drawline(60,0)(70,10)(80,0)
\end{picture}
\end{verbatim}
The default values for the optional stretch argument in \dashline and \drawline are initially 0, but may be changed at any time by redefining (with \renewcommand) \dashlinestretch and \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 picture environment: limited number of slopes for lines, limited line thicknesses, restrictions on circle sizes. This is because the 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 .dvi file by \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 dvips driver, as well as with the previewers xdvi and windvi. It does not work with pdf\LaTeX, unfortunately; on the other hand, it has no problems with 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 \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 epic with one of eepic or eepicemu:

\usepackage{epic,eepic} or \usepackage{epic,eepicemu}

The following standard picture elements are modified by the eepic package:

\line(\Delta x,\Delta y){\textit{length}} (page 293) where \Delta x and \Delta y may take on any positive or negative integer values, and not just those between ±6.

\circle{\textit{diameter}} (page 295) may have any value for diameter; the same for \circle*.

\oval (page 296) may have the maximum diameter of the corners set to any value; this is stored in the length \maxovaldiam which the user may change; the default value is 40 pt.

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{\textbackslash allinethickness\{dimen\}}

sets the line thickness to \textit{dimen} for all picture elements, including circles, ovals, splines, not just for straight lines;

\texttt{\textbackslash Thicklines}

sets thickness of straight lines to 1.5 times that of \texttt{\textbackslash thicklines};

\texttt{\textbackslash path(x_1,y_1) (x_2,y_2) ... (x_n,y_n)}

is a fast version of \texttt{\textbackslash drawline}, without the \textit{stretch} option; a solid line is always drawn;

\texttt{\textbackslash 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{\textbackslash ellipse(x\_diam, y\_diam)}

draws an ellipse with horizontal and vertical sizes \textit{x\_diam}, \textit{y\_diam} respectively; there is also \texttt{*}-form to produce a solid ellipse;

\texttt{\textbackslash 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[width=0.8\textwidth]{figure13.1}
\caption{Comparison of results using the true \texttt{eepic} drawing functions (top) and those with the emulation in \texttt{eepicemu} (bottom). (This example is taken from the \texttt{eepic} user's manual.)}
\end{figure}
13.3 Other drawing packages

13.3.1 The XY-pic package

As indicated at the beginning of this chapter, an exhaustive description of many other specialized packages, for drawing or using \texttt{METAFONT} fonts, can be found in \textit{The \LaTeX\ Graphics Companion} by Goossens et al. (1997). Most of these are designed for \LaTeX\ in general, but will also work with \LaTeX. We mention two of these here just to show the possibilities.

The XY-pic package can carry out general drawing entirely with extra \texttt{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 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/xypic}. The descriptions and illustrations in Chapter 5 of Goossens \textit{et al.} (1997) are especially helpful.

To give a flavor of the syntax in 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 (2,1) {X};
  \node (Y) at (1,1) {Y};
  \node (Z) at (2,0) {Z};

  \draw[->] (U) -- (Y) node [midway, below] {y};
  \draw[->] (Y) -- (Z) node [midway, below] {\theta};
  \draw[->] (U) -- (X) node [midway, right] {x};
  \draw[->] (X) -- (Z) node [midway, right] {p};
  \draw[->] (X) -- (Y) node [midway, right] {q};
  \draw[->] (Y) -- (X) node [midway, above] {(x,y)};
\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 \texttt{&}, 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 means ‘draw an arrow, curving down to the node located two down and one right, and subscript it with $y$.’ And so on for the other connections.

The second example, with \xy...\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 Xy-pic fully.

13.3.2 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 PStricks package for direct use within \TeX{}, and thus within \LaTeX{}.

Actually PStricks consists of over a dozen packages. The complete functionality may be loaded with the package pstricks, or the individual packages may loaded singly. (Since PStricks is really a \TeX{} utility, the \LaTeX{} package pstricks.sty does nothing more than load the pstricks.tex file containing the actual code.) Some of these packages are:

- pst-3d: 3-D drawing
- pst-blur: adding blurred shadows
- pst-char: stroking and filling character paths
- pst-coil: coil and zigzag objects
- pst-eps: export objects to EPS files
- pst-fill: adds \psboxfill command
- pst-gr3d: drawing 3-D grids
- pst-grad: gradient color fills
- pst-lens: simulating effect of a lens
- pst-node: placing and joining nodes
- pst-osci: emulating oscilloscopes
- pst-plot: plotting data
- pst-poly: drawing polygrams
- pst-sline: improvement on pst-grad
- pst-text: setting text along a path
- pst-tree: tree commands

As with Xy-pic, it is beyond the scope of this book on document production with \LaTeX{} to describe the extensive plotting features of PStricks. We refer to Goossens 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 \texmf\generic\source\pstricks.
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{B\LaTeX} 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 \texttt{key} that identifies it, and which may be used in the text document to refer to it. Such a file is called a \textit{bibliographic database}.

14.1 \textbf{The B\LaTeX program}

\LaTeX 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, ...}
\end{verbatim}

at that point in the text where the bibliography is to appear. Here the argument \texttt{database1, database2, ...} 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 \TeX{} program must be run either by clicking the appropriate icon in the editor shell, or from a command line with \TeX{} 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 \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 \TeX{} that this reference is also to be put into the bibliography. With \texttt{\nocite{*}}, \texttt{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 \TeX{} 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 \TeX{} 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:

\texttt{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 \cite commands are issued.

\texttt{unsrt} The entries are ordered according to their first references by the \cite and \nocite commands. The entry for the first \cite
14.2. Creating a bibliographic database

Creating a bibliographic database might seem like more work than typing up a list of references with the \texttt{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 \texttt{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

\begin{verbatim}
@BOOK{knuth:86a,
    AUTHOR = "Donald E. Knuth",
    TITLE = {The \TeX\{}book},
    EDITION = "third",
    PUBLISHER = "Addison--Wesley",
    ADDRESS = {Reading, Massachusetts},
    YEAR = 1986
}
\end{verbatim}

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

```
@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 @b00k 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

```
@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.
@book Entry for a book with a definite publisher.
required fields author or editor, title, publisher, year.
optional fields volume or number, series, address, edition, month, note.

@booklet Entry for a printed and bound work without the name of a publisher or sponsoring organization.
required fields title.
optional fields author, howpublished, address, month, year, note.

@conference Is the same as @inproceedings below.

@inbook Entry for a part (chapter, section, certain pages) of a book.
required fields author or editor, title, chapter and/or pages, publisher, year.
optional fields volume or number, series, type, address, edition, month, note.

@incollection Entry for part of a book that has its own title.
required fields author, title, booktitle, publisher, year.
optional fields editor, volume or number, series, type, chapter, pages, address, edition, month, note.

@inproceedings Entry for an article in conference proceedings.
required fields author, title, booktitle, year.
optional fields editor, volume or number, series, pages, address, month, organization, publisher, note.

@manual Entry for technical documentation.
required fields title.
optional fields author, organization, address, edition, month, year, note.

@mastersthesis Entry for a Master's thesis.
required fields author, title, school, year.
optional fields type, address, month, note.

@misc Entry for a work that does not fit under any of the others.
required fields none.
optional fields author, title, howpublished, month, year, note.

@phdthesis Entry for a PhD thesis.
required fields author, title, school, year.
optional fields type, address, month, note.
Chapter 14. Bibliographic Databases and \texttt{BibTeX}

@proceedings  Entry for conference proceedings.
  \textit{required fields} title, year.
  \textit{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.
  \textit{required fields} author, title, institution, year.
  \textit{optional fields} type, number, address, month, note.

@unpublished  Entry for an unpublished work with an author and title.
  \textit{required fields} author, title, note.
  \textit{optional fields} month, year.

Each entry type may also have the optional fields \texttt{key} and \texttt{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 \texttt{author} field but may also be in the \texttt{editor} or even \texttt{organization} fields. This key field has nothing to do with the \texttt{key} for identifying the entry in the \texttt{\cite} command. The \texttt{crossref} field gives the \texttt{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:

\begin{verbatim}
field_name = \{field text\}  or  
field_name = "field text"
\end{verbatim}

\texttt{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.

\texttt{annote}

An annotation that may be used by non-standard bibliography styles to produce an annotated bibliography. The standard \texttt{BibTeX} styles ignore it.

\texttt{author}

The name(s) of the author(s) as described in Section 14.2.4.

\texttt{booktitle}

The title of a book when only part of it is being cited. See Section 14.2.4 for special considerations on capitalization.

\texttt{chapter}

A chapter or section number.
14.2. Creating a bibliographic database

**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 information 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 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 manage-
ment.

\subsection{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}
14.2. Creating a bibliographic database

The first entry, with key xyz-1, is to obtain all its missing fields from the second entry, by means of the crossref field referring to the entry with key xyz-proceedings. The missing fields are editor, booktitle, and year, those that are common to all articles in the conference proceedings. Note that booktitle is an ignored field for @PROCEEDINGS but needs to be included here since it is required for @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 \cite or \nocite command.

In order for this system to function properly, the entry that is referred to must appear in the database(s) 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 author, editor, title, and booktitle. \TeX 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 \TeX must know what words are always capitalized.

Names

The names in the author and editor fields may be typed in either in the form \{Given Names Surname\} or as \{Surname, Given Names\}. That is, \TeX 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 von or 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 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,

```latex
author = "Kurt G{"o}del",
year = 1931
```

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,

```latex
author = "Helmut Kopka and Daly, Patrick William" or
AUTHOR = {Peter C. Barnes and Tolman, Paul and Mary Smith}
```

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; \texttt{Bib\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.

**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:

\begin{verbatim}
  title = "The \{Giotto\} Mission to Comet \{Halley\}" or
  TITLE = \{The \{G\}iotto Mission to Comet \{H\}alley\}
\end{verbatim}

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

\begin{verbatim}
  " # % ' ( ) , = { }
\end{verbatim}

An abbreviation is defined with the command

\begin{verbatim}
  @string{abbrev_name = \{text\}} or
  @string(abbrev_name = \{text\})
\end{verbatim}

where abbrev_name stands for the name of the abbreviation and text is the replacement text. For example, if the abbreviation

\begin{verbatim}
  @string{JGR = \{Journal of Geophysical Research\}}
\end{verbatim}

has been defined, the following two field statements are identical:

\begin{verbatim}
  journal = JGR
  journal = \{Journal of Geophysical Research\}
\end{verbatim}

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

\begin{verbatim}
  @STRING{jgr = \{Journal of Geophysical Research\}}
  or  @StrinG{jGr = \{Journal of Geophysical Research\}}
\end{verbatim}
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

```latex
@string{yrbk = {Institute Yearbook}}
```

dr

this abbreviation may be combined with other abbreviations and text, as in

```latex
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.

### 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

```latex
@ARTICLE{<key>,
  AUTHOR    = {},
  TITLE     = {},
  YEAR      = {},
  JOURNAL   = {},
  volume    = {},
  number    = {},
  month     = {},
  pages     = {}},
```

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 \BibTeX programming language, something that puzzles even experienced \LaTeX users. There are some 50 .bst files in the \BibTeX 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

```
note = {}  
```

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 \BibTeX has been written by Oren Patashnik, Stanford University, in close association with Leslie Lamport. The \BibTeX installation should contain the two files btxdoc.tex and btxhak.tex for additional information about \BibTeX. 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.
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 geo-
physics can be provided in those .bst files produced by custom-bib,
which then may be used in \BibTeX 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 op-
tions 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 \LaTeX 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 \textit{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 \textit{slide show}, a term which today presupposes the electronic medium. We will use the word \textit{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 \texttt{SLiTEX}

The term \texttt{SLiTEX} was used in the old \texttt{LaTeX} 2.09 system to indicate a separate specialized version of \texttt{LaTeX} for producing projection material. This awkward and cumbersome device was needed then because of the way fonts were hardwired into the formats. With \texttt{LaTeX} 2\textepsilon, this is no longer necessary, and the entire functionality is now found within the \texttt{slides} class, considerably streamlined over the obsolete version, running under regular \texttt{LaTeX}, or \texttt{pdflatex}. The word \texttt{SLiTEX} therefore should be retired, but since is is a nifty expression, we will continue to use it to distinguish the \texttt{slides} class, which is considerably different from \texttt{article} and the other standard classes.

15.1.1 The \texttt{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 \texttt{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 \texttt{LaTeX} class \texttt{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 \texttt{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 \texttt{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 \texttt{slides} class. Instead, one simply makes use of the \texttt{color} package of Section 6.2.

Most of the formatting commands of regular \texttt{LaTeX} may be used with \texttt{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 \texttt{slides} class that is invoked
15.1. Slide production with \TeX

\documentclass{slides}
\usepackage{preamble text}
\begin{document}
\begin{slide} slide text \end{slide}
\end{document}

If colors are wanted the \texttt{color} package of Section 6.2 must be added, for example as

\usepackage[pdftex]{color}

(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 \begin{document} but before the special slide environments described below is output to an unnumbered \textit{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 \textit{slide} itself, possible \textit{overlays}, and additional \textit{notes} to the slide.

\section*{Slides}

A slide or viewgraph is created by means of the environment

\begin{slide} text and commands \end{slide}

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 \texttt{\LaTeX} sans serif \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.

\section*{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} 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

\texttt{\textbackslash invisible and \textbackslash visible}

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{\textbackslash 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{\textbackslash invisible} command does not work with fonts other than the default ones, nor with pdf\LaTeX. However, for pdf\LaTeX, 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{note} text \end{note}

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.

\section{Further features}

\textbf{Page styles}

The \LaTeX command \texttt{\textbackslash pagestyle{style}} may also be used with the \texttt{slides} class. The following styles are available:
15.1. Slide production with \LaTeX

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 \LaTeX, 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 \textit{outside} of the \texttt{slide}, \texttt{overlay}, and \texttt{note} environments.

Timing notes

The option 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, 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
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[\slides]{Which slides to do?}
onlyslides{\slides}

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}

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

- Supports color layers
  as color layers

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\}

\addtime{300}
\begin{note}
Add the overlay only for \LaTeX\ 2.09
\end{note}

\end{document}

15.2 Slide production with seminar

The 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 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\ 2\epsilon 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 seminar (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 seminar is being used for PDF output, as for an electronic presentation, it is vital to include Sebastian Rahtz' hyperref package (Section 10.2.4), which detects the presence of seminar and adjusts the PDF page sizes accordingly. This should be done even if no other hyperref features are utilized. There are some other refinements to make seminar function smoothly with pdf\LaTeX, explained in Section 15.2.7.

15.2.1 Overview

The seminar class is loaded as usual with

\documentclass[options]{seminar}

Here we summarize its main features which differ from St\LaTeX.
• The slide material is put into the \texttt{slide} environment, for landscape slides, and in \texttt{slide*} for portrait slides.

• Regular Roman fonts are used, but this can be changed. The options \texttt{semhelv} and \texttt{sem1cmss} switch to sans serif family with the PostScript Helvetica and the standard \LaTeX{} 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 \texttt{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 \texttt{slide} or \texttt{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 \texttt{slidesonly}, \texttt{notes}, and \texttt{notesonly} control whether only the slides, both slides and notes, or only the notes are printed. By default, both are printed.

• With the \texttt{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 \texttt{notesonly*} option to be used with \texttt{article} to indicate the missing slides.

We now explain these features in more detail.

\subsection*{15.2.2 Magnification and lengths}

Rather than using enlarged fonts as does \LaTeX{}, \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 \texttt{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 \texttt{articlemag\{num\}}; the default step is 0, no magnification.

The output magnification applies to \textit{all} sizes and lengths. This means, any lines drawn with \texttt{rule} or spaces inserted with \texttt{hspace} or \texttt{vspace} are also magnified if they specify absolute units like \texttt{pt} or \texttt{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 \texttt{setslide} and \texttt{addtoslide} in place of \texttt{setlength} and \texttt{addtolength}. For example,

\begin{verbatim}
\newlength{\fixed}
\setslide{\fixed}{3in}
\addtoslide{\fixed}{1cm}
\rule{1pt}{\fixed}
\end{verbatim}

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 \texttt{setarticle} and \texttt{addtoarticle}

The following length commands are slide lengths (do not magnify) and may be used wherever a length is required:

\begin{verbatim}
\semcm \text{1 centimeter}
\semin \text{1 inch}
\textwidth \text{width of the text area}
\textheight \text{height of the text area}
\linewidth \text{current width of text}
\end{verbatim}

These are particularly useful with the \texttt{includegraphics} command and its options \texttt{width=} and \texttt{height=}.

\section*{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.
15.2. Slide production with seminar

When using the `dvips` driver, one can print the landscape and portrait slides together by including the option `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 pdfTeX is given is Section 15.2.7.

When printing both orientations together with the `semrot` option, the head and footlines will not normally be rotated with the text for portrait slides. This can be corrected by issuing `\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, `dvips` can get around this, as explained on page 232. Using `seminar` and `dvips`, one should define

```
\renewcommand{\printlandscape}{\special{landscape}}
```

This also avoids the warning message on processing that the output must be printed in landscape mode.

15.2.4 Slide formatting

Here we describe the various ways to control the size, layout, and other formatting aspects of the slides.

Slide size

By default, the paper size is American letter size, 11 × 8.5 in, and the slide dimensions (slide writing area) are 8.5 × 6.3 in. By specifying the `a4` option the paper size becomes 29.7 × 21 cm and the slide dimensions 22.2 × 15.2 cm. (The `a4paper` option exists, but may cause an error.)

The slide size may be altered temporarily as optional arguments to the `slide` and `slide*` environments, as for example,

```
\begin{slide}{10cm,7cm}
\begin{slide*}{7cm,10cm}
```

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 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 \textit{len} is the maximum space between the end of the line and the right margin. Omitting \textit{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

\slideleftmargin \sliderightmargin \slidetopmargin \slidebottommargin

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[cmds]{style}

The possible values for \textit{style} are \texttt{plain} (the default) or \texttt{none}. The optional \textit{cmds} are settings to adjust the frame parameters, such as \texttt{\fboxrule} and \texttt{\fboxsep}. By including the \texttt{fancybox} package, one also has \texttt{shadow}, \texttt{double}, \texttt{oval}, and \texttt{Oval} as allowed values for \textit{style}, with their corresponding parameters (Section 4.7.9).

All frames make use of the lengths

\slideframewidth 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 \textit{cmds}, one must alter \texttt{\fboxrule} and \texttt{\fboxsep} directly.

It is possible to define one's own frame style with

\newslideframe{new_style}[cmds]{\texttt{box.cmd[#1]}}
where new.style is the name of the new style (for use with \slideframe),
cmds are box parameter settings, and \box.cmd is a box command, like
\fbox or \shadowbox.

There is a semcolor option for seminar, but this is not based on
the \LaTeX\ color package, but depends on PostScript and the dvips driver.
With the color package, one can make use of the \color and \textcolor commands for setting text in color, as well as \colorbox and \fcolorbox for colored boxes and \pagecolor to color the whole page (page 167). Two examples for defining color frame styles are

\newslideframe{gplain}{\fcolorbox{green}{white}{#1}}
\newslideframe{rshadow}{\color{red}\%}{\shadowbox{\color{black}#1\color{red}}}  

The first makes use of the color frame box command \fcolorbox, while
the second shows a trick to combine color commands with \shadowbox, which normally prints a black shadow box. The two \color{red} commands ensure all sides of the box are in red, while the \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 \pagestyle{style} at any time,
or with \thispagestyle{style} for just the current page.

There is also an additional page style align that puts + marks at the
four corners.

You can define your own page styles with

\newpagestyle{style}{headline}{footline}
\renewpagestyle{style}{headline}{footline}

which is preferable to using the styles in the fancyhdr package of Sec-
tion 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

\newpagestyle{mystyle}\%
{\hfill Report\hfill}\%
{March 20, 2004\hfill thepage}\%

Note here that \thepage prints the number for the current page, whereas
\theslide and \thenote refer to the latest slide and note numbers
respectively; \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 \TeX 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 \textit{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 \texttt{slidesonly}, \texttt{notesonly}, or \texttt{notes} to the \texttt{documentclass} command. There is also a \texttt{notesonly*} option to print the notes alone in article output but with indicators for the slides.

A problem arises if \texttt{slidesonly} has been selected. Any global specifications issued after \texttt{\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 \texttt{allversions*} environment. (If you actually want to include text and not just non-printing specifications, use the \texttt{allversions} environment instead.)

Just as for \TeX, 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 \TeX (page 327), the \texttt{\onlyslides\{list\}} command in the preamble ensures that only those slides whose numbers are included in \texttt{list} are processed. However, there is also a \texttt{\nnotslides\{list\}} command to exclude those slides in \texttt{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{\textbackslash ref} commands for slides containing the corresponding \texttt{\textbackslash label}, something that is very useful during development when the slide numbers are yet finalized. For example,

\begin{verbatim}
\onlyslides{4,2,\texttt{\textbackslash ref\{sum\}},10-999}
\end{verbatim}

outputs slides 2, 4, the slide containing \texttt{\textbackslash 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.

\subsection*{15.2.6 Additional features}

\textbf{Counters}

The \texttt{seminar} class provides all the standard \LaTeX\ counters (Section 8.1) as well as \texttt{slide} and \texttt{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 article output.

The \texttt{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 \texttt{equation}, specify this with

\begin{verbatim}
\slidereset{list} or \addtoslidereset{list}
\end{verbatim}

where \texttt{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 \texttt{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}...\end{note}} environments.

\textbf{Overlays}

Overlays for slides can be produced by including the options \texttt{sem\textbackslash layer} and \texttt{sem\textbackslash color}. These make use of PostScript coding and therefore only work with the \texttt{dvips} driver or equivalent.

The \texttt{\overlay{\textbackslash n}} indicates that the following text is to appear on overlay \texttt{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 \pagecolor command from the color package.

However, even more effective is a background image. If this is a graphics file named mybg.png, one can insert it as a background to all slides with the following trick. This does require the fancybox package for the \fancyput command.

\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){
    \includegraphics[totalheight=bgheight,
                       width=bgwidth]{mybg.png}}
\fi

What this does is first to define and set two new lengths, \bgwidth and \bgheight, to be the non-magnified size of the full page, for the current orientation. The \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 \ifarticle construction prevents the background image from being included with 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 .eps file, not .png.

See also page 345 for an alternative means of adding background images.
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 pdf\TeX with seminar

Although the seminar class originated in the pre\-\LaTeX\ 2\epsilon 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 pdf\TeX itself, there are some extra length parameters \texttt{\pdfpagewidth} and \texttt{\pdfpageheight} which need to be properly set to get the output right. This is most simply guaranteed by including the \texttt{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 \texttt{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.)

\begin{verbatim}
\@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}
\end{verbatim}
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 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 article class and with the hyperref package. In fact, it should be considered to be an extension to hyperref. And it should be loaded last, right after hyperref.

The options that may be invoked with the \usepackage command are

- screen to generate the output in the screen version, for online reading;
- print to generate the print version of the output;
- panelleft to place the navigation panel at the left;
- panelright to place the navigation panel at the right;
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:

\emblem{image}
to set the name of the graphics file used for the logo in the navigation panel;
\urlid[url]

to set the web address (URL) of the home page button in the navigation panel;

\screensize{height}{width}

to set the dimensions of the screen; since there are no defaults, this must be given;

\margins{left}{right}{top}{bottom}

to set the margins for the document; again there are no defaults, so this must be given.

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 \panel command. For example,

\renewcommand{\panel}{\colorbox{panelbackground}{
\begin{minipage}[t][\paperheight][b]{\panelwidth}\ . . . . . \\
\Acrobatmenu{FirstPage}
{\addButton{.2in}{$\blacktriangleleft$}}$
{\blacktriangleleft}\\$
\end{minipage}}}

Here the \Acrobatmenu command is from the hyperref package (see page 247) while \panelwidth and \addButton are from pdfscreen. The \panelwidth length is by default 15% of the screen width, and can be redefined. The \addButton{wth}{text} creates a button of width \wth containing \text. In the above example, clicking the button executes the viewer command to go to the first page.

There is also a command

\ImageButton{wth}{ht}{image}

to insert the graphics file \image as a button of width \wth and height \ht.

A row of small navigation buttons can be turned on at the bottom of the page with \bottombuttons and turned off with \nobottombuttons.

Background image

A graphics file can be inserted as background for the whole page with \overlay{image}, or a uniform background color can be selected with \backgroundcolor{color}.

The navigation panel has as background that color previously defined as panelbackground, which the user may alter. Otherwise, the graphics
file \texttt{image} may be inserted as the panel background with the command \\
\texttt{\paneloverlay{image}}.

**Making slides**

There is a \texttt{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 \texttt{pdfscreen} may be written to a file named \texttt{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 \texttt{print} and \texttt{screen} when loading the package.

Text and specifications that are meant exclusively for one output version must be enclosed in \texttt{print} or \texttt{screen} environments. This is especially important for the \texttt{\screensize} and \texttt{\margins} commands, which are inappropriate in \texttt{print} mode.

**Other packages loaded**

The \texttt{pdfscreen} package makes heavy use of several other packages, which it loads automatically. These must then also be installed on the system:

```
hyperref  graphicx  calc  shortvrb
comment   color     amssymb fancybox
truncate  colortbl  amsbsy
```

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, fade overs, 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 \texttt{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 `TExLive` CD, but the latest versions can be obtained from 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

\begin{verbatim}
\hypersetup{pdfpagetransition={pars}}
\end{verbatim}

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:
\texttt{\textbackslash Replace} pages are simply replaced, the default;
\texttt{\textbackslash Dissolve} one page 'dissolves' into the other in a mosaic fashion;
\texttt{\textbackslash VBlinds} several horizontal lines sweep vertically across the screen, like Venetian blinds;
\texttt{\textbackslash HBlinds} same as \texttt{\textbackslash VBlinds}, but vertical lines move horizontally;
\texttt{\textbackslash H0Split} two vertical lines sweep horizontally out from the center;
\texttt{\textbackslash H1Split} as above, but the lines move into the center;
\texttt{\textbackslash V0Split} two horizontal lines sweep vertically out from the center;
\texttt{\textbackslash V1Split} as above, but the lines move into the center;
\texttt{\textbackslash OBox} a box opens out from the center;
\texttt{\textbackslash IBox} a box sweeps into the center;
\texttt{\textbackslash Wipe\{angle\}} a single line sweeps across the screen in the direction given by \texttt{angle}, which can be 0 (left to right), 90 (bottom to top), 180 (right to left), 270 (top to bottom);
\texttt{\textbackslash pageTransitionGlitter\{angle\}} like \texttt{\textbackslash Dissolve} but in a band that moves across the screen in the direction \texttt{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 \texttt{hyperref} commands, and function without PP\texttt{\textbackslash Power4} 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 \texttt{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 \texttt{background} package supplied with PP\texttt{\textbackslash Power4}. This package provides the commands

\begin{verbatim}
\texttt{\textbackslash hpagecolor}\{color1\}\{color2\} and \\
\texttt{\textbackslash 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{\textbackslash definecolor}, or be one of the predefined colors. \textit{This effect requires post-processing with PP\textbackslash Power4!}
The package can also add background images, without PP\textup{ower4} post-processing, if it is loaded with the option \texttt{bgadd}. This then includes the commands

\begin{verbatim}
\texttt{\textbackslash bgadd\{image\} \hspace{1em} and \hspace{1em} \texttt{bgaddcenter\{image\}}}
\end{verbatim}

\texttt{\textbackslash bgadd\{image\}} and \texttt{\textbackslash bgaddcenter\{image\}} to place \texttt{image} at the upper left corner or centered on the page. The \texttt{image} is an \texttt{\textbackslash 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{\textbackslash bgclear}.

\textbf{Note:} The addition of background elements to the pages requires that the package \texttt{eso-pic} also be installed.

\section*{Building a page successively}

\textbf{Package: pause} The electronic equivalent to overlays is the successive building of a page of text. The PP\textup{ower4} \texttt{pause} package introduces the \texttt{\textbackslash pause} command that places markers in the PDF output which PP\textup{ower4} 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{\textbackslash pause} or end of page. This works for both pdf\texttt{T}\textup{EX} and dvipdf\texttt{m}.

For example, we could build a numbered list with

\begin{verbatim}
We proceed as follows:
\begin{enumerate}
    \item We write the list source text.\textbackslash pause
    \item We add the pause markers.\textbackslash pause
    \item We run pdf\TeX.\textbackslash pause
    \item We run PP\textup{ower4}.\textbackslash 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{\textbackslash 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.

\section*{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{\textbackslash pause} with the name of one of the transition declarations from page 344. For example \texttt{\textbackslash pauseHBlinds} is the same as \texttt{\textbackslash pause} plus \texttt{\HBlinds} 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}.

**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 \(n\) will first appear in view \(n\). 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{quote}
And there we have it!\texttt{\textbackslash pause\textbackslash pauselevel\{=1\}}
\end{quote}

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 \(n\);
  \item \{\texttt{=+n}\} increases it by \(n\); bear in mind that the preceding \texttt{\textbackslash pause} has already incremented it by 1;
  \item \{\texttt{=-n}\} decreases it by \(n\);
  \item \{\texttt{=n -d}\} sets level number to \(n\) and makes the change in level number with \texttt{\textbackslash pause} to be \(-d\); 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 \(m\); 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{quote}
\texttt{\textbackslash pause\textbackslash pauselevel\{=2 :2, =5 :6\}Text\textbackslash pause}
\end{quote}

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 \texttt{\rlap} and \texttt{\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:

\begin{verbatim}
We now \rlap{alter}change this word
We now
\pause\pauselevel{=1 :1}\rlap{alter}\pause
change\pause\pauselevel{=1}
this word
\end{verbatim}

The first line shows the text without the \texttt{\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 PP\textsc{ower}4 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

\begin{verbatim}
\pausecolors{textclr}{dullclr}{highclr}
\end{verbatim}

where \texttt{textclr} is a dummy color that is not otherwise being used. Only text indicated in this color will participate in the highlighting. For example

\begin{verbatim}
\pausecolors{cyan}{gray}{red} ...
This is the \textcolor{cyan}{text to be highlighted}
by the ...
\end{verbatim}

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 PP\textsc{ower}4. (Well, it does appear in the first PDF output, before post-processing.)

The highlighting mode is activated with \texttt{\pausehighlight} which makes all text at all levels visible, but with this color switching feature. The build mode can be reactivated with \texttt{\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 \texttt{\textbackslash 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

\begin{verbatim}
\toplink{name}{text}  and  \toptarget{name}
\end{verbatim}

to establish \texttt{text} as a link to the target \texttt{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.

16.1 The \LaTeX{} \texttt{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 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 \name command will be used in the return address in the letterhead, if such has been programmed. The entry in the \signature command will be printed at the end of the letter below the space left blank for the writer's signature. If \signature has not been specified, the \name entry is inserted here instead. This allows a more formal \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 \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 \address is 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} and \telephone{tel\_number}
\end{verbatim}
are provided. In the standard letter class, the entries room\_number and tel\_number are printed at the bottom of the first page only if \address is not issued.

The preamble may also contain the \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 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 \begin{document}. The text consists of one or more letters, each enclosed in a letter environment with the syntax:
\begin{verbatim}
\begin{letter}\{recipient\} text of letter \end{letter}
\end{verbatim}
where recipient stands for the name 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}
16.1. The \texttt{\LaTeX} letter class

The \textit{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 \texttt{Dear Mr Tibs}, and \texttt{regards} stands for the terminating text, for example \texttt{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 \textit{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 \textit{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 \textit{terminating text} and the sender's \textit{name} or \textit{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
Max–Planck-Institut für Aeronomie
Max–Planck-Str. 2
D-37191 Katlenburg-Lindau
Germany

September 8, 2003

TEXproof 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 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 mplett er.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\Germ\any}
\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 \make\label{}s 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 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 *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 *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 *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 *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} 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 *letter* environment 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

*Your Ref.*, *Your letter of*, *Our Ref.*, *Subject:*
Your Ref.: GFM/sdf  Your letter from: April 28, 2003  Our Ref.: PWD/sib

Subject: \LaTeX{} information

Dear George,

Thank you for your inquiry about the latest version of the \LaTeX{} 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 `german` that translates all these words into their German equivalents. The entry commands have the same names, however.

As in the standard `letter` class, the current date is printed automatically, but may be changed to any desired text with the command

\date{\textit{date\_text}}

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 `author` that is given as the argument to the `\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 `\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.

\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}
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 mplettter.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 mplettter.cls. It begins by specifying the \LaTeX format that it requires, and by identifying itself.

\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{mplettter}

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 \texttt{letter} class, along with the package \texttt{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 \texttt{englishnames} and \texttt{germannames}.

\begin{verbatim}
\newcommand{englishnames}{% 
  \newcommand{\yrefname}{\textsl{Your Ref.}} 
  \newcommand{\ymailname}{\textsl{Your letter from}} 
  \newcommand{\myrefname}{\textsl{Our Ref.}} 
  \newcommand{\subjectname}{\textsl{Subject}} 
  \newcommand{\telephonename}{\textsl{Telephone}} 
  \newcommand{\stationname}{Train Station} 
  \newcommand{\germanname}{GERMANY} 
  \newcommand{\telcode}{[49]-5556-401} 
  \newcommand{\postcode}{D--37191} 
} 
\newcommand{germannames}{% 
  \newcommand{\telcode}{(05556) 401} 
  \newcommand{\postcode}{37191} 
} 
\ifthenelse{\boolean{@german}}{\RequirePackage{german}\germannames}{\englishnames}
\end{verbatim}

The last lines test whether the flag \texttt{@german} has been set (by the option \texttt{german}), and if so, the package \texttt{german} is loaded, and the German names are defined, otherwise the English names. The package \texttt{german} already translates the standard names commands \texttt{\toname}, \texttt{\headtoname}, and \texttt{\pasename}, so they are not included in \texttt{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.

\begin{verbatim}
\newcommand{\@yref}{} \newcommand{\@ymail}{} 
\newcommand{\@myref}{} \newcommand{\@subject}{} 
\newcommand{\@internet}{} 
\newcommand{\yref}[1]{\renewcommand{\@yref}{\yrefname: #1}}
\end{verbatim}
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{\xsfsf}{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} \newsavebox{\firmaddress}
\newsavebox{\firm} \newsavebox{\firmreturn}

\sbox{\FIRM}
{\parbox[t]{\leftfield}
{\xviisf MAX--PLANCK--INSTITUT F"UR AERONOMIE}}

\sbox{\firm}
{\xsfsf MAX--PLANCK--INSTITUT F"UR AERONOMIE}

\sbox{\firmreturn}
Using these boxes as building blocks, we put together the actual head and foot of the letterhead page in two further save boxes.

\newsavebox{\firmhead} \newsavebox{\firmfoot}

\sbox{\firmhead}
{\parbox{\textwidth}{\usebox{\FIRM}\raisebox{6pt}{\usebox{\firmaddress}}\[3pt\] \rule{\textwidth}{1pt}}}

\sbox{\firmfoot}
{\parbox{\textwidth}{\rule{\textwidth}{0.6pt}\[5pt\] \viiisf\setlength{\baselineskip}{12pt}\begin{tabular}{@{}ll}\underline{Telephonename} & \telcode-1\\underline{Telefax} & \telcode-240\underline{Telex} & 9\,65\,527 aerli\end{tabular}\hfill\begin{tabular}{l}\underline{Bank}\Kreis--Sparkasse Northeim\41\,104\,449 (BLZ 262\,500\,01)\end{tabular}\hfill\begin{tabular}{l}\underline{Stationname}\ Northeim\(Han.\)\end{tabular} }}

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}{\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\[3pt\]}}
 \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]{1\leftfield}
 {\usebox{\firmreturn}\%
 \parbox[b][3.5cm][c]{1\leftfield}{\toname\toaddress}\%
 \parbox[t]{1\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

Hint: The name has been printed with the font \texttt{cmdunh10 scaled 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
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 tt}, \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{} $2\epsilon$ 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.
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 \TeX\ 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>cmm10</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 cmss 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>
</tr>
</thead>
<tbody>
<tr>
<td>Ultralight</td>
<td>Ultracondensed 50% uc</td>
</tr>
<tr>
<td>Extrapalight</td>
<td>Ultracondensed 62.5% ec</td>
</tr>
<tr>
<td>Light</td>
<td>Condensed 75% c</td>
</tr>
<tr>
<td>Semilight</td>
<td>Semicondensed 87.5% sc</td>
</tr>
<tr>
<td>Medium (normal)</td>
<td>Medium 100% m</td>
</tr>
<tr>
<td>Semibold</td>
<td>Semiexpanded 112.5% sx</td>
</tr>
<tr>
<td>Bold</td>
<td>Expanded 125% x</td>
</tr>
<tr>
<td>Extrabold</td>
<td>Extraexpanded 150% ex</td>
</tr>
<tr>
<td>Ultrabold</td>
<td>Ultraexpanded 200% ux</td>
</tr>
</tbody>
</table>

In \texttt{\fontshape}, the argument \textit{form} is one of the letter combinations \texttt{n}, \texttt{i}, \texttt{s1}, or \texttt{sc} for selecting normal (or upright), italic, slanted, or small caps.

The \texttt{\fontsize} attribute command takes two arguments, the first \texttt{sz} being the point size of the font (without the dimension \texttt{pt} explicitly given) and the second \texttt{line\_sp} being the vertical spacing from one baseline to the next. The second argument becomes the new value of \texttt{\baselineskip} (Section 3.2.4). For example, \texttt{\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 \texttt{pt} is assumed if no dimension is stated.)

Once all five attributes have been set, the font itself is selected with the command \texttt{\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

\texttt{\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 \texttt{\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 \texttt{\selectfont} is issued.

Alternatively, all attributes but the size may be specified and the font activated immediately with the command

\texttt{\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 \texttt{\fontfamily}, \texttt{\fontseries}, and \texttt{\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 \texttt{cmr}, medium series \texttt{m}, normal shape \texttt{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:

- \tiny (5pt)
- \normalsize (10pt)
- \LARGE (17.28pt)
- \scriptsize (7pt)
- \small (9pt)
- \large (12pt)
- \huge (20.74pt)
- \footnotesize (8pt)
- \Large (14.4pt)
- \Huge (24.88pt)

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:

- \rmfamily (cmr)
- \sffamily (cmss)
- \ttfamily (cmtt)

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:

- \mdseries (m)
- \bfseries (bx)

meaning that only a medium and a bold extended series are provided as
standard.

Finally, the shape declarations and their attribute values are:

- \upshape (n)
- \itshape (it)
- \slshape (sl)
- \scshape (sc)

to select upright, slanted, italic, and SMALL CAPS.

Note that there are no high-level declarations for the encoding attrib-
utes. 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

\newcommand{\cyr}{\fontencoding{OT2}\selectfont}
\newcommand{\lat}{\fontencoding{OT1}\selectfont}

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:
Appendix A. The New Font Selection Scheme (NFSS)

Family: \textrm \textsf \texttt
Series: \textmd \textbf
Shape: \textup \textit \textsl \textsc
Other: \textnormal \textemph

The \textemph command is described in Section 4.1.1; \textnormal sets its argument in \textnormalfont.

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 \textitshape are defined as \fontshape{it}\selectfont, whereas in fact they make use of certain default attributes. Thus the true definition of \textitshape is

\fontshape{\itdefault}\selectfont

The default commands available are

Family: \familydefault \sfdefault \ttdefault
Series: \seriesdefault \bfdefault
Shape: \shapedefault \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.

\DeclareFixedFont{\cmd}{\code}{\family}{\series}{\shape}{\size}
defines \cmd to be a declaration that selects the font of the specified attributes. It is rigidly fixed in all attributes. This is equivalent to \newfont except that the font is determined by attributes and not by name.

\DeclareTextFontCommand{\cmd}{\font specs}
defines \cmd to be a font command that sets its argument according to font specs. This command is used internally to define all the font commands like \textbf, which is defined with font specs as \bfseries.

\DeclareOldFontCommand{\cmd}{\text specs}{\math specs}
defines \cmd to be a font 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, \it is defined with

\DeclareOldFontCommand{\it}{\normalfont\itshape}{\mathit}

A.3.3 Mathematical alphabets

The font that has been activated for text processing does 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 \boldmath (Section 5.4.9) must be issued, which remains in effect until it is countermanded by \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

\mathversion{\vers name}
in which the argument vers name currently takes on values normal and bold. The declarations \unboldmath and \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):

\\kr \\cal \\tn \\bf \\sf \\it \\tt
These are all commands operating on arguments rather than 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 \mathsl, give
which means that the new math font command \mathsl selects that font in family cmr with weight m and shape s1, which, with the normal font definitions, is font cmsl in the appropriate size. However, this font will be selected in all math versions, whereas it would be more suitable if a bold font were selected when \mathversion{bold} is in effect. This is accomplished by adding to the definition of \mathsl

\SetMathAlphabet{\mathsl}{bold}{OT1}{cmr}{bx}{sl}

which redefines \mathsl exceptionally for math version bold to be a font with weight bx. For the normal font definitions, this is cmbxs1 in the current size.

New math versions can be created with

\DeclareMathVersion{vers.name}

and the fonts belonging to it are determined by issuing \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 \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

\DeclareSymbolFont{sym_fnt.name}{code}{family}{series}{shape}

which associates the name 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,

\SetSymbolFont{sym_fnt.name}{version}{code}{family}{series}{shape}

may be used to redefine sym_fnt.name for that one version.

The standard \LaTeX setup declares

\DeclareSymbolFont{operators}{OT1}{cmr}{m}{n}\DeclareSymbolFont{letters}{OML}{cmm}{m}{it}\DeclareSymbolFont{symbols}{OMS}{cmsy}{m}{n}\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.
A.3. Installing fonts with NFSS

\DeclareSymbolFontAlphabet{\math_alph}{sym_fnt_name}

defines \math_alph 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

- \mathord: an ordinary symbol
- \mathop: a large operator like Σ
- \mathbin: a binary operator like ×
- \mathrel: a relational operator like ≥
- \mathopen: an opening bracket like {)
- \mathclose: a closing bracket like })
- \mathpunct: punctuation
- \mathalpha: an alphabetic character

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

\DeclareMathDelimiter{\cmd}{type}{sym_fnt1}{pos1}{sym_fnt2}{pos2}
\DeclareMathRadical {\cmd}{sym_fnt1}{pos1}{sym_fnt2}{pos2}

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, \scriptstyle in script, and \scriptscriptstyle 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:

\@encoding \@shape \@size
\@family \@size \@size
\@series \@baselineskip \@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}
A.3. Installing fonts with NFSS

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{code}{family}{series}{shape}{
{font_def}{option}
\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 \texttt{cmr} family are to be represented by external fonts \texttt{cmr5}... \texttt{cmr12} for sizes 5–12 pt, by \texttt{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 \texttt{<-10>} means all sizes up to, but excluding, 10 pt, \texttt{<10-14>} means 10 pt to less than 14 pt, and \texttt{<24->} indicates 24 pt and higher. The possible functions are:

\begin{itemize}
  \item \texttt{(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 \end{verbatim} would load \texttt{cmr10} scaled to 95\% of 11 pt;
  \item \texttt{gen *} generates the font name by appending the point size to the font argument;
  \begin{verbatim}
  <12> gen * cmr \end{verbatim} loads \texttt{cmr12};
  \item \texttt{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 \end{verbatim} loads \texttt{ecss1440};
  \item \texttt{sub *} substitutes a different font whose attributes are given in the font argument as \texttt{fam/ser/shp};
  \begin{verbatim}
  <-> sub * cmtt/m/n \end{verbatim} 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{itemize}
subf \* is like the empty function, but issues a warning that an explicit substitute font has been loaded;

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}

which loads \texttt{cmr12} at 11 pt when 10 pt is requested.

All the above functions may be preceded by an \texttt{s} to suppress messages to the monitor. Thus silent 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}{<-> ssb 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 \texttt{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
This makes Times \texttt{ptm} the default Roman family, invoked with the command \texttt{rmfamily}, Helvetica \texttt{phv} the default sans serif family (called by \texttt{sffamily}), and Courier \texttt{pcr} the default typewriter family (activated by \texttt{ttfamily}). The default attribute for bold face is defined to be \texttt{b} instead of the regular \texttt{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, \texttt{OT2} for Cyrillic, \texttt{T1} for the EC fonts. (These encodings must first be declared, for example with the \texttt{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 \texttt{\O} to print the Scandinavian letter Ø. The position of this character in the font tables depends on the encoding (character 31 in \texttt{OT1} and 216 in \texttt{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 \textit{encoding commands}, which normally appear in the \texttt{codeenc.def} file along with the \texttt{\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 \texttt{\providecommand} and \texttt{\newcommand} except \texttt{\cmd} has the definition \texttt{\def} only when the encoding \texttt{\code} is active. Thus \texttt{\cmd} may have different definitions for each encoding.

\begin{verbatim}
\DeclareTextSymbol{\cmd}{\code}{\pos}
\end{verbatim}

defines \texttt{\cmd} to print the character in the font position \texttt{\pos} when encoding \texttt{\code} is active.

\begin{verbatim}
\DeclareTextAccent{\cmd}{\code}{\pos}
\end{verbatim}

defines \texttt{\cmd} to be an accent command, using the character in font position \texttt{\pos} as the accent symbol, when encoding \texttt{\code} is active.

\begin{verbatim}
\DeclareTextComposite{\cmd}{\code}{\letter}{\pos}
\DeclareTextCompositeCommand{\cmd}{\code}{\letter}{\def}
\end{verbatim}

define the action of command \texttt{\cmd} followed by the single \texttt{\letter} either to print the character in font position \texttt{\pos} or to execute the definition \texttt{\def}. These declarations are most useful with the \texttt{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 (\texttt{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 \texttt{systems} (Figure B.4 on page 390) or from the \TeX Users Group (\url{www.tug.org}).

The system delivered with the \TeXLive CD is that developed by Karl Berry known generically as \texttt{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 \texttt{teTeX} implementation, by Thomas Esser, applies Web2c to Unix machines, while Fabrice Popineau’s \texttt{fpTeX} is the Win32 (Windows 95, 98, NT, 2000, XP) version.

Another excellent \TeX implementation for Windows is \texttt{MikTeX} by Christian Schenk, available from CTAN under \texttt{systems/win32/miktex/}. 
The implementation for DOS, emTeX 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., \texttt{www.YandY.com/}) and \textit{Textures} for Macintosh (Blue Sky Research, \texttt{www.bluesky.com/}). A 'what-you-see-is-what-you-get' \TeX system is \textit{Scientific Workplace}, which can be purchased from MacKichan Software Research (\texttt{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 \texttt{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, \bibTeX, 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.

\subsection{The \TeXLive 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 \TeXLive 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 \texttt{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 TeX. 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 texsetup --maintenance which does the same thing bypassing the Welcome window. Running 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\textbackslash doc\textbackslash t1\textbackslash doc\textbackslash 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\textbackslash doc\ldots}, 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,
B.1.3 Making the \LaTeX format

Adding or updating \LaTeX on to an existing T\TeX installation by hand is relatively straightforward. One must obtain the source files, which on T\TeXLive are located at `\texttt{\textbackslash texmf\textbackslash source\textbackslash latex\textbackslash 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 `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 `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 T\TeX or for some peculiarities of the T\TeX installation; information is to be found in the specific `.txt` file, or in `ltdirchk.dtx`;

*Figure B.2: The T\TeXLive documentation browser*
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, initex may be run on latex.ltx with

\begin{verbatim}
initex latex.ltx or
tex -ini latex.ltx
\end{verbatim}

The resulting latex.fmt must be placed where format files are read. For f\TeX as installed by \TeX Live, this is in texmf-var\web2c\. Other unpacked files must be moved to texmf\tex\latex\base\; these are \texttt{latexbug.tex, testpage.tex, lablst.tex, idx.tex, nfssfont.tex, small2e.tex, sample2e.tex, and docstrip.tex,}

and all files with extensions .cls, .clo, .sty, .fd, .def, and .cfg.

Files with extension .ist are moved to texmf\makeindex\base\. To make up the \LaTeX format with pdf\TeX, run instead

\begin{verbatim}
pdfinitex -fmt=pdf\latex latex.ltx or
pdftex -ini -fmt=pdf\latex latex.ltx
\end{verbatim}

This processes latex.ltx base file once more, but names the resulting format file pdf\latex.fmt rather than 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
meaning ‘run the \TeX or pdf\TeX program with the specified format file’. The system should have shortcuts named \texttt{latex} and \texttt{pdflatex} that translate to the above.

The \texttt{web2c} system, for which \texttt{fp\TeX} is the Win32 implementation, has a different method. It stores all the necessary configuration information in a file named \texttt{texmf.cnf} located in the \texttt{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 \texttt{latex} will use the \texttt{latex.fmt} format file if none is specified. However, there is no real program named \texttt{latex}, so on Unix systems, an alias with this name is established to point to the \texttt{tex} executable program. On Win32, the actual \TeX executable code is in a dynamic link library \texttt{tex.dll} and very small files named \texttt{tex.exe}, \texttt{latex.exe} exist to use this one library. These \texttt{.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 \texttt{pdftex.dll}, \texttt{pdftex.exe}, and \texttt{pdflatex.exe}. If any additional formats are ever needed for these programs, one only needs to copy one of these \texttt{.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 \texttt{web2c}, these are all defined in the \texttt{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 \texttt{web2c} (and \texttt{fp\TeX}), this is carried out with the command

\begin{verbatim}
  mktexlsr
\end{verbatim}

Under \texttt{Mik\TeX}, one must run

\begin{verbatim}
  configure --update-fndb
\end{verbatim}

or click ‘refresh now’ in the \texttt{Mik\TeX} 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 TeXLive, nor on CTAN, since Adobe reserves the exclusive right to distribution, even though free of charge. TeXLive 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><em>1</em>_______.PFB</td>
<td>EuroSans-Regular</td>
<td>zpeurs.pfb</td>
</tr>
<tr>
<td><em>1I</em>_____.PFB</td>
<td>EuroSans-Italic</td>
<td>zpeuris.pfb</td>
</tr>
<tr>
<td><em>1B</em>_____.PFB</td>
<td>EuroSans-Bold</td>
<td>zpeubs.pfb</td>
</tr>
<tr>
<td><em>1BI</em>_____.PFB</td>
<td>EuroSans-BoldItalic</td>
<td>zpeubis.pfb</td>
</tr>
<tr>
<td><em>3</em>_______.PFB</td>
<td>EuroSerif-Regular</td>
<td>zpeur.pfb</td>
</tr>
<tr>
<td><em>3I</em>_____.PFB</td>
<td>EuroSerif-Italic</td>
<td>zpeuri.pfb</td>
</tr>
<tr>
<td><em>3B</em>_____.PFB</td>
<td>EuroSerif-Bold</td>
<td>zpeub.pfb</td>
</tr>
<tr>
<td><em>3BI</em>_____.PFB</td>
<td>EuroSerif-BoldItalic</td>
<td>zpeubi.pfb</td>
</tr>
<tr>
<td><em>2</em>_______.PFB</td>
<td>EuroMono-Regular</td>
<td>zpeurt.pfb</td>
</tr>
<tr>
<td><em>2I</em>_____.PFB</td>
<td>EuroMono-Italic</td>
<td>zpeurit.pfb</td>
</tr>
<tr>
<td><em>2B</em>_____.PFB</td>
<td>EuroMono-Bold</td>
<td>zpeubt.pfb</td>
</tr>
<tr>
<td><em>2BI</em>_____.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 texmf/fonts/type1/adobe/euro (or better in 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 TeXLive 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.
In the early days, anyone setting up a \TeX installation would try to create some sort of order by distributing the files into different directories (also called 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 roadmap.

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 \TeXLive 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 \TeXLive 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 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 .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 \TeXLive 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 Bib\TeX (Section 14.1),

---

**Figure B.3**: The TDS directory tree
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{}3 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 \texttt{.tex} extension. These are:

- \texttt{small2e.tex} a short sample document;
- \texttt{sample2e.tex} a longer sample document;
- \texttt{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;
- \texttt{idx.tex} (Section 9.4) lists all the \texttt{\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;
- \texttt{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;
- \texttt{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 \texttt{\help} prints out a list of commands that can be given;
- \texttt{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 (\texttt{book}, \texttt{report}, \texttt{article}, \texttt{letter}) the standard installation also contains

- \texttt{proc} for producing camera-ready copy of conference proceedings; it is a variant of the two-column \texttt{article} class; it contains one extra command: \texttt{\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 \SuTeX{}, 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 lasy fonts (Sec-
tion 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 MakelIndex program to generate a keyword index;

showidx (Section 9.4) makes the \index entries visible by printing them
as marginal notes;
B.5. Additional standard files

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};

syntaxly 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 \LaTeX3 Team, as individuals, have written a number of packages that they collectively make available; these are to be found in the \texttt{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 \texttt{\afterpage\clearpage}. If \texttt{\clearpage} were issued directly, a new page would be inserted at that point. With \texttt{\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{\sym}}. Thus $\texttt{\alpha + \beta}$ 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{\name}{\sym}} that defines \texttt{\name} to be \texttt{\bm{\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.

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 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 multicol 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 multicolumn 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).


varioref (Section 9.2.4) defines \vref and \vpageref commands, analogous to \ref and \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.

verbatim (Section 4.9.1) is a re-implementation of the verbatim environment that prevents memory overflow for long texts; it also defines a command \verbatiminput to input a file and to print its contents literally, as well as a comment environment for a block comment in the input text.

xr (Section 9.2.3) is a package to allow \ref to cross-reference \label commands from other documents.

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 \PS defined as

\newcommand{\PS}{PostScript\xspace}

may be used as \PS file without having to terminate the \PS command with \_ or \{.

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:

root_name.extension

For every \LaTeX document, there is a main file, whose root name is used when the program \LaTeX is called. It normally possesses the extension .tex, as do any other files that it might read in with \input or \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 \include commands, there will be additional files created having the same root names as the included .tex files but with the extension .aux.

Some of these files are created with every \LaTeX run, whereas others appear only when certain \LaTeX commands have been issued, such as \tableofcontents or \makeindex. The .aux files as well as those produced by special \LaTeX commands may be suppressed by including the command \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 \TeX file extensions with a short description of the role they play in the \TeX processing.

- **.aux** This is the auxiliary file written by \TeX, 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 \TeX, 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 \TeX 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.
Appendix B. The \LaTeX{} Clockwork

\texttt{.def} Additional definition files of various types carry the \texttt{.def} extension. Examples are \texttt{t1enc.def} (to define the T1 encoding) and \texttt{latex209.def} (for compatibility mode between \LaTeX{} 2\epsilon and 2.09).

\texttt{.dtx} These are documented source files for the \LaTeX{} installation. If a \texttt{.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, \texttt{.ltx}, \texttt{.fd} or other file(s) are extracted by processing the \texttt{.dtx} file with DocStrip (Section D.7.1), usually by running \LaTeX{} or \TeX{} on a supplied \texttt{.ins} file.

\texttt{.dvi} 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 \texttt{\nofiles}. The \texttt{.dvi} file must be further processed by a \textit{printer driver} program that converts it into the special commands for the desired output device (printer).

\texttt{.fd} Files containing the NFSS commands associating external font names with font attributes (Section A.3.6) are \textit{font description files}. The root names consist of the encoding and family designations, such as \texttt{ot1cmr.fd}. When a font of a given family is requested for the first time, the corresponding \texttt{.fd} file is automatically read in, if it exists.

\texttt{.fdd} Documented sources for \texttt{.fd} files have the extension \texttt{.fdd}, and are nothing more than a special type of \texttt{.dtx} file. They may be processed for the documentation, or have the \texttt{.fd} files extracted by DocStrip.

\texttt{.fmt} This extension belongs to a \TeX{} \textit{format} file that has been created with the special program \texttt{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 \texttt{latex.fmt}. (Plain \TeX{} uses the format file \texttt{plain.fmt}.)

\texttt{.glo} This file is only generated when the command \texttt{\makeglossary} exists in the preamble. It has the same root name as the main file and contains only \texttt{\glossaryentry} commands that have been produced by \texttt{\glossary} commands in the input text. This file will be suppressed in spite of the presence of \texttt{\makeglossary} if the command \texttt{\nofiles} is issued in the preamble.

\texttt{.gls} This file is the glossary equivalent of the \texttt{.ind} file. It too is generated by the MakelIndex program, but with the \texttt{.glo} file as input. The input and output extensions must be explicitly given. There are no standard \LaTeX{} commands to read in the \texttt{.gls} file. The \texttt{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 \LaTeX) 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 \LaTeX 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 METAPOST source files (Sections G.1 and G.3) contain drawing instructions for the characters in a font. The METAPOST 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 \texttt{nn} from real font \texttt{xx}. 

Appendix B. The \LaTeX Clockwork
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}
Here the command \textbf has been mistakenly typed as \textbf. During the processing, \LaTeX\ assumes that the user wants to invoke a \TeX\ command \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{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 \{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{\input} or \texttt{\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 \texttt{\textbackslash def\textbackslash ed}. If you have misspelled it (e.g., '\texttt{\textbackslash hbox}'), type 'I' and the correct spelling (e.g., '\texttt{\textbackslash 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\texttt{\textbf{texbf}}. 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
problem. 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{qoute}. The \LaTeX processing writes the error message:

\begin{verbatim}
! LaTeX Error: Environment qoute undefined.
See the LaTeX manual or LaTeX Companion for explanation.
Type \texttt{H <return>} for immediate help.
\end{verbatim}

\texttt{1.3 \begin{quote}}
\texttt{\slshape}
\texttt{?}

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 \{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} = −1. The knowledgeable user can set it to 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 \texttt{\begin{quote}}, and the remainder on the next line. Here this remainder is \texttt{\textit{}}. The line indicator \texttt{l.3} (small letter 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 H \langle \textit{return} \rangle yields the additional information:

\begin{quote}
Your command was ignored.
Type I <command> \langle \textit{return} \rangle to replace it with another command, or \langle \textit{return} \rangle to continue without it.
\end{quote}

This means the last command in the upper line of the pair beginning with \texttt{l.3} has not been read into the processing. The mistake may thus be corrected \textit{for the current process run only} with I\texttt{\begin{quote} \langle \textit{return} \rangle}. 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 \langle \textit{return} \rangle, then \texttt{\begin{quote}} is ignored, as though this text were not in the source file. This leads directly to another error when the command \texttt{\end{quote}} is encountered, since now the matching \texttt{\begin{quote}} command is missing. The actual text of this second error message is:

\begin{quote}
! LaTeX Error: \texttt{\begin{document} ended by \texttt{\end{quote}}.  
See the LaTeX manual or LaTeX Companion for explanation.  
Type H \langle \textit{return} \rangle for immediate help.
\end{quote}

\begin{quote}
1.5 \texttt{\end{quote}}
\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 H for help. The error indicator in this case is

\begin{quote}
\texttt{\begin{document} ended by \texttt{\end{quote}}.  
This arises because when \LaTeX encountered \texttt{\end{quote}}, it checked the name of the current environment. Since \texttt{\begin{quote}} was missing, the matching \texttt{\begin} command is the \texttt{\begin{document}} statement, producing a mismatched \texttt{\begin\ldots\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 I entry will not do any good since the missing \texttt{\begin{quote}} can no longer be inserted ahead of the environment text. Entering I\texttt{\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{(return)} 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{qoute} 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 \texttt{H <return>} for immediate help.
...
1.5 \end{qoute}
?
\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 I\texttt{\end{quote}}. However, now a new message appears on the monitor:

\begin{verbatim}
! Extra \endgroup.  
<recently read> \endgroup
1.5 \end{qoute}
?
\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{(return)}. 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 \textit{⟨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{environment}} command, the proper correction of the error is

\texttt{I\begin{environment}{right\_name}}

If the misspelling has occurred in the \texttt{\end{environment}} command, the best way to handle this mistake is just to press \textit{⟨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{environment}} 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

\texttt{I \ correction}

Otherwise, one may give \textit{⟨return⟩} and wait and see what happens. Even if more peculiar \TeX{} error messages appear, one can keep on pressing \textit{⟨return⟩} until the processing is finally at an end. The following print-out should indicate where the mistake was.

Instead of continually pressing \textit{⟨return⟩}, one may also enter S, R, or Q and \textit{⟨return⟩} to speed up the error treatment (Section C.1.1). Here, as with the simple \textit{⟨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{environment}} command with a valid environment name earlier. It could be assumed that the name of the environment in the \texttt{\end{environment}} command is wrong. \LaTeX{} thus tries to execute this command using the name of the current environment.

\subsection{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
called as a unit. $\TeX$ macros are structures similar to those that may be defined by the $\LaTeX$ command $\texttt{\newcommand}$. 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 $\texttt{\newcommand}$.

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 $\texttt{\centerline}$ 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 \invalid command}
end{document}
\end{verbatim}

in which a $\texttt{\textbackslash}$ has been placed before the word $\texttt{invalid}$, making a false command named $\texttt{\invalid}$. During the $\LaTeX$ processing, the following $\TeX$ error message appears:

\begin{verbatim}
! Undefined control sequence.
<argument> This is an \invalid command
1.3 \centerline{This is an \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 \invalid command
\end{verbatim}

The following pair of lines claim that the error was recognized after the ‘command’ $\texttt{\invalid}$ was processed, and that the next piece of text to be read in is the word 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 \{return\} key led to a second error message in spite of the fact that the \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{itemize}
\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 \texttt{itemie} undefined.

See the \LaTeX{} manual or \LaTeX{} Companion for explanation. Type \texttt{H \langle return \rangle} for immediate help.

\ldots

1.3 \begin{itemize}
\?

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 \langle return \rangle} for immediate help.

\ldots

1.4 \texttt{\item T}

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 \{return\}} for help yields
Appendix C. Error Messages

Try typing <return> to proceed. If that doesn't work, type X <return> to quit.
?

Following this advice and pressing \(\text{return}\), one obtains the same error message once again, but this time on line 5, for the second \item command. Persevering further and pressing \(\text{return}\) once again, one is presented with

! LaTeX Error: \begin{document} ended by \end{itemize}.

See the LaTeX manual or LaTeX Companion for explanation. Type H <return> for immediate help.
...

1.6 \end{itemize}
?

The itemize environment has now come to its end, but since it was never properly started, \LaTeX{} complains about a mismatch of \begin{document} and \end{document} commands. Now at last, typing another \(\text{return}\) 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 \textlangle return\textrangle} 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: even when faced with a host of error messages, don't panic! Continue to press the \textlangle return\textrangle key to advance the processing.

Pressing S \textlangle return\textrangle 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).

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

\begin{verbatim}
This file has no preamble.
\end{verbatim}

is put directly into the \LaTeX\ processing, the following message appears on the monitor:

\begin{verbatim}
! \LaTeX\ Error: Missing \texttt{\begin{document}}.
See the \LaTeX\ manual or \LaTeX\ Companion for explanation.
Type \texttt{H <return>} for immediate help.
...
\end{verbatim}

\texttt{l.1 T}

\begin{verbatim}
his file has no preamble.
\end{verbatim}

? with the accompanying help message after typing \texttt{H <\textbackslash return>}:

\begin{verbatim}
You're in trouble here. Try typing \texttt{<\textbackslash return>} to proceed.
If that doesn't work, type \texttt{X <\textbackslash return>} to quit.
? 
\end{verbatim}

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{<\textbackslash 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{verbatim}
\begin{document}
This file has no preamble.
\end{document}
\end{verbatim}

it is not possible to carry out a proper processing. The \TeX\ error message now printed is

\begin{verbatim}
! \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 <\textbackslash return>} for immediate help.
...
\end{verbatim}

\texttt{l.1 \begin{document}}

?
The error indicator is rather peculiar since the command \texttt{\normalshape size} 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 \texttt{\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 \texttt{\documentclass} command that must precede it is missing.

Here again there is no choice but to halt the program with \texttt{X} or \texttt{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 \texttt{\begin{document}} statement, including the standard font. It is here that \texttt{\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

\texttt{\documentclass{fred}}

the following message appears:

! LaTeX Error: File ‘fred.cls’ not found.

Type \texttt{X} to quit or \texttt{<RETURN>} to proceed, or enter new name. (Default extension: \texttt{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 (\texttt{.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 \texttt{X} (\texttt{return}) only produces the
Some sample errors

message once again, this time stating that it cannot find the file x\text{.ext}. Some installations provide a dummy file named null\text{.tex}, while one of the extension packages available (Section B.5.4) offers a set of files named after the standard response letters x\text{.tex}, e\text{.tex}, r\text{.tex}, h\text{.tex}, s\text{.tex} to emulate the responses to normal error messages.

\textbf{Emergency stop:} sometimes one reaches a situation where the program cannot be halted after an error, even with I\text{\textbackslash stop} or X. In such a case, one has to apply the operating system's program interrupt, usually CTRL \text{ Z}.

\section*{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\textunderscore sample’, and types the input text:

\begin{verbatim}
The price is $3.50 and the order number is type\textunderscore sample.
\end{verbatim}

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 $\text{\textbackslash}$. 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>
  $\text{\textbackslash}$
  1.4
?

If \textit{\textbackslash return} 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:

\begin{verbatim}
The price is 3.50 and the order number is type\textunderscore sample.
\end{verbatim}

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
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 \langle return\rangle here tells \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, \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 \langle return\rangle response, with the result:

The price is $3.50 and the order number is typeample.

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: \textit{keep on pressing \langle return\rangle or S to get through to the end
of the processing, and then look at the printed output to find the mistake.}

\section*{C.2.4 Errors from multi-file texts}

If the document text is split over many files that are to be read in with
\texttt{\textbackslash input} or \texttt{\textbackslash include} commands, the line number in the error message
refers to the file currently being read. A response with E \langle return\rangle 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 \texttt{.log} file. As the
files are opened for reading, \TeX{} writes an opening parenthesis ( and the
name of the file to the monitor and to the transcript \texttt{.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
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: ... undefined.

The argument of a \renewcommand or \renewenvironment has not been previously defined. The corresponding \new... command should be used instead.

! \LaTeX Error: \(< in mid line.

The command \(< 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.)
C.3. List of \LaTeX{} error messages

! \LaTeX{} Error: Command ... undefined in encoding ....
The specified command has been defined with \texttt{\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 \texttt{\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 \texttt{\langle \text{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 \texttt{\marginpar} command was given within a vertical box (\texttt{\parbox} or \texttt{\minipage} environment), or these structures were issued within a \LaTeX{} command that uses vertical boxes internally, 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 \texttt{\tabular} or \texttt{\array} environment contains an unknown column formatting character (see Section 4.8.1) or the formatting entry in the second argument of a \texttt{\multicolumn} command is wrong.

! \LaTeX{} Error: \texttt{\include} cannot be nested.
An attempt has been made to call \texttt{\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 \LaTeX\ 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 `lrbox` 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 thebibliography 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 \texttt{return}. In this case, the processing should be brought to a halt with I\textbackslash 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. List of \LaTeX\ error messages

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 mypack could print the error

```
Package mypack Error: cannot mix options 'good'
(mypack) and 'bad'.
```

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: \LoadClass in package file.
A package file has called \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: \RequirePackage or \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 \LaTeX2e, the format name specified must be \LaTeX2e. 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 LATEX2ε. 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 \DeclareFontShape command has been issued for a font encoding and family combination that has not been previously declared by means of a \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 \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 \DeclareMathAlphabet with an empty shape attribute, but no \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 \Set... or \Declare... commands that take a math version name as argument. The version name must be declared with \DeclareMathVersion before it may be used.

! LaTeX Error: Not a command name: ‘...’.
The first argument of \DeclareMathAccent must be a command name, like \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 \Set.. or \Declare.. commands that take a symbol font name as argument. The font name must be declared with \DeclareSymbolFont before it may be used as a symbol font.
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.

\begin{itemize}
\item \textbf{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 \texttt{\thanks} commands on a title page.
\item \textbf{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 $x_{23}$ is with $x_{2\{3\}}$ or $x_{2\{3\}}$ (Section 5.2.2).
\item \textbf{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 $x^{23}$ is with $x^{2\{3\}}$ or $x^{2\{3\}}$ (Section 5.2.2).
\end{itemize}
! 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 I\& 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 `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 `\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 `return` and try to deduce where it might be from the printed output.

! Missing `$` inserted.
Most likely a symbol or command that may only appear in math mode was used in normal text. Recall that all those commands described in Chapter 5 are only allowed in math mode unless otherwise stated. If an `\mbox` command is inserted within a math formula, its argument has temporarily exited from math to normal text mode. The message may also occur if a blank line appears within a math formula, signalling a new paragraph and the end of the formula without the necessary `$` sign.

! Not a letter.
The word list in a `\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 `\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
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 tabbing, tabular, array, or 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 \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 \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 \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 } has been forgotten in the argument of a counter command such as \setcounter or in a \newenvironment or \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 \multiput command contains a picture environment, which in turn possesses a \footnotesize declaration with another \multiput command, and so on. Such a nesting must be simplified, unless the real problem is a forgotten closing brace } 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 \textbackslash\textbackslash 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 \textbackslash@array, there is a
faulty @-expression (Section 4.8.1) in a \texttt{tabular} or \texttt{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 \textbackslash# in order to print # itself. This can be corrected during the
processing with the response I\textbackslash# and \textbackslash\textbackslash return.

\section{Warnings}

\LaTeX 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.

\subsection{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 \texttt{\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 \texttt{\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 \texttt{article.sty}, that have a new equivalent with a different name (in this case \texttt{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 \texttt{\label} or \texttt{\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 \texttt{\ref}, \texttt{\pageref}, and \texttt{\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 \texttt{\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.}
\texttt{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 ‘...’},
\texttt{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 ...,}
\texttt{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{},}
\texttt{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: \textit{LaTeX Font Warning:} plus the warning text.

\textbf{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.

\textbf{LaTeX Font Warning:} Command \texttt{\textbackslash tracingfonts} not provided.

(Font) Use the \texttt{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.

\textbf{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.

\textbf{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.

\textbf{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 \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

```
Overfull \hbox (17.2122pt too wide) in paragraph at lines 4--6
\OT1/cmr/m/n/10 If \TeX{} can-not find an ap-pro-pri-ate
spot to di-vide a word at the end of the line, as right
here aaaaaaaaaaaaaaaaaaaaaaaaa
```

one knows that the line is about 17.2 pt (6 mm) too long and extends this amount into the right-hand margin. The line is part of the paragraph in lines 4 to 6. The font used is designated by its attributes \OT1/cmr/m/n/10. The text of the problem line is If ... ... right here aaaaaaaaaaaaaaaaaaaaaaaaa. Possible word divisions are shown with hyphens, such as di-vide. The last word cannot be divided, which is the cause of the problem.

A way around this is to include some suggested hyphenations with the \- command, such as aaaaaaaaa-a-aaaaaaaaaaaaaaaa. Similarly, a \linebreak command before the problem word, or setting the whole paragraph in a 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.

```
Overfull \vbox ....
```

This warning occurs very rarely. \TeX{} could not break the page properly so that the text extends beyond the bottom of the page. More often \TeX{} 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 \textheight, such as a long table.

```
Underfull \hbox ....
```

This is the opposite of the Overfull \hbox warning. It appears when \TeX{} 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 sloppypar environment, a sloppy declaration, or a \linebreak command. It may also come about after an inappropriate application of a \ or \newline command, such as two \ 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 Overfull \hbox a \linebreak is included in the text at the spot ..., as right\linebreak here, then the warning becomes

```
Underfull \hbox (badness 5504) in paragraph at lines 4--6
\OT1/cmr/m/n/10 If \TeX{} can-not find an ap-pro-pri-ate
spot to di-vide a word at the end of the line, as right
```

which states that the paragraph in lines 4 to 6 contains an output line in which the interword spacing may be unacceptably wide. The text of this line reads If ... ..., as right and is set in the font \OT1/cmr/m/n/10. The evaluation
badness 5504 is \TeX's estimate of how unacceptable the spacing is: *the smaller this number the better.*

Underfull \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 *previous* and a *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.

D.1 Class and package files

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 *The \LaTeX{} Companion* (Goossens *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.
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
D.1. Class and package files

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{} 2\epsilon, 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{} 2\epsilon. 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{format}[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 LaTeX2e actually recognizes this statement. All others will immediately issue the error message

! Undefined control sequence.
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 \LaTeX, a
warning is printed. For example, the command \DeclareRobustCommand
did not exist in the preliminary test release of \LaTeX, 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 com-
mands \usepackage and \ProvidesPackage.
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 \input.

\ProvidesFile{file_name}[version]

There is no checking of the name or version in this case, but both pieces of information will be printed by \listfiles.

D.2.2 Loading further classes and packages

In the main document file, classes are read in by means of the initializing \documentclass command, and packages with \usepackage. Within class and package files, the commands

\LoadClass[options]{class}[version]
\RequirePackage[options]{package}[version]
\LoadClassWithOptions{class}[version]
\RequirePackageWithOptions{package}[version]

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 \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 \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

\DeclareOption{option}{code}

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 \textit{not} be used within the option code!) Two examples from \texttt{article.cls} are

\DeclareOption{fleqn}{\input{fleqn.clo}}
\DeclareOption{twocolumn}{\setboolean{@twocolumn}{true}}
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

\begin{verbatim}
\DeclareOption*{%\IfFileExists{\CurrentOption.sty}{}{\OptionNotUsed}}
\end{verbatim}

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

\begin{verbatim}
\ExecuteOptions{option_list}
\ProcessOptions
\ProcessOptions*
\end{verbatim}

where \ExecuteOptions calls those commands defined for the options in \textit{option_list}. This is normally done to establish certain options as being present by default. \ProcessOptions executes all the requested options \textit{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 \textit{in the order requested}.

It is also possible to transfer options to a class or package file with

\begin{verbatim}
\PassOptionsToClass{options}{class_name}
\PassOptionsToPackage{options}{package_name}
\end{verbatim}

where \textit{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 \texttt{\documentclass} statement are designated \textit{global}; they are considered to apply to all subsequent packages, but not to classes loaded with \texttt{\LoadClass}; if they are not defined in the class, no error or warning is issued;
• all options requested with other commands, includingältex\LoadClass and theältex\PassOptionsTo., are local; if they are not defined in that class or package, an error is issued;

• if there 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, unlessältex\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

ältex\AtEndOfClass{cmds}
ältex\AtEndOfPackage{cmds}
ältex\AtBeginDocument{cmds}
ältex\AtEndDocument{cmds}

The first two store away 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 cmds to be executed withältex\begin{document} andältex\end{document} respectively. All of these may be issued more than once, in which case the cmds are processed in the order in which they were issued.

The cmds stored withältex\AtBeginDocument are inserted into the processing stream effectively within the preamble, but after the commandältex\begin{document} has done almost everything else that it does. Thus the 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.
D.2. \LaTeX programming commands

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}[\narg][\opt]{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}[\narg][\opt]{def}
\end{verbatim}

and issues a warning message if the actual definition is not the same as \texttt{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 \texttt{\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* \CheckCommand*
\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.
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}}{\textit{error\_text}}{\textit{help}}
\PackageError{\textit{package\_name}}{\textit{error\_text}}{\textit{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 \texttt{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 \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 \texttt{H <return>} for immediate help.

Typing \texttt{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}}{\textit{warning\_text}}
\ClassWarningNoLine{\textit{class\_name}}{\textit{warning\_text}}
\PackageWarning{\textit{package\_name}}{\textit{warning\_text}}
\PackageWarningNoLine{\textit{package\_name}}{\textit{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

\begin{verbatim}
\documentclass{article}
\usepackage{ifthen}
\listfiles
\begin{document}
\input{mymacros}
This is \te.
\end{document}
\end{verbatim}

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

\begin{verbatim}
\begin{filecontents}{file name}
file contents
\end{filecontents}
\end{verbatim}

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 \textit{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
the newly written file \texttt{mymacros.tex} contains

\begin{verbatim}
\begin{filecontents}{mymacros}
  \newcommand{\te}{the end}
\end{filecontents}
\end{verbatim}

Note that the \texttt{filecontents} environment adds some comment lines to explain where the new file came from. If this is undesirable, the \texttt{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.

\begin{verbatim}
\@namedef{cmd}{def}
\@nameuse{cmd}
\end{verbatim}

define and execute a command named \texttt{cmd}, where the backslash is not included in the command name. This name may contain any characters, even those normally forbidden in command names.

\begin{verbatim}
\@ifundefined{cmd}{true}{false}
\end{verbatim}
executes \texttt{true} if the command \texttt{cmd} does not exist, else \texttt{false}. Again, the backslash is not included in \texttt{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 \texttt{providecommand}. It may also be employed to determine whether the main class is \texttt{article}-like or not: \begin{verbatim}
\@ifundefined{chapter}{..}{..}
\end{verbatim} tests for the existence of the \texttt{chapter} command.

\begin{verbatim}
\@ifnextchar{true}{false}
\end{verbatim}
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.

\begin{verbatim}
\@ifstar{true}{false}
\end{verbatim}
Appendix D. \LaTeX Programming

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 The \TeXbook (Knuth, 1986a), but also in the excellent reference manual by
Eijkhout (1992), \TeX by Topic, a \TeXnician's Reference.

We do not intend this book to a manual for \TeX; nevertheless, there are a few
common \TeX 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 \newcommand*
except that there is no check for name clashes, and the arguments are specified
differently. For example, a command \Exp to write scientific notation can be
defined as

\def\Exp#1#2{\ensuremath{#1\times10^{#2}}} 

or as \newcommand*{\Exp}[2]{\ensuremath{#1\times10^{#2}}}

In both cases, \Exp{1.1}{4} produces $1.1 \times 10^4$. However, \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 \Exp{1.1}(4), something that cannot be
produced with \newcommand. The \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 \def: the first makes a global definition, valid even outside the
current environment or {...} bracketing; the second is an 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

\noexpand \expandafter
control the expansion of commands in definitions and execution. Any commands
in the definition part of \edef are expanded (their meanings inserted) unless
they have \noexpand before them. The opposite is achieved with \expandafter,
which jumps over the following command, expands the next one, and then
executes the one skipped. This is very deep \TeXnology, and is best illustrated by
an example with the \Exp command defined above.

\newcommand*{\mynums}{1.1(4)} \expandafter\Exp\mynums
is identical to \Exp1.1(4), whereas \Exp\mynums is not; \mynums is expanded
to 1.1(4) before \Exp is executed.

\let\cmd_a = \cmd_b or \let\cmd_a\cmd_b
makes \cmd_a take on the current meaning of \cmd_b. This is often employed
to save the current meaning of a command before redefining it, possibly using
the older meaning too.

\relax
does absolutely nothing, but it is often inserted in places where something should
be but nothing is wanted.

\ifcond true_code \else false_code \fi
is the form of a \TeX conditional. There are too many variations on the condition
cond to explain here, but one common application is the equivalent of the \LaTeX
boolean switch commands:

\newif\ifflag = \newboolean{flag}
\flagtrue = \setboolean{flag}{true}
\flagfalse = \setboolean{flag}{false}
\ifflag .. \else .. \fi = \ifthenelse{\boolean{flag}}{..}{..}

For those who are used to it, the \TeX form is more compact, but does not conform
to the general \LaTeX style of doing things.

\ifcase num text_0 \or text_1 \or ... \fi
executes one of the text_num according to the value of num.

\endinput

terminates the current file being input. This is not really necessary, but it is
considered good programming to end all files this way. The main document file
does not need it since \end{document} has the same effect.

D.3 Sample packages

We present here some demonstration packages to illustrate the program-
ing 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 \texttt{a4paper} or \texttt{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 \texttt{geometry} package from Section 3.2.6 does this and much more. Here we present a simpler package for this single function.

The \texttt{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 \texttt{ifthen} package is then loaded since conditionals will be required.

\begin{verbatim}
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{fullpage}[2002/02/15 1.1 (PWD)]
\RequirePackage{ifthen}
\end{verbatim}

Next the options are prepared. These are to be \texttt{in} and \texttt{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.

\begin{verbatim}
\newlength{\FP@margin}
\DeclareOption{in}{\setlength{\FP@margin}{1in}}
\DeclareOption{cm}{\setlength{\FP@margin}{1.5cm}}
\end{verbatim}

Furthermore, the four standard page styles are also to be option names. They will set two internal boolean switches and the page style.

\begin{verbatim}
\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}
\end{verbatim}
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 (true). Then set the space reserved for the footline to zero for the empty page style (FP@empty is (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 \topmargin and \oddsidemargin.

\setlength{\textwidth}{\paperwidth}
\addtolength{\textwidth}{-2\FP@margin}
\setlength{\oddsidemargin}{\FP@margin}
\addtolength{\oddsidemargin}{-1in}
\setlength{\evensidemargin}{\oddsidemargin}
\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 \paperheight and \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{0pt}:˜}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}\n
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}{\thechapter-\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 subsubsection; 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{\textbackslash section} command which is to be centered, \texttt{\textbackslash Large} and upper case.

\begin{verbatim}
\renewcommand{\section}{\@startsection {section}{1}{0pt}%-3.5ex plus -1ex minus -.2ex}% {2.3ex plus 2ex}%;\centering\normalfont\Large\MakeUppercase}
\end{verbatim}

The subsections and subsubsection titles are in small caps and bold slanted fonts respectively.

\begin{verbatim}
\renewcommand{\subsection}{\@startsection{subsection}{2}{20pt}%-3.25ex plus -1ex minus -.2ex}% {1.5ex plus .2ex}%;\normalfont\large\scshape}
\renewcommand{\subsubsection}{\@startsection{subsubsection}{3}%-3.25ex plus -1ex minus -.2ex}% {1.5ex plus .2ex}%;\normalfont\normalsize\bfseries\slshape}
\end{verbatim}

The paragraph and subparagraph titles are to be run-in types (negative post-skip).

\begin{verbatim}
\renewcommand{\paragraph}{\@startsection{paragraph}{4}{0pt}%-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}
\end{verbatim}

Finally, redefine the section numbers to have a period after them.

\begin{verbatim}
\renewcommand{\@seccntformat}[1]{\@nameuse{the#1}.\quad}
\end{verbatim}

As an example of output with mysects, consider the short document:

\begin{verbatim}
\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}
\end{document}
\end{verbatim}
The priest fulfills several functions.

\subparagraph{At the Altar}
Here the priest was master of ceremonies.
\end{document}

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 \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 \chaptername exists only in classes \book and \report, but not in \article. This means, if a package were to redefine the word ‘Chapter’ for all classes, it must do so with something like

\providecommand*{\chaptername}{}
\renewcommand*{\chaptername}{Chapitre}

The standard set of name commands and their initial values are:

(defined in \book, \report, and \article classes)

\begin{itemize}
    \item \contentsname   {Contents}
    \item \listfigurename {List of Figures}
    \item \listtablename   {List of Tables}
    \item \indexname      {Index}
    \item \figurename     {Figure}
    \item \tablename      {Table}
\end{itemize}
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 \caption*{} such as \caption*{german} and \caption*{english} that allow for convenient switching back and forth between different languages. For example,

\newcommand*{\captionsgerman}{% \renewcommand*{\contentsname}{Inhaltsverzeichnis} ... \renewcommand*{\alsoname}{siehe auch}}

### D.4.2 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

\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
    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
    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.

D.5 Direct typing of special letters

Package: inputenc

In Section 2.5.9, we explained the 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{\textbackslash ss}. The source file however, receives only the code number 223, which \LaTeX{} would normally reject as unknown. The inputenc package with the ansinew option tells it that character 223 is to be treated as \texttt{\textbackslash 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 .def file defining the extended characters with the commands

\begin{verbatim}
\DeclareInputText{pos}{text} or \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}{\textbackslash ss}
\end{verbatim}

to translate the input character code 223 to the \LaTeX{} command \texttt{\textbackslash ss}.

Some \TeX{} installations automatically contain such conversions for their own local input coding scheme; files written under such a system without
Appendix D. \LaTeX Programming

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><strong>Ligatures</strong></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>’</td>
<td>‘</td>
</tr>
<tr>
<td>\textquotedblright</td>
<td>’</td>
<td>’</td>
</tr>
<tr>
<td>\textquoteleft</td>
<td>‘</td>
<td>‘</td>
</tr>
<tr>
<td>\textquoteright</td>
<td>’</td>
<td>’</td>
</tr>
<tr>
<td><strong>Math symbols</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\textbullet</td>
<td>•</td>
<td>$\bullet$</td>
</tr>
<tr>
<td>\textperiodcentered</td>
<td>.</td>
<td>$\cdot$</td>
</tr>
<tr>
<td>\textbackslash</td>
<td>\</td>
<td>$\setminus$</td>
</tr>
<tr>
<td>\textbar</td>
<td></td>
<td></td>
</tr>
<tr>
<td>\textless</td>
<td>&lt;</td>
<td>$&lt;$</td>
</tr>
<tr>
<td>\textgreater</td>
<td>&gt;</td>
<td>$&gt;$</td>
</tr>
<tr>
<td><strong>Miscellaneous</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\textvisiblespace</td>
<td>_</td>
<td>\verb*+ +</td>
</tr>
<tr>
<td>\textasciicircum</td>
<td>^</td>
<td>\verb+^+</td>
</tr>
<tr>
<td>\textasciitilde</td>
<td>-</td>
<td>\verb+~+</td>
</tr>
<tr>
<td>\textsuperscript{123}</td>
<td>123</td>
<td>$^{123}$</td>
</tr>
<tr>
<td><strong>Special symbols</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\textcompwordmark</td>
<td></td>
<td>(ligature break)</td>
</tr>
<tr>
<td>\textregistered</td>
<td>R</td>
<td>™</td>
</tr>
<tr>
<td>\texttrademark</td>
<td>™</td>
<td></td>
</tr>
<tr>
<td>\textcircled{x}</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

the code is doing.

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. \LaTeXXing the driver file generates the documentation. This is usually superfluous since the .dtx file itself is so constructed that
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, Sec-
tion 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 Wołinski, 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
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

\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
\askforoverwritedefalse
\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 `\preamble` and `\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 `\declarepreamble` and `\declarepostamble`, and activated with declarations `\usepreamble` and `\usepostamble`.

The instructions `\keepsilent` and `\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 `\generate` which specifies the files to be created with a series of `\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 `\keepsilent` or `\usepreamble`, but the main one is the command `\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 `\from` commands.

The created `fullpage.sty` file now contains

```
%%
%% 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’.
```

It is possible to have a master batch job to process individual ones, in which case the master must input them with `\batchinput`, and not with `\input`.

**Rules for removing lines**

The `\generate` and `\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;
D.7. Managing code and documentation

2. any lines beginning with double \texttt{!!} signs are retained;

3. any line beginning with \texttt{\%<opt>} (or \texttt{\%<+opt>}) will have the rest of its text transferred if \texttt{opt} is one of the selected options; otherwise it is removed;

4. any line beginning with \texttt{\%!opt>} (or \texttt{\%<-opt>}) will have the rest of its text transferred if \texttt{opt} is \textit{not} one of the selected options; otherwise it is removed;

5. all lines between \texttt{\%<*opt>} and \texttt{</opt>} are retained or removed depending on whether \texttt{opt} is a selected option or not.

Options can be combined logically, negated, and grouped:

\begin{verbatim}
    a\&b     (a and b);
    a|b      (a or b);
    !a       (not a);
    (a|b)&c  (c and one of a or b)
\end{verbatim}

For more information on DocStrip, see the documentation which can be obtained by processing docstrip.dtx with \LaTeX.

D.7.2 Documenting \LaTeX{} 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 \LaTeX{} styles, as well as the basic \verb|latex.tex| file, were heavily commented by Leslie Lamport, but only with straightforward normal text. It was Frank Mittelbach’s \verb|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 DocStrip utility of Section D.7.1) and another to print the documentation (the doc package). The basic idea behind this package is that the comments are in fact regular \LaTeX{} 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 \LaTeX. Such a driver for a source file named \verb|fullpage.dtx| would be, in its simplest form,

\begin{verbatim}
\documentclass{article}
\usepackage{doc}
\begin{document}
 \DocInput{fullpage.dtx}
\end{document}
\end{verbatim}
The \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 fullpage.dtx become real text to be processed!

We describe some of the extra features that the doc package makes available for the 'comments' in the .dtx files, illustrated by an example, the source file fullpage.dtx for the fullpage package presented in Section D.3.1. As usual, a more complete manual can be acquired by processing doc.dtx.

The description part

The documentation consists of two parts: the description, which is a manual for the end user, and the 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

\OnlyDescription

in the preamble.

Special commands for the description part are:

\DescribeMacro{\macro_name}
\DescribeEnvironment{\env_name}

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 \verb command frequently for printing input text, some abbreviations are provided with

\MakeShortVerb{\c} \and \DeleteShortVerb{\c}

which first turn the character c into shorthand for \verb{c}, and then restore its normal use. For example, after \MakeShortVerb{|}, |mycom| prints \mycom. (These commands can be made available for any document by loading the package shortvrb, which is actually extracted from the 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 \iffalse to form a block, or meta comment, as

\iffalse
% These lines are ignored even when the percent character is inactive
\fi

The other method is to use ^^A in place of %, a special 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 \Describexxx 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

\begin{verbatim}
  makeindex -s gind.ist filename
\end{verbatim}

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

\begin{verbatim}
  makeindex -s gglo.ist -o filename.gls filename.glo
\end{verbatim}

(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.
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{\Finalize} 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\textbackslash{}B\textbackslash{}C\textbackslash{}D\textbackslash{}E\textbackslash{}F\textbackslash{}G\textbackslash{}H. . .
. . . . . . . . . . . . . . . . Tilde \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{\Provides} command identifying the specified file (Section D.2.1). The idea is that the \texttt{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{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}, issue \texttt{\OnlyDescription} by means of \texttt{\AtBeginDocument}, and so on.

### A sample \texttt{.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)]

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

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.
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.


E.1 Converting to HTML

E.1.1 The \LaTeX{}2HTML program

The \LaTeX{}2HTML 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 \LaTeX{}2HTML 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 \LaTeX{}2HTML 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 \LaTeX{}2HTML 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 \LaTeX{}2HTML 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{}.

\LaTeX{}2HTML 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 .tex $\rightarrow$ .dvi $\rightarrow$ .ps $\rightarrow$ .pbm $\rightarrow$ .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 html.sty package. For example:

- explicit hyperlinks, internal and external
E.1. Converting to HTML

- conditional text for the HTML or \LaTeX 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.

\LaTeX2HTML is available on the CTAN servers (Figure B.4 on page 390) under the support directory. It can also be downloaded from the \LaTeX2HTML 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 \TeX4ht program

An alternative program is that by Eitan M. Gurari, called \TeX4ht. It produces much the same results as \LaTeX2HTML, 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 TeX4ht.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 \TeX4ht program; no other driver will be able to process this .dvi file.

The \TeX4ht 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 \LaTeX2HTML, 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 \TeX4ht output with a GIF representation of a math equation can be seen in Figure E.1 on page 482.

Since \TeX4ht needs to redefine all the \LaTeX markup commands, one could argue that it too is duplicating \LaTeX just like \LaTeX2HTML. 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 \TeX4ht. A package like natbib which adds new citation commands \citet and \citep will totally confuse \LaTeX2HTML without the additional Perl script natbib.per1, whereas under \TeX4ht 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.)

\TeX4ht is available from the \TeXLive CD, or from http://www.cis.ohio-state.edu/~gurari/TeX4ht/mn.html.

\LaTeX2HTML and \TeX4ht 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.

### 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.
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.

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.

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 rendering.

David Carlisle has written macros in a file xml\text{latex}.tex to act as an
XML parser for output in \TeX{} or \LaTeX{}. Sebastian Rahtz has written an
additional macro package Passive\TeX{}, to incorporate an XSL style sheet
for the TEI DTD into xml\text{latex}. 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 document.xml as
E.3 The techexplorer Hypermedia Browser

\def\xmlfile{document.xml} 
\input xmltex.tex

and then to process this with either L\TeX or pdfL\TeX. The Passive\TeX macros will be found automatically, if they are on the system.

Another means is to generate an xmltex format, as described in Section B.1.3. This would be done with

\texttt{\texttt{initex \&latex xmltex.tex}} \texttt{ or } \texttt{tex -ini \&latex xmltex.tex}

to generate a format named xmltex.fmt. One can then invoke it with

\texttt{\texttt{tex \&xmltex document.xml}}

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 \texttt{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 \LaTeX 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 \LaTeX 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 \LaTeX2HTML and \TeX4ht: it is not primarily intended for viewing arbitrary \LaTeX documents, but rather for preparing hypertext Web pages by means of \TeX or \LaTeX. The only problem here is that most
We wish to demonstrate this interesting formula, in two representations:

A) as converted to GIF by \texttt{\LaTeX4ht}:

\[
\Gamma(\theta) = \int_0^\infty \frac{\partial f(\theta, \omega, x)}{\partial x} \, dx
\]

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.

\textbf{Figure E.1}: Example of a math formula represented by \texttt{\LaTeX4ht} (as a GIF image of the original \LaTeX output) and by \texttt{techexplorer} rendering.

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 \LaTeX commands, especially mathematics, inside a regular HTML file. If the commands are stored in a file \texttt{formula.tex}, they could be embedded with

\begin{verbatim}
<EMBED SRC="formula.tex" WIDTH=300 HEIGHT=70>
\end{verbatim}

or they can be entered directly as

\begin{verbatim}
<EMBED TYPE="application/x-techexplorer"
TEXDATA="$$\Gamma(\theta)=\int_0^\infty \frac{\partial f(\theta, \omega, x)}{\partial x} \, dx$$" WIDTH=300 HEIGHT=70>
\end{verbatim}

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 \texttt{\LaTeX4ht} 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\epsilon 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\epsilon; 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\epsilon 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.

\section*{F.1 The 2.09 preamble}

As for \LaTeX{} 2\epsilon, the document preamble in \LaTeX{} 2.09 contains overall specifications for the entire document, including general layout.

\subsection*{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

$$\texttt{\documentstyle[options]{style}}$$
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\epsilon 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 .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\epsilon 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\epsilon 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\epsilon, in compatibility mode, it first tries to load the main style as a class file with the extension .cls, in this case article.cls, but if that fails, it then looks for the extension .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\epsilon.

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\epsilon 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 \{\bf bold face\} to yield \textbf{bold face}.

The emphasizing declaration \em also belongs to this group, but it has been taken over as part of \LaTeX\ 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}} \Rightarrow \textit{slanted bold}
{\slshape slanted \bfseries bold}} \Rightarrow \textit{slanted 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 classes, and not just in compatibility mode. Note carefully that wording: they are not part of \LaTeX\ 2 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, these attributes remain unchanged. Compare the results of \{\sf slanted \Large larger\}:

\begin{verbatim}
\LaTeX\ 2.09: slanted larger
\LaTeX\ 2: slanted larger
\end{verbatim}

In \LaTeX\ 2 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, 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\,\epsilon 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{\documentclass} in place of \texttt{\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: LaTeX2e.

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}
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.

\section*{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 em and 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.

.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 METAFONT program (Section G.3) as needed.

Since these files might exist several times for a given font, in difference sizes or resolutions, many installations distinguish them by adding the resolution to the extension, as cmr10.300pk or cmr10.600pk for 300 and 600 dpi versions, respectively. On systems that cannot support extensions with more than three characters, both files are named cmr10.pk but are stored in directories dpi300 and dpi600.

Font magnification is achieved by selecting an appropriate resolution. If cmr10 is to be printed on a 300 dpi printer at double its normal size, the 600 dpi version is used instead.

-mf METAFONT source files contain the drawing instructions for each of the fonts. These are not read by the DVI driver but rather by the METAFONT program which converts them into .pk bitmap files at the required size and resolution.

-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.

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 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 fonts begin with the letters \texttt{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 \texttt{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 \texttt{cmti*} for the italic upright font. Why is this not \texttt{cmri*}? The answer is that this is a text italic font, as opposed to the math italic font \texttt{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.
### Appendix G. TeX Fonts

<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>(\Pi)</td>
</tr>
<tr>
<td>'02x</td>
<td>1</td>
<td>16</td>
<td>J</td>
<td>17</td>
<td>(\ddot{\imath})</td>
<td>18</td>
<td>(\ddot{\imath})</td>
</tr>
<tr>
<td>'03x</td>
<td>(\epsilon)</td>
<td>24</td>
<td>(\beta)</td>
<td>25</td>
<td>(\alpha)</td>
<td>26</td>
<td>(\alpha)</td>
</tr>
<tr>
<td>'04x</td>
<td>(\iota)</td>
<td>32</td>
<td>(\iota)</td>
<td>33</td>
<td>(\pi)</td>
<td>34</td>
<td>(\pi)</td>
</tr>
<tr>
<td>'05x</td>
<td>( )</td>
<td>40</td>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</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 1: The character font cmr10. This is the standard character assignment for the 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>(\Pi)</td>
</tr>
<tr>
<td>'02x</td>
<td>1</td>
<td>16</td>
<td>J</td>
<td>17</td>
<td>(\ddot{\imath})</td>
<td>18</td>
<td>(\ddot{\imath})</td>
</tr>
<tr>
<td>'03x</td>
<td>(\epsilon)</td>
<td>24</td>
<td>(\beta)</td>
<td>25</td>
<td>(\alpha)</td>
<td>26</td>
<td>(\alpha)</td>
</tr>
<tr>
<td>'04x</td>
<td>(\iota)</td>
<td>32</td>
<td>(\iota)</td>
<td>33</td>
<td>(\pi)</td>
<td>34</td>
<td>(\pi)</td>
</tr>
<tr>
<td>'05x</td>
<td>( )</td>
<td>40</td>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</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 cmrsc10. 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.
G.2. Computer Modern fonts

Font Layout 3: The character font \texttt{cmti10}. The only difference from Layout 1 is symbol 36 (\£ instead of \$). All other \textit{text italic} fonts have this same pattern.

Font Layout 4: The character font \texttt{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 \$. 
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 \AMS fonts in Section 12.4.1 add additional bold face sizes for \texttt{cmmib*} and \texttt{cmbsy*} as well as some more sizes for \texttt{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, \texttt{cmff10} leaning to the left and \texttt{cmfi10} which leans to the right. Both are series \texttt{m} with shapes \texttt{n} and \texttt{i} respectively.

**cmfib** *Fibonacci* family, contains one font \texttt{cmfib8} derived from the Fibonacci series of numbers. It has series \texttt{m} and shape \texttt{n}.

**cmdunh** *Dunhill* family, with one font \texttt{cmdunh10} with series \texttt{m} and shape \texttt{n}. This font is demonstrated in the sample letterhead on page 364.

### Logo fonts

The fonts \texttt{logo8}, \texttt{logo9}, \texttt{logo10}, \texttt{logosl10}, and \texttt{logobf10} contain only the nine letters A, E, F, M, N, O, P, S, and T, for generating the logos

\begin{verbatim}
METAFont METAFont METAFont
METAPOST METAPOST METAPOST
\end{verbatim}

### The \LaTeXasy fonts

*Package: \latexsym, amsfonts*  

As an extension of the \texttt{cmsy*} fonts for mathematical symbols, \LaTeX provides some additional symbols with the font \texttt{lasy*} in design sizes 5–10 pt. It contains the 15 symbols: \texttt{〈〈〉〉〈〈} \texttt{〉〉〈〈} \texttt{〉〉〈〈} \texttt{□□□□□□}.

These symbols are not defined unless one of the packages \texttt{latexsym} or \texttt{amsfonts} has been loaded.

### Fonts for making pictures

The special picture elements for use in the \LaTeX picture environment are stored in fonts named \texttt{line10}, \texttt{lcircle10}, \texttt{linew10}, and \texttt{lcirclew10}. The first two are used for lines, ovals, and circles when \texttt{\thinlines} is in effect. The sloping lines and arrow heads are to be found in \texttt{line10}
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>'00x</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>0</td>
<td>Γ</td>
<td>Δ</td>
<td>Θ</td>
<td>Π</td>
<td>Δ</td>
<td>Σ</td>
<td>Τ</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>ω</td>
<td>ψ</td>
<td>Ψ</td>
<td>Φ</td>
<td>Ω</td>
<td>Ψ</td>
<td>Φ</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>ζ</td>
<td>η</td>
<td>θ</td>
<td>ι</td>
<td>κ</td>
<td>λ</td>
<td>μ</td>
<td>ν</td>
</tr>
<tr>
<td>3</td>
<td>ϵ</td>
<td>ζ</td>
<td>θ</td>
<td>ι</td>
<td>κ</td>
<td>λ</td>
<td>μ</td>
<td>ν</td>
</tr>
<tr>
<td>4</td>
<td>α</td>
<td>β</td>
<td>γ</td>
<td>δ</td>
<td>ε</td>
<td>ζ</td>
<td>θ</td>
<td>ι</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</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 A ... 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.

#### Font Layout 8: The character font `wncy10`, one of the Cyrillic fonts from the University of Washington. Its coding scheme is designated OT2.

while the circles and oval segments (Section 13.1.4) are in `lcircle10`. The second pair of fonts with the added `w` contain thicker lines, for use with `\thicklines`.

### G.2.3 \AMS Cyrillic fonts

The Cyrillic fonts that are part of the `amsfonts` collection are described in Section 12.4.2. They conform to the font encoding scheme OT2, displayed in Layout 8. They are available as upright (`wncy*`), bold (`wncyb*`), small caps (`wncysc*`), and upright sans serif (`wncyss*`) fonts. As for the CM fonts, `*` represents the design size in points.

The NFSS system assigns these fonts to families `cmr` and `cmss`; that is, they are treated as members of the Computer Modern set. This is not as absurd as it sounds since they are intended to be used with the CM fonts.
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.

\section{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, 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 TeX or \LaTeX.

Fashioning diacritical characters as a combination of letters and special symbols has one great disadvantage for the 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 TeX with the CM fonts (Section 2.5.6). However, other special letters, such as ñ, ñ, Þ, þ, and ð, are missing completely and cannot be easily constructed from existing ones.

G.4.2 The Cork proposal

At the 1990 International 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 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 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
```
<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>`7</td>
<td>`6</td>
<td>`5</td>
<td>`4</td>
<td>`3</td>
<td>`2</td>
<td>`1</td>
<td>`0</td>
</tr>
<tr>
<td>‘7</td>
<td>‘6</td>
<td>‘5</td>
<td>‘4</td>
<td>‘3</td>
<td>‘2</td>
<td>‘1</td>
<td>‘0</td>
</tr>
<tr>
<td>’7</td>
<td>’6</td>
<td>’5</td>
<td>’4</td>
<td>’3</td>
<td>’2</td>
<td>’1</td>
<td>’0</td>
</tr>
<tr>
<td>&quot;7</td>
<td>&quot;6</td>
<td>&quot;5</td>
<td>&quot;4</td>
<td>&quot;3</td>
<td>&quot;2</td>
<td>&quot;1</td>
<td>&quot;0</td>
</tr>
<tr>
<td>&quot;7</td>
<td>&quot;6</td>
<td>&quot;5</td>
<td>&quot;4</td>
<td>&quot;3</td>
<td>&quot;2</td>
<td>&quot;1</td>
<td>&quot;0</td>
</tr>
<tr>
<td>’0x</td>
<td>’1x</td>
<td>’2x</td>
<td>’3x</td>
<td>’4x</td>
<td>’5x</td>
<td>’6x</td>
<td>’7x</td>
</tr>
<tr>
<td>ß</td>
<td>å</td>
<td>ä</td>
<td>ë</td>
<td>è</td>
<td>â</td>
<td>û</td>
<td>ø</td>
</tr>
<tr>
<td>0x</td>
<td>1x</td>
<td>2x</td>
<td>3x</td>
<td>4x</td>
<td>5x</td>
<td>6x</td>
<td>7x</td>
</tr>
<tr>
<td>ø</td>
<td>ð</td>
<td>ø</td>
<td>ð</td>
<td>ø</td>
<td>ð</td>
<td>ø</td>
<td>ð</td>
</tr>
<tr>
<td>0x</td>
<td>1x</td>
<td>2x</td>
<td>3x</td>
<td>4x</td>
<td>5x</td>
<td>6x</td>
<td>7x</td>
</tr>
<tr>
<td>240</td>
<td>232</td>
<td>224</td>
<td>216</td>
<td>208</td>
<td>200</td>
<td>192</td>
<td>184</td>
</tr>
</tbody>
</table>

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 ecxxn
nn, where xx represents two letters specifying the font characteristics, and nnnn the design size in hundredths of points. Thus ecrm1000 is the upright Roman font in size 10 points.

The \METAFONT source files are available for the following extended fonts (without the size specification):

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ecrm</td>
<td>ecrb</td>
<td>eccc</td>
<td>ecci</td>
<td>ecvi</td>
<td>ecss</td>
<td>ecdh</td>
</tr>
<tr>
<td>ecbx</td>
<td>ecti</td>
<td>ecxc</td>
<td>ectt</td>
<td>ecvi</td>
<td>ecsi</td>
<td></td>
</tr>
<tr>
<td>ecb1</td>
<td>ecui</td>
<td>ecsc</td>
<td>ecit</td>
<td>ectl</td>
<td>ecsx</td>
<td></td>
</tr>
<tr>
<td>ecrb</td>
<td>ecbi</td>
<td>ecoc</td>
<td>ecvt</td>
<td>ects</td>
<td>ecsx</td>
<td></td>
</tr>
</tbody>
</table>

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, ecbx* is the extended bold font corresponding to the CM font 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:

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0500</td>
<td>0600</td>
<td>0700</td>
<td>0800</td>
<td>0900</td>
<td>1000</td>
<td>1095</td>
</tr>
<tr>
<td>1200</td>
<td>1440</td>
<td>1728</td>
<td>2074</td>
<td>2488</td>
<td>2986</td>
<td>3583</td>
</tr>
</tbody>
</table>

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 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.

### 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.
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 Š = 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}`: ℹ
- special letters `\DH = Đ`, `\DJ = Đ`, `\NG = Đ`, `\TH = Þ`
  `\dh = đ`, `\dj = đ`, `\ng = ñ`, `\th = ð`
- special symbols `\guillemotleft = «`, `\guillemotright = »`
  `\guilsinglleft = ‹`, `\guilsinglright = ›`
  `\textquotedblbase = "`, `\quotesinglbase = ’`

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>Name</th>
<th>Description</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>pcrro</td>
<td>Courier</td>
</tr>
<tr>
<td>pcrro</td>
<td>Courier</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>
</tbody>
</table>

Table G.2: Root names of the 35 standard PostScript fonts
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 \LaTeX\ text encoding</td>
<td>OT1</td>
<td>492</td>
</tr>
<tr>
<td>7m</td>
<td>\LaTeX\ math italics encoding</td>
<td>OML</td>
<td>495</td>
</tr>
<tr>
<td>7v</td>
<td>\LaTeX\ math extension encoding</td>
<td>OMX</td>
<td>496</td>
</tr>
<tr>
<td>7y</td>
<td>\LaTeX\ 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 \texttt{tm} for Times-Roman. Next come various letters to specify weight, for example \texttt{r} for regular (upright Roman) or \texttt{b} for bold, and variant, like \texttt{i} for italic, \texttt{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 \texttt{ptmr7t} and \texttt{ptmr8t} are both constructed from the same raw font \texttt{ptmr8r}, differing only in the character assignments. There are also raw fonts that are modifications of the basic fonts. For example, \texttt{ptmro8t} is an oblique Times-Roman virtual font, based on \texttt{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 \texttt{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 \texttt{pdflatex}, \texttt{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 \texttt{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 \texttt{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 \texttt{bsr.map} from \texttt{texmf/dvips/bluesky/} and add it to the list of mapping files in \texttt{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 \texttt{bsr-interpolated.map}, which substitutes the missing sizes with the nearest approximation, or employ the \texttt{bakoma} fonts that fill the gaps, with the map file \texttt{bakomaextra.map}. The \TeX\ Live CD does not include the complete \texttt{bakoma} set, but only those fonts that are missing in the Bluesky ensemble.

Together Bluesky and \texttt{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 \texttt{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 \texttt{METAFONT.mf} source files and should therefore produce exactly the same results as the corresponding pixel fonts.

To use them, one must install the \texttt{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.
This appendix contains a brief description of all the \LaTeX commands, in alphabetical (ASCII) order, neglecting the leading backslash \, 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-LaTeX;
- [p] those allowed only in the preamble.

\begin{itemize}
  \item[\textbackslash\ ] \hfill (2.1), (2.7.1) – 19, 29
    \begin{flushright}
    Normal space between words after a command without arguments or after a period that is not the end of a sentence.
    \end{flushright}

  \item[\!]
    \hfill (9.4.2) – 226
      \begin{flushright}
      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’.
      \end{flushright}

  \item[\!\textbackslash m]
    \hfill (2.5.6) – 24
      \begin{flushright}
      Produces \textbackslash i.
      \end{flushright}

  \item[\textbackslash ! \[m]]
    \hfill (5.5.1) – 145
      \begin{flushright}
      In math mode, a negative space of −1/6 quad: xx\textbackslash!x = xxx.
      \end{flushright}
\end{itemize}
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 Bi\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 = à.
H.1. Brief description of the \LaTeX\ commands

2. Within the tabbing environment, a command to jump to the end of the current column.

( )  
For a picture command in picture environment, specifies a coordinate pair. In \TeX, an alternative form for the outermost grouping of the entry type.

\( \)  
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.

\( \)  
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.

\( \)  
Within the tabbing environment, increments the left margin by one tab stop (moves it to the right).

\( \)  
Small space, the size of 1/6 quad, for use in text and math mode: xx\,x = xx\ x.

\( \)  
As --, produces the hyphen - for compound words and word division, as --_, the en dash –, and as ---, the em dash —.

\( \)  
Within the tabbing environment, decrements the left margin by one tab stop (moves it to the left).

\( \)  
Command to produce a dot accent: \texttt{\.o} = ˙o.

\( \)  
Command to break up ligatures: shelf\textbackslash f\textbackslash ul = shelfful.

\( \)  
In math mode, a medium space, the size of 2/9 quad: xx\,:\,x = xx\ x.

\( \)  
In math mode, a large space, the size of 5/18 quad: xx\;\,x = xx\ x.
Appendix H. Command Summary

\(<\) ............... (4.6.3) – 82
Within the tabbing environment, moves to the left by one tab stop.

\(=\) ............... (2.5.7), (4.6.1) – 24, 81
1. Command to produce a macron accent: \(\alpha = \bar{\alpha}\).
2. Within the tabbing environment, sets a tab stop at the current position within the line.

\(>\) ............... (4.6.1) – 81
Within the tabbing environment, moves right to the next tab stop.

\(?\) produces \(?\) ............... (2.5.6) – 24

@ .................. (9.4.2), (14.2) – 226, 311
1. In MakeIndex, separates an entry in an \(\text{index}\) command into a lexical (for alphabetization) and printing part. Example: \(\text{index}{\text{sum}@\Sigma}\) means that the entry appears in the index at the location of the word 'sum' but what is printed is the summation sign \(\Sigma\).
2. In \LaTeX, denotes the entry type. Example: @BOOK indicates that the following literature entries correspond to those of a book.

\(@\) .................. (2.7.1) – 29
Extra space at the end of a sentence ending with a capital letter.

[ ] .................. (2.1) – 18
After commands or environment calls, specifies an optional argument.

\[\] .................. (5.1) – 120
Switches from text mode to displayed math mode for putting a formula on a line by itself. Has the same effect as \(\text{begin}\{\text{displaymath}\}\) or $$ in text mode.

\[[space]\] .................. (2.7.2) – 31
Ends the current line (without right justifying it). The optional argument [space] inserts additional vertical spacing of length space before the next line.

\[[space]\] .................. (2.7.2) – 31
The same as \(\text{\textbackslash{}\textbackslash{}}}\) but prevents a page break from occurring between the current and next line.

\] .................. (5.1) – 120
Switches back from displayed math mode to text mode. Has the same effect as \(\text{end}\{\text{displaymath}\}\) or $$ in math mode.
H.1. Brief description of the \LaTeX\ commands

\[ \hat{m} \] \hfill (5.2.2) – 121
Exponents and superscripts in equations: \(x^2 = x^2, \ x^{-2n} = x^{-2n}\).

\[ \hat{\cdot} \] \hfill (2.5.7) – 24
Command to produce a circumflex accent: \( \hat{o} = \hat{o} \).

\[ \_ [m] \] \hfill (5.2.2) – 121
Subscripts in equations: \(a_n = a_n, \ a_{i,j,k} = a_{i,j,k}\).

\[ \underline{\cdot} \] \hfill (2.5.4) – 23
Command to produce the underbar sign: \( t_\nu = t_\nu \).

\[ \prime \] \hfill (2.5.7), (4.6.4) – 24, 83
1. Command to produce a grave accent: \( \prime o = \prime o \).
2. Within the tabbing environment, pushes the following text up against the right margin of the line.

\[ \{ \} \] \hfill (2.1), (2.2), (14.2) – 18, 20, 311
1. After a command or environment call, specifies a mandatory argument.
2. Grouping a section of text to create a nameless environment.
3. In \TeX, delimiting the name of an entry type, as well as an alternative delimiter for the text field.

\[ \{ \] \hfill (2.5.4) – 23
produces a left curly brace: \( \{ = \{ \) \( \) \( \) \) \( \).

\[ | [m] \] \hfill (5.3.4), (5.4.1) – 126, 132
produces | \hfill (5.3.6) – 127
In MakeIndex, the command character within a \index\ command.

1. After the command \newcommand{\ii}[1]{\textit{#1}} has been defined, \index{entry|ii} produces the page number for ‘entry’ in the index in italic type.
2. The cross-reference command \see from makeidx.sty can be invoked with \index{bison|see{buffalo}} to produce cross-references within the index.

\[ \| [m] \] \hfill (5.3.6) – 127
produces || \hfill (2.7.1) – 29
A normal space between words, but without the possibility that the line will be broken there. Example: Prof. '\Jones ensures that 'Prof.' and 'Jones' both remain on the same line.

\[ \hat{\cdot} \] \hfill (2.7.1) – 29
Command to produce a tilde accent: \( \hat{n} = \hat{n} \).
\a= . . . . . . . . . . . . . . . . . . . . . . . . . . . (4.6.4) – 83

Produces a macron accent within tabbing environment: \a=o = ò.

\a’ . . . . . . . . . . . . . . . . . . . . . . . . . . . (4.6.4) – 83

Produces an acute accent within tabbing environment: \a’o = ó.

\a‘ . . . . . . . . . . . . . . . . . . . . . . . . . . . (4.6.4) – 83

Produces a grave accent within tabbing environment: \a‘o = ò.

\AA produces À . . . . . . . . . . . . . . . . . . (2.5.6) – 24
\aa produces â . . . . . . . . . . . . . . . . . . (2.5.6) – 24
\above displayskip [m] . . . . . . . . . . . . . (5.5.4) – 149

Vertical space between a long displayed equation and the preceding
line of text. A new value may be assigned with the \setlength
command:
\setlength{\above displayskip}{10pt plus2pt minus5pt}

\above displayshortskip [m] . . . . . . . . . . . . . (5.5.4) – 149

Vertical space between a short displayed equation and the preceding
line of text. A new value may be assigned with the \setlength
command as in the above example.

\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.

\acute{x} [m] . . . . . . . . . . . . . . . . . . . . . (5.3.9) – 129

Acute accent over math variable x: \acute{a} = ´a

\Acute{x} [m][a] . . . . . . . . . . . . . . . . . . . . (12.2.2) – 263

With the amsmath package, can be used like \acute, but with multiple
AMS-LaTeX math accents the positioning will be correct.

\addcontentsline{file}{format}{entry} . . . . (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, to be formatted as the heading of a sectioning
command, as given by format, for example
\addcontentsline{toc}{section}{References}

\address{sender} . . . . . . . . . . . . . . . . . . . . . (16.1) – 351

In the letter document class, enters the sender’s address. Multiple
lines in sender are separated by \.

\addtime{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 com-
mand adds the specified number of seconds to the marker. See also
\settime.
H.1. Brief description of the \LaTeX\ commands  

\addtocontents{file}{entry}  . . . . . . . . . . . . . (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}  . . . . . . . . . . . . . (8.1.3) - 182

Adds number to the current value of the number stored in counter.

\addtolength{length}{length}  . . . . . . . . . . . . . . . . . . . (8.2) - 184

Adds the quantity length to the current value of the length command \lengthname.

\addvspace{length}  . . . . . . . . . . . . . . . . . . . . . . . . (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 \AE . . . . . . . . . . . . . . . . . . . . . . . . . . (2.5.6) - 24
\ae\ produces \ae . . . . . . . . . . . . . . . . . . . . . . . . . . (2.5.6) - 24
\aleph\[m\]\ produces \aleph . . . . . . . . . . . . . . . . . . . (5.3.6) - 127
\allowdisplaybreaks[\num] \[p] \[a]  . . . . . . . . . . . . . . . . . . . . . . . . . . (12.2.7) - 278

With the amsmath package, allows automatic page breaks to occur within multiline math formulas. If 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 \displaybreak.

\Alph{\counter}  . . . . . . . . . . . . . . . . . . . . . . . . . . (8.1.4) - 183

Prints the current value of \counter as a capital letter.

\alph{\counter}  . . . . . . . . . . . . . . . . . . . . . . . . . . (8.1.4) - 183

Prints the current value of \counter as a lower case letter.

\alpha\[m\]\ produces \alpha . . . . . . . . . . . . . . . . . . . (5.3.1) - 125
\alsoname  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (D.4.1) - 460

Command for use in the makeidx package. It prints the text for a command \seealso. In English, this is 'see also' but may be altered for adaptation to other languages.

\amalg\[m\]\ produces \amalg . . . . . . . . . . . . . . . . . . . (5.3.3) - 125
\and  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (3.3.1) - 53

Used to separate author names within the \author command for generating a title page with \maketitle.

\angle\[m\]\ produces \angle . . . . . . . . . . . . . . . . . . . (5.3.6) - 127
\appendixname  . . . . . . . . . . . . . . . . . . . . . . . . . . (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.
Appendix H. Command Summary

\approx [m] \text{ produces } \approx \quad \text{(5.3.4) - 126}
\arabic{counter} \quad \text{ (8.1.4) - 183}

Prints the current value of \textit{counter} as an Arabic number.

\arccos [m] \quad \text{(5.3.8) - 128}
Command to produce the function name 'arccos' in equations.

\arcsin [m] \quad \text{(5.3.8) - 128}
Command to produce the function name 'arcsin' in equations.

\arctan [m] \quad \text{(5.3.8) - 128}
Command to produce the function name 'arctan' in equations.

\arg [m] \quad \text{(5.3.8) - 128}
Command to produce the function name 'arg' in equations.

arraycolsep \quad \text{(4.8.2) - 98}
Half the width of the intercolumn spacing in the \texttt{array} environment.
Value is assigned with the \LaTeX\ command \setlength:
\setlength{\arraycolsep}{3mm}

arrayrulewidth \quad \text{(4.8.2) - 98}
The thickness of vertical and horizontal lines in the \texttt{array} and \texttt{tabular} environments. Its value is assigned to a length with \setlength:
\setlength{\arrayrulewidth}{0.5mm}

\arraystretch \quad \text{(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 \renewcommand{\arraystretch}{factor}.

\ast [m] \text{ produces } \ast \quad \text{(5.3.3) - 125}
\asymp [m] \text{ produces } \asymp \quad \text{(5.3.4) - 126}
\AtBeginDocument\{\texttt{code}\} \[p] \quad \text{(D.2.4) - 444}

Stores the \texttt{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{code}. A package might include coding in this way to ensure that it is not overwritten by another package.

\AtEndDocument\{\texttt{code}\} \[p] \quad \text{(D.2.4) - 444}

Stores the \texttt{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 \textit{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 \textit{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 \texttt{\maketitle} command.

\b{x} ......................................................... (2.5.7) – 24

Command to produce an underbar accent: $\backslash b\{o\} = \bar{o}$.

\backmatter .................................................. (3.3.5) – 57

In the \texttt{book} class, introduces the material that comes at the end (bibliography, index) by turning off the chapter numbering of the \texttt{chapter} command.

\backslash [m] produces \ 
................................................... (5.3.6) – 127

\bar{x} [m] .................................................. (5.3.9) – 129

Macron accent over the math variable $x$: $\bar{a}$.

\Bar{x} [m][a] .............................................. (12.2.2) – 263

With the \texttt{amsmath} package, can be used like \texttt{\bar}, but with multiple \texttt{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 \texttt{\setlength}:

\begin{verbatim}
\setlength{\baselineskip}{12pt plus2pt minus1pt}
\end{verbatim}

\baselinestretch ........................................ (3.2.4), (4.1.2) – 46, 63

A factor with the normal value of 1 by which the internal length \texttt{\baselineskip} is multiplied to produce the actual interline spacing. May be changed with:

\begin{verbatim}
\renewcommand{\baselinestretch}{factor}
\end{verbatim}

The new value takes effect after the next change in font size!
Appendix H. Command Summary

\begin{envrnmnt} \ldots \ldots (2.2) - 19
Start of an environment with the name \texttt{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} \ldots (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} \ldots (12.2.6) - 274
With the \texttt{amsmath} package, switches to displayed math mode to produce a set of aligned equations. Line are terminated by \texttt{\\}} commands. The lines are split into columns aligned on the first, third, fifth \ldots& characters. Each line receives an equation number unless the *-form of the environment has been selected.

\begin{alignat}{num} \ldots (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} \ldots (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} \ldots (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} \ldots (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 character 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} \ldots (4.2.1) - 67
Start of the environment \texttt{center}. Each line of text terminated by \texttt{\\} appears centered. See also \texttt{\centering}.
\begin{command_name} \hspace{0.4\textwidth} (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 \small until the countercommand \end{small} is given.

\begin{alltt} \hspace{0.4\textwidth} (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} \hspace{0.4\textwidth} (12.2.4) - 266
Like the matrix environment, but enclosed in square brackets [].

\begin{Bmatrix} \hspace{0.4\textwidth} (12.2.4) - 266
Like the matrix environment, but enclosed in curly braces {}.

\begin{cases} \hspace{0.4\textwidth} (12.2.6) - 277
With the amsmath package, writes math expressions on several lines, terminated by \( \backslash \), in left-justified columns, separated by & , with a curly brace enclosing all lines at the left, the whole being centered vertically.

\begin{description} \hspace{0.4\textwidth} (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[\textit{label}].

\begin{displaymath} \hspace{0.4\textwidth} (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} \hspace{0.4\textwidth} (1.5.2) - 12
Start of the outermost environment of a text document. This command terminates the preamble. It is obligatory for every \LaTeXX document, as is its counterpart \end{document} for ending the document.

\begin{enumerate} \hspace{0.4\textwidth} (4.3.2) - 70
Start of the environment enumerate to produce a numbered, indented 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} \hspace{0.4\textwidth} (5.4.7) - 138
Switches from text to displayed math mode to produce a set of equations or a multiline formula in the form of a three-column table \{ rcl \}. The individual lines of the formula are ended with the command \( \backslash \); 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*} \[m\] . . . . . . . . . . . . . . . . (5.4.7) - 138 
Is the same as the eqnarray environment except that no equation numbers are printed.

\begin{equation} \[m\] . . . . . . . . . . . . . . . . . . (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.

\begin{falign} \[a\] . . . . . . . . . . . . . . . . . . (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.

\begin{figure} \[loc\] . . . . . . . . . . . . . . . . . . (7.1) - 169 
Float environment for entering text for a figure. The optional argument \texttt{loc} can be any combination of the letters \texttt{h, t, 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{figure*} \[loc\] . . . . . . . . . . . . . . . . . . (7.1) - 169 
The same as the \texttt{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.

\begin{filecontents}{\texttt{file name}} \[p\] . . . . . . . . (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.

\begin{filecontents*}{\texttt{file name}} \[p\] . . . . . . . . (D.2.9) - 448 
Is the same as the \texttt{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.

\begin{flushleft} . . . . . . . . . . . . . . . . . . . (4.2.2) - 67 
Start of the \texttt{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}.

\begin{flushright} . . . . . . . . . . . . . . . . . . . (4.2.2) - 67 
Start of the \texttt{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  519

\begin{gather} [a] \quad \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (12.2.6) - 273
With the amsmath package, switches to displayed math mode to produce several lines of equations, all centered with no alignment. Lines are terminated by $\backslash \backslash$ commands. Each line receives an equation number, unless the *-form of the environment has been selected.

\begin{gathered} \[pos\] [m][a] \quad \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (12.2.6) - 276
With the amsmath package, is like the gather environment but is used as an element within math mode. The optional argument \textit{pos} determines the vertical positioning relative to neighboring elements: \texttt{t} or \texttt{b} for top or bottom, no argument for centering.

\begin{itemize} \item \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (4.3.1) - 69
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 $\bullet$ generated by each \texttt{item} command.

\begin{longtable} \item \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (4.8.4) - 108
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}{\textit{recipient}} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (16.1) - 352
Start of a letter with the document class \texttt{letter}. Name and address of the \textit{recipient} are given within the second pair of brackets; lines of text within this argument are ended with the command $\backslash$.

\begin{list}{\textit{standard\_label}}{\textit{list\_decl}} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (4.4) - 74
Start of a generalized list environment. The label is defined by \textit{standard\_label}, which is generated by each \texttt{item} command. The desired list declarations are contained in \textit{list\_decl} (see page 75).

\begin{lrbox}{\textit{boxname}} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (4.7.1) - 87
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\{boxname\}}. The contents of the box may be printed as often as desired with \texttt{\usebox\{boxname\}}.

\begin{math} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (5.1) - 119
Switches from text to in-line math mode to produce formulas within a line of text. This environment has the same effect as $\backslash( \or$ $\backslash$ in text mode.

\begin{matrix} [m][a] \quad \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (12.2.4) - 266
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{verbatim}
\begin{minipage}[pos][height][inner_pos]{width}
\end{verbatim}

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{verbatim}
\begin{multicols}{num_cols}[header][pre_space]
\end{verbatim}

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{\columnssep} and \texttt{\columnsprule}.

With the starred version \texttt{multicols*}, the columns of text are not balanced on the last page of the environment.

\begin{verbatim}
\begin{multline}
\end{verbatim}

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{verbatim}
\begin{note}
\end{verbatim}

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 521

\begin{overlay} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (15.1.2) - 325

In 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}}) \ldots \ldots (13.1.2) - 288

Environment to generate a picture with the width \texttt{x\_dimen} and height \texttt{y\_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 picture environment. The picture is displaced to the left by \texttt{x\_offset} and downwards by \texttt{y\_offset}.

\begin{pmatrix} \ldots \ldots \ldots \ldots (12.2.4) - 266

Like the matrix environment, but enclosed in round parentheses ()

\begin{quotation} \ldots \ldots \ldots \ldots (4.2.3) - 67

Start of the 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} \ldots \ldots \ldots \ldots (4.2.3) - 67

The same as the quotation environment except that the first line of a paragraph is not indented but instead additional line spacing comes between paragraphs.

\begin{slide} \ldots \ldots \ldots \ldots (15.1.2), (15.2.1) - 325, 331

In slides and seminar classes, the main environment for producing a slide.

\begin{sloppypar} \ldots \ldots \ldots \ldots (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} \ldots \ldots \ldots \ldots (12.2.6) - 272

With the 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 centertags (default) or with tbtags it appears at the right of the last line, or at the left of the first line, depending on class options reqno (default) and leqno, respectively.
\begin{subarray}{pos}
\{1st line\}..\{last line\} \m{a} \quad (12.2.2) - 261

With the \texttt{amsmath} package, sets multiline text for superscripts and
subscripts, like \texttt{\substack}, but the parameter \texttt{pos} may take on values
c or \texttt{l} for centered or left-justified lines.

\begin{subequations} \quad (12.2.7) - 277

With the \texttt{amsmath} package, numbers equations within it with a fixed
main number and sequence of lower case letters attached, as \texttt{7a}, \texttt{7b},
\texttt{7c}, ...

\begin{tabbing}
\begin{tabbing}
. . . . . . . . . . . . . . . . . . . . \quad (4.6.1) - 81
\end{tabbing}

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] \quad (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] \quad (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} \quad (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\}}
for a column of width \texttt{wd} in which the text may extend over several
lines.

When the entry \texttt{@\{text\}} 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 \& and the
rows are terminated by \texttt{\\}.
\begin{tabular*}{\textwidth}[pos]{cols} ................................ (4.8.1) - 95

The same as \texttt{\begin{tabular}} except that the total width of the table is given by the argument \textit{width}. This may only be achieved successfully if there is rubber spacing between the columns. This may be added with @\texttt{\{extracolsep\|fill\}} somewhere within the \texttt{cols} format definition.

\begin{thebibliography}{sample\label} ............ (9.3.1) - 217

Environment to generate a list of literature references. The \textit{sample\label} is the longest reference marker that will appear. Each entry in the bibliography starts with the command \texttt{\bibitem} which prints the marker for that entry; lines after the first are indented by an amount equal to the width of \textit{sample\label}.

\begin{theindex} ................................... (9.4.1) - 225

Environment to produce a keyword index in two-column format. Entries are made with the \texttt{\item}, \texttt{\subitem}, \texttt{\subsubitem}, or \texttt{\indexspace} commands.

\begin{theorem\type}[extra\title] .................. (4.5) - 80

Environment to invoke a theorem-like structure that has previously been defined by the user with the \texttt{\newtheorem} command. The name of the environment, \textit{theorem\type}, something like \texttt{theorem} or \texttt{axiom}, is the first argument of the \texttt{\newtheorem} command. The \textit{extra\title} is text that added after the name and number of the structure in () parentheses.

\begin{titlepage} .................................... (3.3.1) - 52

Environment to produce a title page without a page number. The user has total control over the composition of this page.

\begin{trivlist} .................................... (4.4.5) - 79

Environment to generate a trivial list \textit{without} a sample label and list declarations. The parameters \texttt{\leftmargin}, \texttt{\labelwidth}, and \texttt{\itemsep} are all set to 0 pt while \texttt{\listparindent = \parindent} and \texttt{\parsep = \parskip}.

\begin{verbatim} .................................. (4.9) - 110

Environment to print out source text, that is, as from a typewriter. Blank lines, line breaking, and commands are all output literally without any interpretation or formatting.

\begin{verbatim*} ................................. (4.9) - 110

The same as the \texttt{verbatim} environment except that blanks are printed as \_\_ to make them visible.

\begin{verse} ..................................... (4.2.4) - 68

Environment for setting rhymes, poems, verses, etc. Stanzas are separated by blank lines, individual lines by the \texttt{\\}} command.
\begin{vmatrix} \end{vmatrix} \quad (12.2.4) - 266
Like the matrix environment, but enclosed in vertical lines \|\|.

\begin{Vmatrix} \end{Vmatrix} \quad (12.2.4) - 266
Like the matrix environment, but enclosed in double vertical lines \|\|.

\belowdisplayskip \quad (5.5.4) - 149
Vertical spacing between a long displayed formula and the following text. A new value may be assigned with the \setlength command:
\setlength{\belowdisplayskip}{\abovedisplayskip}
sets \belowdisplayskip to the same value as \abovedisplayskip.
See further examples under \abovedisplayskip.

\belowdisplayshortskip \quad (5.5.4) - 149
Vertical spacing between a short displayed formula and the following text. Value is set with the \setlength command as in the above example.

\beta \quad (5.3.1) - 125
\bfdefault \quad (A.3.1) - 372
This command defines the series attribute that is selected with the \bfseries command. It may be redefined with \renewcommand:
\renewcommand{\bfdefault}{b}
\bfseries \quad (4.1.3), (A.2) - 64, 371
This declaration switches to a font in the current family and shape, but with the bold series attribute.

\bibitem[\textit{label}]{\textit{key}} \textit{entry_text} \quad (9.3.1), (9.3.4) - 217, 221
Command to enter the text for a literature reference in the \thebibliography environment. The reference word \textit{key} is used in the main body of the text with citation commands to refer to this entry. In standard \LaTeX, the bibliography list will be sequentially numbered, except for those entries with an optional \textit{label}, in which case \textit{label} replaces the number. In author-year bibliographies, the \textit{label} must have a special form to transfer author and year texts to the citation commands.

\bibliography{\textit{file_list}} \quad (9.3.2), (14.1) - 219, 309
For producing a bibliography with the aid of the \BibTeX program; \textit{file_list} is the root name of one or more files containing the literature databases to be searched.

\bibliographystyle{\textit{style}} \quad (9.3.2), (14.1) - 219, 310
In conjunction with the \BibTeX program, this command selects the style in which the bibliography entries are to be written. Choices for \textit{style} are plain, unsrt, alpha, and abbrv, or plainnat, unsrtnat, abbrvnat with natbib. Many other contributed styles also exist.
H.1. Brief description of the \LaTeX\ commands

\texttt{\bibname} \hspace{1em} (D.4.1) – 460

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.

\texttt{\bigbrsymbol} \hspace{1em} (5.5.3) – 148

A bracket symbol larger than normal, but smaller than \texttt{\Big}. Example: \texttt{\big(}.

\texttt{\Bigbrsymbol} \hspace{1em} (5.5.3) – 148

A bracket symbol larger than \texttt{\big}, but smaller than \texttt{\bigr}. Example: \texttt{\Big[}.

\texttt{\bigcap} \hspace{1em} (5.3.7) – 128

\texttt{\bigcirc} \hspace{1em} (5.3.3) – 125

\texttt{\bigcup} \hspace{1em} (5.3.7) – 128

A bracket symbol larger than \texttt{\Big}, but smaller than \texttt{\Bigr}. Example: \texttt{\bigr]}.

\texttt{\Bigrbrsymbol} \hspace{1em} (5.5.3) – 148

The largest bracket symbol. Example: \texttt{\Bigrlangle}.

\texttt{\bigodot} \hspace{1em} (5.3.7) – 128

\texttt{\bigoplus} \hspace{1em} (5.3.7) – 128

\texttt{\bigvee} \hspace{1em} (5.3.7) – 128

\texttt{\bigwedge} \hspace{1em} (5.3.7) – 128

With the \texttt{amsmath} package, prints a binomial expression:

\[ \binom{n}{k} \]
\bmod[m] \quad \text{(5.3.8), (12.2.5) - 129, 268}
Command to produce the function name ‘mod’ in the form
\[ a \bmod b = a \text{ mod } b \]

\boldmath \quad \text{(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 \( \mbox{\boldmath$...$} \) to return
temporarily to text mode.

\boldsymbol{symbol} \quad \text{(12.2.1) - 258}
When one of the packages amsmath or amssymb has been loaded, this
command prints symbol in bold face. Unlike \textbf, it also affects
math symbols and lower case Greek letters.

\bot[m] produces \( \perp \) \quad \text{(5.3.6) - 127}
\botfigrule \quad \text{(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}

\bottomfraction \quad \text{(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}{\text{decimal,frac}}
\end{verbatim}

\bottomnumber \quad \text{(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}}.

\bowtie[m] produces \( \bowtie \) \quad \text{(5.3.4) - 126}
\Box[m] produces \( \Box \) \quad \text{(5.3.3) - 125}
\boxed{formula} \quad \text{(12.2.5) - 270}
With the amsmath package, sets the mathematical \textit{formula} in a box.

\breve{x} \quad \text{(5.3.9) - 129}
Breve accent over the math variable \( x \): \( \breve{a} = \breve{a} \).

\Breve{x} \quad \text{(12.2.2) - 263}
With the amsmath package, can be used like \textit{\breve}, but with multiple
\textit{AMSTeX} math accents the positioning will be correct.

\bullet[m] produces \( \bullet \) \quad \text{(5.3.3) - 125}
\c{x} \quad \text{(2.5.7) - 24}
Produces a cedilla under \( x \): \( \c{C} = \c{C} \).
\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.

\captionlanguage ........................................ (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 \\ . See also \texttt{\begin{center}}.

\centerline{text} .............................................. (4.2.1) - 67

An additional \TeX{} command that sets \texttt{text} centered on a horizontal
line by itself.

\cfrac[\texttt{pos}]{}{\over}{\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 \l{} or \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*{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.
Appendix H. Command Summary

\chaptername ... (D.4.1) - 460
Command containing the chapter heading. In English, this is ‘Chapter’ but may be altered for adaptation to other languages.

\check\{x\} [m] ... (5.3.9) - 129
Háček accent over the math variable \( x \): \( \check{a} = \dot{a} \).

\Check\{x\} [m][a] ... (12.2.2) - 263
With the amsmath package, can be used like \check, but with multiple \( \text{AMS-LATEX} \) math accents the positioning will be correct.

\CheckCommand\{\com\name\} [narg] [opt] \{def\} ... (D.2.5) - 445
Tests that the current definition of \com\name 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 [m] produces \( \chi \) ... (5.3.1) - 125
\circ [m] produces \( \circ \) ... (5.3.3) - 125
\circle\{diameter\} ... (13.1.4) - 295
Picture element command to produce a circle with diameter \texttt{diameter} in the \texttt{picture} environment. To be used as an argument in a \texttt{put} or \texttt{multiput} command.
\circle*\{diameter\} ... (13.1.4) - 295
Like \circle but produces a solid circle, filled in black.

\cite\{key\} ... (9.3.3), (14.1) - 219, 310
Literature citation using the identifier \texttt{key} to produce a reference label in the text. The optional note text is included with the label.

\citet\pre-note\post-note\{key\} ... (9.3.4) - 221
With the natbib package, inserts a parenthetical literature citation as ‘[Jones et al., 1999]’, using the identifier \texttt{key}. The optional note texts are included within the parentheses, before and after; if only one is present, it is a \texttt{post-note}. In numerical citation mode, the citation number is printed as ‘[21]’, just as with the standard \cite command. Multiple keys may be given.

\citet\{key\} ... (9.3.4) - 221
With the natbib package, inserts an in-text literature citation as ‘Jones et al. [1999]’, using the identifier \texttt{key}. In numerical citation mode, it prints the author’s name before the citation number, as ‘Jones et al. [21]’.

\ClassError\{class\name\} \{error\text\} \{help\} [p] ... (D.2.7) - 446
Writes an error message \texttt{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 H\texttt{return} is typed, the \texttt{help} text is printed. Both \texttt{error\text} and \texttt{help} may contain \texttt{MessageBreak} for a new line, \texttt{\space} for a forced space, and \texttt{\protect} before commands that are to have their names printed literally and not interpreted.
\texttt{\textbackslash ClassInfo\{class\_name\}\{info\_text\} [p]} . . . . . . . . . (D.2.7) – 447

Is like \texttt{\textbackslash ClassWarningNoLine} except that the text \texttt{info\_text} is only written to the transcript file, and not to the monitor.

\texttt{\textbackslash ClassWarning\{class\_name\}\{warning\_text\} [p]} . . . . . (D.2.7) – 446

Writes \texttt{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 \texttt{warning\_text} is formatted in the same way as that for \texttt{\textbackslash ClassError}.

\texttt{\textbackslash ClassWarningNoLine\{class\_name\}\{warning\_text\} [p]} . . (D.2.7) – 446

Is like \texttt{\textbackslash ClassWarning} except that the current line number of the input file is not printed.

\texttt{\textbackslash 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.

\texttt{\textbackslash clearpage} . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (2.7.4) – 33

Ends the current page and outputs all unprocessed floats on to one or more float pages.

\texttt{\textbackslash cline\{n-m\}} . . . . . . . . . . . . . . . . . . . . . . . . (4.8.1) – 97

In \texttt{\textbackslash tabular} environment, produces a horizontal rule from the beginning of column \texttt{n} to the end of column \texttt{m}. Example: \texttt{\textbackslash cline\{2-5\}}.

\texttt{\textbackslash closing\{regards\}} . . . . . . . . . . . . . . . . . . . . . . . . (16.1) – 353

End of the text in the \texttt{\textbackslash letter} environment; \texttt{regards} stands for the desired terminating text.

\texttt{\textbackslash clubsuit [m]} produces ♠ . . . . . . . . . . . . . . (5.3.6) – 127

\texttt{\textbackslash color 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{\textbackslash color} command. The \texttt{col\_spec} is either the name of a color defined (or predefined) by \texttt{\textbackslash definecolor} or of the form \texttt{[model]{specs}}, where the arguments have the same meaning as they do for \texttt{\textbackslash definecolor}. Examples:

\texttt{\color[rgb]{0.5,0.5,0} \color{magenta}}

\texttt{\textbackslash colorbox col\_spec\{text\}} . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (6.2) – 167

A command made available with the \texttt{color} package. The \texttt{text} is set in an LR box with the specified color as the background color. The \texttt{col\_spec} is the same as for \texttt{\textbackslash color}. 

H.1. Brief description of the \LaTeX{} commands 529
\columnsep \hfill (3.1.1) - 40

Declaration for the amount of intercolumn spacing in two-column page formatting. May be changed with the \setlength command:
\setlength{\columnsep}{1pt}

\columnseprule \hfill (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:
\setlength{\columnseprule}{1pt}

\cong \hfill (5.3.4) - 126

\[m\] produces ≊ \hfill (5.3.4) - 126

\texttt{\contentsline{sec}{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.

\contentsname \hfill (D.4.1) - 459

Command containing the heading for the table of contents. In English, this is 'Contents' but may be altered for adaptation to other languages.

\coprod \hfill (5.3.7) - 128

\[m\] produces \coprod \hfill (5.3.7) - 128

\copyright \hfill (2.5.5) - 23

\cos \hfill (5.3.8) - 128

Command to produce the function name 'cos' in formulas.

\cosh \hfill (5.3.8) - 128

Command to produce the function name 'cosh' in formulas.

\cot \hfill (5.3.8) - 128

Command to produce the function name 'cot' in formulas.

\coth \hfill (5.3.8) - 128

Command to produce the function name 'coth' in formulas.

\csc \hfill (5.3.8) - 128

Command to produce the function name 'csc' in formulas.

\cup \hfill (5.3.3) - 125

\[m\] produces ∪ \hfill (5.3.3) - 125

\CurrentOption \hfill (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

\d{x} ....................................................... (2.5.7) – 24

Produces a ‘dot under’ accent: \( \d{o} = o \).

\dag produces \( \dagger \) ................................. (2.5.5) – 23
\dagger \[m\] produces \( \dagger \) ........................ (5.3.3) – 125

\dashbox{\textit{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 \textit{dash} in the
picture environment. Without the optional \textit{pos}, the contents \textit{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 \texttt{put} or \texttt{multiput}
command.

\dashv \[m\] produces \( \dashv \) ............................. (5.3.6) – 127
\date{\textit{date\_text}} ........................................ (3.3.1), (16.1) – 52, 353

1. The command \texttt{\maketitle} normally prints the current date on
the title page. The declaration \texttt{\date} will replace the date with
whatever text is given in \textit{date\_text}.

2. Prints the text \textit{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 \textit{language}. It oc-
curs in \texttt{german} package as well as in the \texttt{babel} system. It may also be
used for ‘dialects’, such a \texttt{\dateUSenglish} and \texttt{\dateenglish}. This
command is normally part of the definition of the \texttt{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*{-.4pt}
\rule{\textwidth}{.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}{\textit{decimal\_frac}}

\dbinom{\textit{over}}{\textit{under}} \[m][a] .................... (12.2.3) – 264

With the \texttt{amsmath} package, produces a binomial as \texttt{\binom} does, but
in \texttt{\displaystyle} size.
\dblfloatsep \hspace{172pt} (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 \setlength command:
\setlength{\dblfloatsep}{12pt plus 2pt minus 4pt}

\dbltextfloatsep \hspace{172pt} (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 \setlength command.

\dbltopfraction \hspace{172pt} (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
\renewcommand{\dbltopfraction}{\textit{decimal}_frac}.

\dbltopnumber \hspace{172pt} (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
\setcounter{\dbltopnumber}{\textit{num}}.

\ddag \hspace{23pt} (2.5.5) - 23
\ddagger \[m\] \hspace{125pt} (5.3.3) - 125
\dddot{x} \[m]\[a] \hspace{263pt} (12.2.2) - 263

With the amsmath package, a triple dot accent in math formulas:
\ddddot{a} = \dddot{a}.

\ddddot{x} \[m]\[a] \hspace{263pt} (12.2.2) - 263

With the amsmath package, a four-dot accent in math formulas:
\ddddot{a} = \dddot{a}.

\ddot{x} \[m] \hspace{129pt} (5.3.9) - 129

A double dot accent in mathematical formulas: \ddot{a} = \dddot{a}.

\Ddot{x} \[m]\[a] \hspace{263pt} (12.2.2) - 263

With the amsmath package, can be used like \ddot, but with multiple \textit{AMS-Li\TeX} math accents the positioning will be correct.

\ddots \[m] \hspace{123pt} (5.2.6) - 123
\DeclareErrorFont{\textit{code}}{{\textit{fam}}}{{\textit{ser}}}{{\textit{shp}}}{{\textit{sz}}} \[p] \hspace{376pt} (A.3.6) - 376

If no valid font can be found, even after substituting the attributes as given by \DeclareFontSubstitution, the font declared with this command is selected as a last resort.
\DeclareFixedFont{\cmd}{\fam}{\ser}{\shp}{\sz} [p]

Defines \cmd to be a font declaration that selects a font with the fixed attributes given in the definition.

\DeclareFontEncoding{\code}{\textset}{\mathset} [p] \hspace{0.5em} (A.3.6) – 376

Declares \code to be the name of a new encoding scheme; \textset is a set of commands that is to be executed when switching to text mode, \mathset a set for math mode.

\DeclareFontEncodingDefaults{\textset}{\mathset} [p] \hspace{0.5em} (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] \hspace{0.5em} (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] \hspace{0.5em} (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] \hspace{0.5em} (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] \hspace{0.5em} (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 extensions, such as .eps, .ps.

\DeclareGraphicsRule{\ext}{\type}{\bb}{\cmd} [p] \hspace{0.5em} (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}{\sym\fnt}{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 \sym\fnt. 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}{\sym\fnt1}{pos1}

{\sym\fnt2}{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 \sym\fnt1, while the larger is from position \pos2 of font \sym\fnt2.

\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}{\sym\fnt1}{pos1}

{\sym\fnt2}{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 \sym\fnt1, while the larger is from position \pos2 of font \sym\fnt2.

\DeclareMathSizes{\text}{\math\text}{\script}{\sscript} [p] (A.3.4) – 375

Sets the point sizes for the three math styles \textstyle, \scriptstyle, and \scriptscriptstyle when the text is being printed in point size \text. The unit pt is not included.

\DeclareMathSymbol{\symbol}{\type}{\sym\fnt}{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 \sym\fnt. 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.
\DeclareMathVersion{\text{ver}} \[p\] \hspace{1cm} \textit{(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}}{\text{math specs}} \[p\] \hspace{1cm} \textit{(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}} \[p\] \hspace{1cm} \textit{(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}} \[p\] \hspace{1cm} \textit{(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:

\begin{verbatim}
\DeclareOption*{\InputIfFileExists{\CurrentOption.sty}{}{\OptionNotUsed}}
\end{verbatim}

\DeclareRobustCommand{\text{cmd}}[\text{narg}][\text{opt}]{\text{def}} \hspace{1cm} \textit{(D.2.5) - 445}

Defines or redefines the command \text{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}} \hspace{1cm} \textit{(D.2.6) - 445}

The same as \texttt{\DeclareRobustCommand} except that the arguments to \text{cmd} must be 'short', not containing any new paragraphs.

\DeclareSymbolFont{\text{sym}_fnt}{\text{code}}{\text{fam}}{\text{ser}}{\text{shp}} \[p\] \hspace{1cm} \textit{(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 com-
mand \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 \textit{T1} encoding when the accented letter
exists as a separate symbol. For example:
\\DeclareTextComposite{\'}{\textit{T1}}{\textit{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}{\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

\DeclareTextSymbolDefault{\cmd}{\code}{\p} . . . . (A.3.7) – 380
Declarations 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}{0.25,0,0.75,0.0}
\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 Δ . . . . . . . . . . . . . . . . . . . . (5.3.1) – 125
\delta[m] produces δ . . . . . . . . . . . . . . . . . . . . (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.
 Appendix H. Command Summary

\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 đ.

\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,
one, 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\} \hspace{1cm} (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\} \hspace{1cm} (5.3.9) – 129

A dot accent in mathematical formulas: $\dot{a}$.

\Dot\{x\} \hspace{1cm} (12.2.2) – 263

With the `amsmath` package, can be used like `\dot`, but with multiple `\text{AMS-L}\text{T\text{E}X}` math accents the positioning will be correct.

\doteq \hspace{1cm} (5.3.4) – 126

\dotfill \hspace{1cm} (2.7.1) – 30

Fills up the space in a line with a dotted leader: \ldots = \dotfill.

\dots \hspace{1cm} (5.2.6) – 123

\dottimes \hspace{1cm} (12.2.2) – 264

With the `amsmath` package, places continuation dots in math mode automatically at a height determined by the following symbol.

\doublerulesep \hspace{1cm} (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 \hspace{1cm} (5.3.5) – 127

\downarrow \hspace{1cm} (5.3.5) – 127

\ell \hspace{1cm} (5.3.6) – 127

\em \hspace{1cm} (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.
\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 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{\textbackslash enclname}.

\texttt{\textbackslash enclname} \hspace{1cm} (D.4.1) - 460

Command in the \texttt{letter} document class containing the word to be printed by the \texttt{\textbackslash 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{\textbackslash 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{\textbackslash 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{\textbackslash enlargethispage*} \hspace{1cm} (2.7.4) - 34

Is the same as \texttt{\textbackslash 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\}} ........................................ (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 $\epsilon$ ........................................ (5.3.1) – 125

\texttt{\textbackslash eqref\{\textit{marker}\}} \{a\} ............................... (12.2.7) – 278

With the \texttt{amsmath} package, is a variation on the \texttt{\ref} command,
and prints the equation number defined with \texttt{\label\{\textit{marker}\}} in
parentheses, as (5.6).

\texttt{\textbackslash equiv [m]} produces $\equiv$ ........................................ (5.3.4) – 126

\texttt{\textbackslash eta [m]} produces $\eta$ ........................................ (5.3.1) – 125

\texttt{\textbackslash euro} ................................................................. (2.5.8) – 25

With the \texttt{eurosym} package, prints the euro symbol € from the euro-
sym \texttt{METAFONT} fonts. With the \texttt{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} ................................................................. (2.5.8) – 25

With the \texttt{europs} package, prints the euro symbol € from the Adobe
(PostScript) euro fonts, such that it matches the font family and other
attributes. \texttt{\EURofc} prints the invariable symbol €.

\texttt{\textbackslash evensidemargin} .................................................. (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:
\begin{verbatim}
\setlength{\evensidemargin}{2.5cm}
\end{verbatim}

\texttt{\textbackslash ExecuteOptions\{option_list\} \{p\}} ............. (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 $\exists$ ........................................ (5.3.6) – 127

\texttt{\textbackslash exp [m]} ............................................................ (5.3.8) – 128

Command to produce the function name 'exp' in formulas.

\texttt{\textbackslashextracolsep\{extra.width\}} .................................... (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:
\begin{verbatim}
\begin{tabular}{lr@{\extracolsep{2.5mm}}} |c|}
\end{verbatim}
542 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; \text exclude the head and footlines, \text includes them. The arguments set box parameters like \fboxrule but must end with a box command like \shadowbox. Usually one set of \text 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 \text as the background color, but with a frame of color \text around it. The \texts 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 ♭ . . . . . . . . . . . . . . . . . . . . . . . (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}{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

\fnsymbol{counter} \quad (8.1.4) - 183

Prints the current value of the given \texttt{counter} as a `footnote symbol':
* † ‡ § ¶ ∥

\fontencoding{enc} \quad (A.1) - 368

This command selects the font encoding scheme. Possible values of \texttt{enc} are \texttt{OT1} for the standard and \texttt{T1} for the Cork encodings. Other values are also possible.

\fontfamily{fam} \quad (A.1) - 368

This command selects the `family' of fonts. Possible values of \texttt{fam} for standard \LaTeX{} with the Computer Modern fonts are \texttt{cmr}, \texttt{cmsl}, \texttt{cmtt}, and \texttt{cmfi}.

\fontseries{ser} \quad (A.1) - 368

This command selects the `series' of fonts within a `family'. Possible values of \texttt{ser} for standard \LaTeX{} are \texttt{m} (medium) and \texttt{bx} (bold extended).

\fontshape{form} \quad (A.1) - 368

This command selects the `shape' of fonts. Possible values of \texttt{form} are \texttt{n} (normal), \texttt{i}t (italic), \texttt{s}l (slanted), \texttt{s}c (small caps), and \texttt{u} ('unslanted' italic).

\fontsize{sz}{line_sp} \quad (A.1) - 368

This command selects the font size. The argument \texttt{sz} specifies the size of the characters in points (without the dimension \texttt{pt}) and \texttt{line_sp} determines the value of the interline spacing (\texttt{\baselineskip}), with an explicit dimension. Example: \texttt{\fontsize{12}{14pt}}.

\footnote{\texttt{num}}{\texttt{footnote_text}} \quad (4.10.1), (4.10.2) - 112, 114

Produces a footnote containing the text \texttt{footnote_text}. The optional argument \texttt{num} will be used as the footnote number in place of the next number in the automatic sequence.

\footnotemark{\texttt{num}} \quad (4.10.3) - 114

Produces a footnote marker in the current text. The optional argument \texttt{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.

\footnoterule \quad (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}
  {\vspace{-\ht2}\vrule width\ht2\vspace{-\ht2}}
\end{verbatim}
\footnotesep ........................................... (4.10.6) - 117

The vertical spacing between two footnotes. A new value is assigned with the \setlength command:
\setlength{\footnotesep}{6.5pt}

\footnotesize ......................................... (4.1.2) - 62

Switches to the font size \footnotesize, which is smaller than \small but larger than \scriptsize.

\footnotetext[num]{footnote_text} .................. (4.10.3) - 114

Produces a footnote with the text 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 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 ................................................... (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 [m] produces \(\forall\) ......................... (5.3.6) - 127

In the babel system, sets a short text in the selected language.

\frac{numerator}{denominator} [m] ............... (5.2.3) - 122

Math command for generating fractions.

\frame{text} ............................................ (13.1.4) - 298

Produces a frame without any intervening spacing around [EX]. Mainly used as a picture element in a \put or \multiput command within the picture environment.

\framebox[width][pos]{text} ....................... (4.7.1) - 86

Produces a frame of width width around text. By default, the text is centered within the frame, but may be left or right justified by giving the optional argument pos as l or r. It may also have the value s, to stretch the text to the given width.

\framebox(x_dimen,y_dimen)[pos]{text} .......... (13.1.4) - 291

Picture element command to produce a frame 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 \texttt{pos} to a combination of the letters \texttt{l}, \texttt{r}, \texttt{t}, and \texttt{b}, such as \texttt{tr} for top, right; \texttt{pos} may also contain \texttt{s} to stretch the text to the full width. The command is to be used as the argument of a \texttt{put} or \texttt{multiput} command.

\frenchspacing \hfill (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 \texttt{nonfrenchspacing}.

\frontmatter \hfill (3.3.5) - 57

In the \texttt{book} class, introduces the material that comes at the beginning (preface, table of contents) by turning off the chapter numbering of the \texttt{chapter} command and switching to Roman numbers for the pagination.

\frown \texttt{[m]} produces ⌢ \hfill (5.3.4) - 126
\fussy \hfill (2.8.3) - 36

Countercommand of \texttt{sloppy} that allows larger interword spacings than normal. After \texttt{fussy} has been given, the normal spacings apply once more.

\Gamma \texttt{[m]} produces Γ \hfill (5.3.1) - 125
\gamma \texttt{[m]} produces γ \hfill (5.3.1) - 125
\gcd \texttt{[m]} \hfill (5.3.8) - 128

Command to produce the function name 'gcd' in formulas. A lower limit may be given as a subscript.

\ge \texttt{[m]} produces ≥ \hfill (5.3.4) - 126
\genfrac{\{left\}}{\{right\}}{\{thkns\}}{\{mathsz\}}{\{over\}}{\{under\}} \texttt{[m]} \texttt{[a]} \hfill (12.2.3) - 265

With the \texttt{amsmath} package, produces a generalized fraction with delimiters \texttt{left} and \texttt{right}, line thickness \texttt{thkns}, math font size \texttt{mathsz} 0–3, and with \texttt{over} on top of \texttt{under}. If \texttt{mathsz} is empty, the sizing is automatic.

\geq \texttt{[m]} produces ≥ \hfill (5.3.4) - 126
\gets \texttt{[m]} produces ← \hfill (5.3.5) - 127
\gg \texttt{[m]} produces ≫ \hfill (5.3.4) - 126
\glossary{glossary.entry} \hfill (9.4.4) - 230

Write a \texttt{glossaryentry} command to the .glo file if \texttt{makeglossary} has been issued in the preamble; else it does nothing.

\glossaryentry{glossary.entry}{page.number} \hfill (9.4.4) - 230

The form in which the entry is written to the .glo file by the \texttt{glossary} command.
\texttt{graphpaper} \texttt{[num]} \texttt{(x,y \ (lx, ly))} ............. (13.1.5) – 301

A command added with the \texttt{graphpap} package for use in the \texttt{picture}
environment. It plots a labeled grid system with the lower left corner at \texttt{(x,y)}, \texttt{lx} wide and \texttt{ly} high. Grid lines are drawn every \texttt{num} units, with the fifth ones thicker. If \texttt{num} is not specified, it is assumed to be 10. All arguments must be integers, not decimal fractions.

\texttt{\grave{x}} \texttt{[m]} ................................ (5.3.9) – 129

A grave accent over the math variable \texttt{x}: \texttt{\grave{a}} = \texttt{`a}.

\texttt{\Grave{x}} \texttt{[m][a]} ................................ (12.2.2) – 263

With the \texttt{amsmath} package, can be used like \texttt{\grave}, but with multiple \texttt{\LaTeX} math accents the positioning will be correct.

\texttt{\guillemotleft} ..................................... (G.4.5) – 503

When T1 encoding is active, prints the symbol «.

\texttt{\guillemotright} .................................... (G.4.5) – 503

When T1 encoding is active, prints the symbol ».

\texttt{\guilsinglleft} ..................................... (G.4.5) – 503

When T1 encoding is active, prints the symbol ‹.

\texttt{\guilsinglright} .................................... (G.4.5) – 503

When T1 encoding is active, prints the symbol ›.

\texttt{\H{x}} ................................................ (2.5.7) – 24

Hungarian double acute accent: \texttt{\H{o}} = ò.

\texttt{\hat{x}} \texttt{[m]} .................................... (5.3.9) – 129

Circumflex over the math variable \texttt{x}: \texttt{\hat{a}} = \hat{a}.

\texttt{\Hat{x}} \texttt{[m][a]} ................................ (12.2.2) – 263

With the \texttt{amsmath} package, can be used like \texttt{\hat}, but with multiple \texttt{\LaTeX} math accents the positioning will be correct.

\texttt{\hbar} \texttt{[m]} produces \texttt{ℏ} ............... (5.3.6) – 127

\texttt{\headheight} ........................................ (3.2.5) – 48

The height of the head at the top of each page. A new value is assigned with the \texttt{\setlength} command:

\texttt{\setlength{\headheight}{25pt}}

\texttt{\headsep} ........................................... (3.2.5) – 48

Vertical spacing between the lower edge of the page head and the top of the main text. A new value is assigned with the \texttt{\setlength} command:

\texttt{\setlength{\headsep}{0.25in}}
H.1. Brief description of the \LaTeX{} commands

\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 \textit{width} specification of \texttt{\makebox}, \texttt{\framebox}, or \texttt{\savebox}, or in the \textit{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 \texttt{\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: _______________ = \texttt{\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.
\huge \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (4.1.2) - 62

Switches to the font size \huge, which is smaller than \Huge but larger than \LARGE.

\Huge \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (4.1.2) - 62

Switches to the largest font size available \Huge, which is larger than \huge.

\hyperlink{name}{link} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (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 \texttt{hypertarget} command. The \textit{link} may be text, a symbol, or a graphics loaded with \texttt{includegraphics}. The link is framed with a colored box, unless the option colorlinks has been set to \texttt{true}, in which case the \textit{link} is set in a text color determined by the option linkcolor.

\hypertarget{name}{text} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (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 \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (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 \texttt{usepackage} loading command, by with \texttt{hypersetup}, these values may be changed within the document. The list of possible keys and values is given on pages 242-246.

\hyphenation{hyphenation_list} [p] \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (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: \texttt{hyphenation permitted}.

\texttt{i} produces \texttt{i} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (2.5.7) - 24
\texttt{idotsint} \texttt{[m][a]} produces \texttt{\ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (12.2.2) - 260
\texttt{iff} \texttt{[m]} produces \texttt{\Leftrightarrow} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (5.3.5) - 127
\texttt{IfFileExists}{file_name}{true}{false} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (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 \texttt{true} is executed; otherwise, \texttt{false}. This is like \texttt{InputIfFileExists} except the file is not input.

\texttt{iflanguage}{language}{yes.text}{no.text} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (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

A conditional command available when the standard package ifthen has been loaded. If the logical statement test evaluates to \langle true\rangle 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 ℍ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (5.3.6) – 127
\imath\{m\} produces ℎ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (5.3.6) – 127
\in\{m\} produces ∈ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (5.3.4) – 126
\include\{file\} . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (9.1.2) – 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.
\includegraphics[\text{key}=\text{value},...]{\text{file.name}} \hspace{1cm} (6.1.3) \hspace{1cm} 157

With the graphicx package, this command has a different syntax in which the scaling, rotating, clipping are effected through \text{key}=\text{value} pairs, like width=7cm, angle=90, scale=.5.

\includeonly{\text{file.list}}[p] \hspace{1cm} (9.1.2) \hspace{1cm} 209

Only those files whose names are in \text{file.list}, separated by commas, will be read in by the \include commands. The \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.

\text{indent} \hspace{1cm} (3.2.4) \hspace{1cm} 46

The first line of the next paragraph is to be indented.

\text{index}\{\text{index.entry}\} \hspace{1cm} (9.4.2), (9.4.2) \hspace{1cm} 225, 226

Writes a \text{indexentry} command to the .idx file if the \makeindex command has been issued in the preamble; otherwise it does nothing. The MakeIndex program (Section 9.4.3) can process this file if the entries are in the forms

\begin{verbatim}
\text{index}\{\text{main.entry}\}
\text{index}\{\text{main.entry!sub.entry}\}
\text{index}\{\text{main.entry!sub.entry!sub.sub.entry}\}
\end{verbatim}

making up a theindex environment with the entries alphabetically ordered and organized with \text{item}, \text{subitem}, and \text{subsubitem} commands.

\text{indexentry}\{\text{index.entry}\}\{\text{page.number}\} \hspace{1cm} (9.4.2) \hspace{1cm} 226

The form in which the entry is written to the .idx file by the \text{index} command.

\text{indexname} \hspace{1cm} (D.4.1) \hspace{1cm} 459

Command containing the heading for the index. In English, this is ‘Index’ but may be altered for adaptation to other languages.

\text{indexspace} \hspace{1cm} (9.4.1) \hspace{1cm} 225

Command within theindex environment to produce a blank line.

\text{inf}[m] \hspace{1cm} (5.3.8) \hspace{1cm} 128

Command to produce the function name ‘inf’ in formulas. A lower limit may be set as a subscript.

\text{infty}[m] \hspace{1cm} \text{produces} \infty \hspace{1cm} (5.3.6) \hspace{1cm} 127
\text{intertext}\{\text{insert.text}\}[m][a] \hspace{1cm} (12.2.1) \hspace{1cm} 259

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} \hfill (9.1.1) – 207

 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. The file that is read in may also contain further \input commands.

\InputIfFileExists{file name}{true}{false} \hfill (D.2.8) – 447

 Tests if the file file name can be found in the places where \LaTeX\ looks for files; if so, the code true is executed and the file is input; otherwise, false is executed.

\int \[m\] produces ∫ \hfill (5.2.5) – 123
\iint \[m\][a] produces ∬ \hfill (12.2.2) – 260
\iiint \[m\][a] produces ∭ \hfill (12.2.2) – 260
\iiiint \[m\][a] produces ∬∬ \hfill (12.2.2) – 260
\intextsep \hfill (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 \hfill (15.1.2) – 326

 In 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 ι \hfill (5.3.1) – 125
\itdefault \hfill (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[label] \hfill (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 enumerate environment. The optional argument inserts label in place of this standard item label.

\item \hfill (9.4.1) – 225

 Produces a main entry in the index environment.

\itemindent \hfill (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 `\parsep` 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 `\text{j}` ................................ (2.5.7) - 24
`\jmath` produces `j` ................................... (5.3.6) - 127
`\Join` produces `\bowtie` .......................... (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{\text{x}}` ........................................ (G.4.5) - 503
When T1 encoding is active, prints the ogonek accent `\k{\text{A}}` = Ą.

`\kappa` produces `\text{kappa}` ......................... (5.3.1) - 125
`\ker` ................................................ (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{marker}` .................................... (9.2.1) - 213
Sets a marker in the text at this position with the name `marker`. It may be referred to either earlier or later in the document with the command `\ref{marker}` to output the counter that was then current, such as the section, equation, or figure number, or with the command `\pageref{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 `n` is one of `i`, `ii`, `iii`, or `iv`. For example,
```
\renewcommand{\labelenumii}{\arabic{enumii}.}
```
changes the standard labels of the second-level `enumerate` environment to be 1.), 2.), etc.
H.1. Brief description of the \LaTeX commands

\labelitemn \hspace{1.5cm} (4.3.5) – 73

A set of commands to produce the standard labels for the nesting levels of the itemize environments, where \( n \) is one of \( i, ii, iii, \) or \( iv \). For example,

\begin{verbatim}
\renewcommand{\labelitemi}{$\Rightarrow$}
\end{verbatim}

changes the standard labels of the outermost itemize environment to \( \Rightarrow \).

\labelsep \hspace{1.5cm} (4.4.2) – 77

In a list environment, the distance between the label box and the list text. A new value is assigned with the \setlength command:

\begin{verbatim}
\setlength{\labelsep}{5pt}
\end{verbatim}

\labelwidth \hspace{1.5cm} (4.4.2) – 77

In a list environment, the width of the box reserved for the label. A new value is assigned with the \setlength command:

\begin{verbatim}
\setlength{\labelwidth}{2.2cm}
\end{verbatim}

\Lambda [m] produces \( \Lambda \) \hspace{1.5cm} (5.3.1) – 125

\lambda [m] produces \( \lambda \) \hspace{1.5cm} (5.3.1) – 125

\langle [m] produces \( \langle \) \hspace{1.5cm} (5.4.1) – 132

\language{\num} \hspace{1.5cm} (11.1) – 255

In \TeX versions 3.0 and later, the set of hyphenation patterns number \( \num \) is made active. The patterns must be previously loaded into the format file by an initex run in which \language{\num} was given before those patterns were read in.

\large \hspace{1.5cm} (4.1.2) – 62

Switches to the font size \large, which is smaller than \Large but larger than \normalsize.

\Large \hspace{1.5cm} (4.1.2) – 62

Switches to the font size \Large, which is smaller than \LARGE but larger than \large.

\LARGE \hspace{1.5cm} (4.1.2) – 62

Switches to the font size \LARGE, which is smaller than \huge but larger than \Large.

\LaTeX produces \LaTeX \hspace{1.5cm} (2.1) – 18

\LaTeXe produces \LaTeXe \hspace{1.5cm} (2.1) – 19

\lceil [m] produces \( \lceil \) \hspace{1.5cm} (5.4.1) – 132

\ldots \hspace{1.5cm} (5.2.6) – 123

\le [m] produces \( \le \) \hspace{1.5cm} (5.3.4) – 126

\leadsto [m] produces \( \leadsto \) \hspace{1.5cm} (5.3.5) – 127
Appendix H. Command Summary

\leftlbrack \text{[}] \quad (5.4.1) - 131

Adjusts the size of the bracket symbol \text{lbrack} to fit the height of the formula between the \text{\left ... \right} pair. For example, \text{\left[}. If there is to be no matching bracket, the \text{\left} and \text{\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, \text{\right}).

\Leftarrow \text{[m]} \quad (5.3.5) - 127

\leftarrow \text{[m]} \quad (5.3.5) - 127

\leftharpoonup \text{[m]} \quad (5.3.5) - 127

\lefteqn \text{[m]} \quad (5.4.7) - 140

A command inside the \text{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 \quad (4.4.2) - 76

In a \text{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 \text{\setlength} command. For nested \text{list} environments, different values for the indentation can be specified by adding \text{i} ... \text{vi} to the declaration name, such as

\text{\setlength{\leftmarginiii}{0.5cm}}

See also \quad (4.3.4) - 71.

\Leftarrowrightarrow \text{[m]} \quad (5.3.5) - 127

\leftrightharpoons \text{[m]} \quad (5.3.5) - 127

\leftroot{shift} \text{[m][a]} \quad (12.2.5) - 269

With the amsmath package, used in the index to a \text{\sqrt} command to shift it slightly to the left. The \text{shift} is a number specifying how many units to move it. Example:

\text{\sqrt[\leftroot{-1}\uproot{3}\beta]{k}}

\leq \text{[m]} \quad (5.3.4) - 126

\lfloor \text{[m]} \quad (5.4.1) - 132

\lg \text{[m]} \quad (5.3.8) - 128

Command to produce the function name 'lg' in formulas.

\leq \text{[m]} \quad (5.3.3) - 125

\lim \text{[m]} \quad (5.3.8) - 128

Command to produce the function name 'lim' in formulas. A lower limit may be set as a subscript.

\liminf \text{[m]} \quad (5.3.8) - 128

Command to produce the function name 'lim inf' in formulas. A lower limit may be set as a subscript.
\limits[m] .......................... (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.

\limsup[m] .......................... (5.3.8) - 128
Command to produce the function name 'lim sup' in formulas. A lower limit may be set as a subscript.

\line(\Delta x, \Delta y)\{length\} .......................... (13.1.4) - 293
A picture element command within a 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 (\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.

\linebreak[n] .......................... (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 \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 an obligatory line break.

\linethickness\{thickness\} .......................... (13.1.4) - 300
Sets the thickness of the horizontal and vertical lines in the picture environment. The argument \textit{thickness} is a length specification with units, for example, 1.2mm.

\linewidth .......................... (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

\includegraphics[width=0.8\linewidth]{..}

\listfigurename .......................... (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.

\listfiles [p] .......................... (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 \Provides... commands.
\listoffigures \hspace{1cm} \text{(3.4.4) - 59}\hspace{1cm}

Produces a list of figures with the entries from all the \caption commands in \texttt{figure} environments.

\listoftables \hspace{1cm} \text{(3.4.4) - 59}\hspace{1cm}

Produces a list of tables with the entries from all the \caption commands in \texttt{table} environments.

\listparindent \hspace{1cm} \text{(4.4.2) - 77}\hspace{1cm}

Depth of indentation for the first line of a paragraph inside a \texttt{list} environment. A new value may be assigned with the \texttt{setlength} command:

\begin{verbatim}
\setlength{\listparindent}{1em}
\end{verbatim}

\listtablename \hspace{1cm} \text{(D.4.1) - 459}\hspace{1cm}

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 \hspace{1cm} \text{(5.3.4) - 126}\hspace{1cm}

\ln \hspace{1cm} \text{(5.3.8) - 128}\hspace{1cm}

Command to produce the function name 'ln' in formulas.

\LoadClass[\texttt{options}]{\texttt{class}}[\texttt{version}][\texttt{p}] \hspace{1cm} \text{(D.2.2) - 442}\hspace{1cm}

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 \texttt{.cls}. Any options specified in the \texttt{documentclass} command are not passed over as global options.

The optional \texttt{version} is a date, given in the form \texttt{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{\texttt{class}}[\texttt{version}][\texttt{p}] \hspace{1cm} \text{(D.2.2) - 442}\hspace{1cm}

Like \texttt{\LoadClass} except all the currently specified options are automatically passed to \texttt{class}.

\location{\texttt{number}} \hspace{1cm} \text{(16.1) - 352}\hspace{1cm}

In the \texttt{letter} document class, enters the sender's room number. In the standard \LaTeX \texttt{letter} class, \texttt{number} is only output if \texttt{\address} has not been called. It is intended to be used in company letterheads.

\log \hspace{1cm} \text{(5.3.8) - 128}\hspace{1cm}

Command to produce the function name 'log' in formulas.
H.1. Brief description of the \LaTeX\ commands 557

\Longleftarrow \[m\] produces $\Leftarrow$. . . . . . . . . . . (5.3.5) – 127
\longrightarrow \[m\] produces $\rightarrow$. . . . . . . . . . . (5.3.5) – 127
\Longleftrightarrow \[m\] produces $\iff$. . . . . . . . . . . (5.3.5) – 127
\longleftrightarrow \[m\] produces $\leftrightarrow$. . . . . . . . . (5.3.5) – 127
\longmapsto \[m\] produces $\mapsto$. . . . . . . . . . . . . (5.3.5) – 127
\Longrightarrow \[m\] produces $\Longrightarrow$. . . . . . . . . . . (5.3.5) – 127

\lq produces ', identical to the ' symbol.
\lvert \[m]\[a\] produces | (left delimiter) . . . . . . . . (12.2.5) – 270
\lVert \[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(\emph{x_dimen},\emph{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 \c a shorthand form for \verb:\verb c: everything that appears between two occurrences of \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 \author and \title commands, and optionally those in the \date and \thanks commands.

\MakeUppercase{\textcmd} . . . . . . . . . . . . . . (D.3.2) - 455
Converts \textcmd (text and commands) to upper case.

\mapsto [m] produces \mapsto \rightarrow . . . . . . . . . . . . . . (5.3.5) - 127
\marginpar[\letxt]{\rtext} . . . . . . . . . . . . . . (4.10.5) - 117
Produces a marginal note at the right of the text containing \rtext. With two-sided formatting, the marginal note goes into the left margin on the even pages, in which case the optional \letxt will be written instead. For two-column text, the marginal notes always go into the ‘outer’ margin, and again \letxt 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 \settlength command.

\marginparskip . . . . . . . . . . . . . . (4.10.6) - 118
The spacing between the edge of the text and a marginal note. A new value may be assigned with the \settlength command.

\marginparwidth . . . . . . . . . . . . . . (4.10.6) - 118
The width of the box reserved for marginal notes. A new value may be assigned with the \settlength command.

\markboth{\lethead}{\rhead} . . . . . . . . . . . . . . (3.2.1) - 43
Sets the text entries for the left and right page headlines in two-sided formatting when the page style myheadings has been selected, or when the automatic entries of page style headings are to be changed.

\markright{\head} . . . . . . . . . . . . . . (3.2.1) - 43
Sets the text entry for the page headline when the page style myheadings has been selected, or when the automatic entry of page style headings is to be manually changed. In two-sided formatting, only the right headline is set with this command.

\mathbf{\text}[m] . . . . . . . . . . . . . . (5.4.2), (A.3.3) - 133, 373
This command sets text in a bold font (\bfseries) within math mode. Spaces are ignored as usual.
\textbf{H.1. Brief description of the \LaTeX{} commands}  \hfill 559

\texttt{\textbackslash mathcal\{text\} [m]}  \ldots  \ldots  (5.3.2), (5.4.2), (A.3.3) - 125, 133, 373
This command sets \textit{text} in calligraphic letters within math mode:
\$	exttt{\textbackslash mathcal\{ABC\}} = \mathcal{ABC}$.

\texttt{\textbackslash mathindent}  \ldots  \ldots  \ldots  \ldots  (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} command:
\texttt{\setlength\{\textbackslash mathindent\}\{25pt\}}

\texttt{\textbackslash mathit\{text\} [m]}  \ldots  \ldots  \ldots  \ldots  (5.4.2), (A.3.3) - 133, 373
This command sets \textit{text} in a text italic font (\texttt{\itshape}) within math
mode. It differs from \texttt{\textbackslash mathnormal} in that the spacing between the
letters is as in regular text. Compare \textit{mathit} and \textit{mathnormal}.

\texttt{\textbackslash mathnormal\{text\} [m]}  \ldots  \ldots  (5.3.1), (5.4.2), (A.3.3) - 125, 133, 373
This command sets \textit{text} in the normal (italic) math font within math
mode. In this font, capital Greek letters are also set in italics:
\$	exttt{\textbackslash Gamma\textbackslash mathnormal\{\textbackslash Gamma\}} = \Gamma$.

\texttt{\textbackslash mathring\{x\} [m]}  \ldots  \ldots  \ldots  (5.3.9) - 129
A ring accent in mathematical formulas: \texttt{\textbackslash mathring\{a\} = \acute{a}}.

\texttt{\textbackslash mathrm\{text\} [m]}  \ldots  \ldots  \ldots  \ldots  (5.4.2), (A.3.3) - 133, 373
This command sets \textit{text} in a Roman font (\texttt{\rmfamily}) within math
mode. Spaces are ignored as usual.

\texttt{\textbackslash mathsf\{text\} [m]}  \ldots  \ldots  \ldots  \ldots  (5.4.2), (A.3.3) - 133, 373
This command sets \textit{text} in a sans serif font (\texttt{sffamily}) within math
mode. Spaces are ignored as usual.

\texttt{\textbackslash mathtt\{text\} [m]}  \ldots  \ldots  \ldots  \ldots  (5.4.2), (A.3.3) - 133, 373
This command sets \textit{text} in a typewriter font (\texttt{ttfamily}) within math
mode. Spaces are ignored as usual.

\texttt{\textbackslash mathversion\{ver\}}  \ldots  \ldots  \ldots  \ldots  (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]}  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  (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 \textit{text}  \ldots  \ldots  (4.7.1) - 86
\texttt{\textbackslash mddefault}  \ldots  \ldots  \ldots  \ldots  (A.3.1) - 372
This command defines the series attribute that is selected with the
\texttt{\textbackslash mdseries} command. It may be redefined with \texttt{\renewcommand}:
\texttt{\renewcommand\{\textbackslash mddefault\}\{m\}}
\texttt{\textbackslash mdseries} \hspace{1cm} (4.1.3), (A.2) - 64, 371

This declaration switches to a font in the current family and shape, \textbf{but with the medium series attribute.}

\texttt{\textbackslash medskip} \hspace{1cm} (2.7.3) - 32

Inserts large vertical spacing of the amount \texttt{\medskipamount}. See also \texttt{\bigskip} and \texttt{\smallskip}.

\texttt{\medskipamount}

Standard value for the amount of vertical spacing that is inserted with the command \texttt{\medskip}. May be changed with the \texttt{\setlength} command:

\texttt{\setlength{\medskipamount}{3ex plus1ex minus1ex}}

\texttt{\medspace [m][a]} \hspace{1cm} (12.2.5) - 269

With the \texttt{amsmath} package, this is an alias for \texttt{\:}, a medium space in a math formula.

\texttt{\MessageBreak} \hspace{1cm} (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.

\texttt{\mho [m] produces \Upsilon} \hspace{1cm} (5.3.6) - 127

\texttt{\mid [m] produces |} \hspace{1cm} (5.3.4) - 126

\texttt{\min [m]} \hspace{1cm} (5.3.8) - 128

Command to produce the function name 'min' in formulas. A lower limit may be set as a subscript.

\texttt{\mod{arg} [m][a]} \hspace{1cm} (12.2.5) - 268

With the \texttt{amsopn} or \texttt{amsmath} packages, command to produce the function name 'mod' in formulas in the form:

\texttt{y \mod{a+b} = y \mod a + b}

\texttt{\models [m] produces \vdash} \hspace{1cm} (5.3.4) - 126

\texttt{\mp [m] produces \mp} \hspace{1cm} (5.3.3) - 125

\texttt{\mspace{mu} [m][a]} \hspace{1cm} (12.2.5) - 269

With the \texttt{amsmath} package, inserts spacing in math formulas; \texttt{mu} is a math space in units of mu (=1/18 em): \texttt{\mspace{-4mu}}.

\texttt{\mu [m] produces \mu} \hspace{1cm} (5.3.1) - 125

\texttt{\multicolumn{n}{col}{text}} \hspace{1cm} (4.8.1) - 97

Merges the next \texttt{n} columns in the \texttt{array} and \texttt{tabular} environments, formatting the text entry \texttt{text} according to the single column definition \texttt{col}, which may be \texttt{l}, \texttt{c}, \texttt{r} as well as \texttt{|}. 
\multput(x,y)(\Delta x,\Delta y){n}\{pic\_elem\} \quad (13.1.3) - 289

Multiple positioning command in the picture environment. The object \texttt{pic\_elem} is placed \texttt{n} times, at \((x, y), (x + \Delta x, y + \Delta y), \ldots (x + (n - 1)\Delta x, y + (n - 1)\Delta y)\).

\multlinegap [m][a] \quad (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.

\nabla [m] produces $\nabla$. \quad (5.3.6) - 127

\natural [m] produces $\natural$. \quad (5.3.6) - 127

\name{sender} \quad (16.2) - 356

In the \texttt{letter} document class, enters the sender’s name.

\natural [m] produces $\natural$. \quad (5.3.6) - 127

\nearrow [m] produces $\nearrow$. \quad (5.3.5) - 127

\NeedsTeXFormat{version} \quad (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{version} is \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:

\NeedsTeXFormat{LaTeX2e}[1994/06/01]

\neg [m] produces $\neg$. \quad (5.3.6) - 127

\negmedspace [m][a] \quad (12.2.5) - 269

With the \texttt{amsmath} package, this inserts a negative medium space in a math formula.

\negthinspace [m][a] \quad (12.2.5) - 269

With the \texttt{amsmath} package, this is an alias for $\!$, a negative thin space in a math formula.

\negthinspace [m][a] \quad (12.2.5) - 269

With the \texttt{amsmath} package, this is an alias for $\!$, a negative thin space in a math formula.

\newboolean{switch} \quad (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 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{com\_name}{narg}[opt]{def} \quad (8.3) - 185

Defines a user command with the name \texttt{com\_name} to be \texttt{def}. The first optional argument \texttt{narg} \leq 9 specifies how many variable arguments the command is to have, which appear in the \texttt{def} as the replacement characters \#1 to \#narg. If the second optional argument is present, the first argument of the new command is optional, and takes on the value \texttt{opt} if it is not explicitly given.

\newcommand*{com\_name}{narg}[opt]{def} \quad (D.2.6) - 445

The same as \texttt{\newcommand} except that the arguments to \texttt{com\_name} must be ‘short’, not containing any new paragraphs.

\newcounter{counter\_name}{in\_counter} \quad (8.1.2) - 182

Establishes a new counter with the name \texttt{counter\_name}. The optional argument \texttt{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 \texttt{in\_counter}.

\newenvironment{env\_name}{narg}[opt]{beg\_def}{end\_def} \quad (8.4) - 195

Defines a user environment with the name \texttt{env\_name} which has the \texttt{\begin} definition \texttt{beg\_def} and the \texttt{\end} definition \texttt{end\_def}. The optional argument \texttt{narg} \leq 9 specifies how many variable arguments the environment is to have, which appear in the \texttt{beg\_def} as the replacement characters \#1 to \#narg. If the second optional argument is present, the first argument of the \texttt{\begin} command is optional, and takes on the value \texttt{opt} if it is not explicitly given.

\newenvironment*{env\_name}{narg}[opt]{beg\_def}{end\_def} \quad (D.2.6) - 445

The same as \texttt{\newenvironment} except that the arguments to \texttt{\begin{env\_name}} must be ‘short’, not containing any new paragraphs.

\newfont{\font\cmd}{\font\name scaled size} \quad (4.1.5) - 66

Establishes the relation between the font file name \texttt{file\_name} magnified by the scaling factor \texttt{size} and a font selection command \texttt{\font\cmd}. After \texttt{\font\cmd} has been called, \texttt{\baselineskip}, the interline spacing, still has its previous value.

\newlength{\length\cmd} \quad (8.2) - 185

Creates a new length command with the name \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{\length\cmd}{length}}

The quantity \texttt{length} must have units (cm, pt, etc.) and may be a rubber length.
H.1. Brief description of the \LaTeX\ commands 563

\texttt{\textbackslash newline} ................................................. (2.7.2) – 31
Terminates and starts a line of text 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 \texttt{\textbackslash boxname} in which LR boxes may be saved with the \texttt{\textbackslash savebox} command.

\texttt{\textbackslash newtheorem{\texttt{type}}[\texttt{num_like}]{\texttt{title}}}  \\
\texttt{\textbackslash newtheorem{\texttt{type}}{\texttt{title}}[\texttt{in ctr}]}  (4.5) – 80
Defines a new theorem-like environment named \texttt{type} which when called prints a theorem declaration with the name \texttt{title} in \textbf{bold face}, followed by an automatic sequential number, and the actual text of the environment in \textit{italic}. The optional argument \texttt{num_like} is the name of another theorem structure which is to share the same numbering counter. The other optional argument \texttt{in ctr} is the name of a sectioning counter, such as \texttt{chapter}, which is to reset the theorem counter every time it is incremented. That is, the theorem counter is a sub-counter of \texttt{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 D.

\texttt{\textbackslash ng} ....................................................... (G.4.5) – 503
When T1 encoding is active, prints the character η.

\texttt{\textbackslash ni [m]} produces \( \ni \) ................................... (5.3.4) – 126
\texttt{\textbackslash nocite{key}} ................................................. (14.1) – 310
The entry in the literature database with the keyword \texttt{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 \texttt{.aux,.glo,.idx,.lof,.lot}, and \texttt{.toc}.

\texttt{\textbackslash noindent} ............................................... (3.2.4) – 46
The first line of the \textit{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

\nolinebreak[\ n] \hspace{2cm} (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 \( n \) 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.

\nonfrenchspacing \hspace{2cm} (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.

\nonumber [\ m] \hspace{2cm} (5.4.7) - 139

The formula line in an \texttt{eqnarray} environment in which this command appears will \textit{not} contain an equation number.

\nopagebreak[\ n] \hspace{2cm} (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 \( n \) 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.

\normalcolor \hspace{2cm} (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.

\normalfont \hspace{2cm} (4.1.3), (A.2) - 65, 371

This declaration switches to the font with the default family, shape, and series attributes.

\normalmarginpar \hspace{2cm} (4.10.5) - 117

Countermands \texttt{reversemarginpar}, switching back to the standard placement of marginal notes in the 'outer' margin.

\normalsize \hspace{2cm} (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}.

\not [\ m] \hspace{2cm} (5.3.4) - 127

Changes the following comparison symbol into its negative counterpart:
\[
\not\cong = \neq
\]
\textbf{H.1. Brief description of the \LaTeX{} commands}

\texttt{\textbackslash notag\{mark\}} [m][a] \hspace{1cm} (12.2.6) - 271

Within one of the $\texttt{\textbackslash AMS-L\LaTeX{}}$ alignment environments, suppresses the automatic equation number.

\texttt{\textbackslash notin [m]} produces $\notin \hspace{1cm}$ (5.3.4) - 127

\texttt{\textbackslash nu [m]} produces $\nu \hspace{1cm}$ (5.3.1) - 125

\texttt{\textbackslash numberwithin\{ctr\}\{in ctr\} [a]} \hspace{1cm} (12.2.7) - 277

With the \texttt{amsmath} package, redefines the counter $\texttt{ctr}$ to be a sub-counter 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:

\texttt{\textbackslash numberwithin\{equation\}\{section\}}

\texttt{\textbackslash nwarrow [m]} produces $\nwarrow \hspace{1cm}$ (5.3.5) - 127

\texttt{\textbackslash \textbackslash O produces $\varnothing \hspace{1cm}$ (2.5.6) - 24

\texttt{\textbackslash o produces $\emptyset \hspace{1cm}$ (2.5.6) - 24

\texttt{\textbackslash oddsidemargin} \hspace{1cm} (3.2.5) - 48

Sets the left margin for the odd-numbered pages in document class 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{\textbackslash setlength} command:

\texttt{\textbackslash setlength\{\textbackslash evensidemargin\}\{1.5cm\}}

\texttt{\textbackslash oint [m]} produces $\oint \hspace{1cm}$ (5.3.7) - 128

\texttt{\textbackslash \textbackslash \Omega produces $\Omega \hspace{1cm}$ (5.3.1) - 125

\texttt{\textbackslash \textbackslash \omega produces $\omega \hspace{1cm}$ (5.3.1) - 125

\texttt{\textbackslash \textbackslash \ominus produces $\ominus \hspace{1cm}$ (5.3.3) - 125

\texttt{\textbackslash onecolumn} \hspace{1cm} (3.2.7) - 51

Starts a new page and switches from two-column to one-column page formatting.

\texttt{\textbackslash onlynotes\{note_nums\}} \hspace{1cm} (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 \textit{note_nums}. The command behaves the same as \texttt{\textbackslash onlyslides} for \texttt{slides}.

\texttt{\textbackslash onlyslides\{slide_nums\}} \hspace{1cm} (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 \textit{slide_nums}. The numbers are separated by commas, and may include a range with a hyphen: \texttt{\textbackslash onlyslides\{4,10-13,23\}.
\opening{\textit{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 \texttt{[m]} produces $\oplus$ ........................................ (5.3.3) – 125
\OptionNotUsed \texttt{[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 \texttt{[m]} produces $\oslash$ ........................................ (5.3.3) – 125
\otimes \texttt{[m]} produces $\otimes$ ........................................ (5.3.3) – 125
\oval(x_{\text{dimen}},y_{\text{dimen}})[\text{part}] ........................ (13.1.4) – 296

Picture element command to produce an oval with width $x_{\text{dimen}}$
and height $y_{\text{dimen}}$ 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 $t_1$
or $l_1$ for the top left part. To be used as an argument in a \texttt{put}
or \texttt{multiput} command.

\ovalbox{\textit{text}} .................................................. (4.7.9) – 94

With the \texttt{fancybox} package, is a variant of \texttt{fbox}, drawing a framed
box with round corners around \textit{text}; the thickness of the lines is
given by \texttt{thinlines}.

\Ovalbox{\textit{text}} .................................................. (4.7.9) – 94

With the \texttt{fancybox} package, is the same as \texttt{ovalbox} but the thick-
ness of the lines is given by \texttt{thicklines}.

\overbrace{\textit{sub.form}} [\texttt{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 horizon-
tal brace.

\overbrace{a+b} = \overline{a+b}

\overbrace{x+y+z}^{\xi\eta\zeta} = \overline{x+y+z}

\overleftarrow{\textit{expr}} [\texttt{m}[\texttt{a}] ......................... (12.2.2) – 262

With the \texttt{amsmath} package, places a long leftwards pointing arrow
over the mathematical expression \textit{expr}.

\overleftright\texttt{arrow}[\textit{expr}] [\texttt{m}[\texttt{a}] .................. (12.2.2) – 262

With the \texttt{amsmath} package, places a long double arrow over the
mathematical expression \textit{expr}. 

H.1. Brief description of the LATEX commands
\overline{sub form} [m]

567

. . . . . . . . . . . . . . . (5.4.4) – 136

Produces a horizontal bar over the math formula sub form:
\overline{a-b} = a − b

\overrightarrow{expr} [m][a] . . . . . . . . . . . .

(12.2.2) – 262

With the amsmath package, places a long rightwards pointing arrow
over the mathematical expression expr.

\overset{char}{\symbol} [m][a] . . . . . . . . . . .

(12.2.2) – 262

With the amsmath package, places char over the math symbol \symbol
in superscript size.

\P produces ¶ . . . . . . . . . . . . . . . . . . . . . . . (2.5.5) – 23
\PackageError{pkg name}{error text}{help} [p] . . . . (D.2.7) – 446
Writes an error message 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 Hhreturni 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.

\PackageInfo{pkg name}{info text} [p]

. . . . . . . . (D.2.7) – 447

Is like \PackageWarningNoLine except that the text info text is only
written to the transcript file, and not to the monitor.

\PackageWarning{pkg name}{warn text} [p] . . . . . . (D.2.7) – 446
Writes 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 warn text is formatted in the same way as
that for \PackageError.

\PackageWarningNoLine{pkg name}{warn text} [p]

. . (D.2.7) – 446

Is like \PackageWarning except that the current line number of the
input file is not printed.

\pagebreak[n]

. . . . . . . . . . . . . . . . . . . . . (2.7.4) – 33

A recommendation to break the page at this point. The urgency of
the recommendation is given by the integral number 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

. . . . . . . . . . . . . . . . . . . (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 \pagecolor is called once more.
The col spec is the same as for \color.


\texttt{\pagename} \hfill (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{\pagename{style}} \hfill (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{\pageref{marker}} \hfill (9.2.1) – 213

Prints the number of the page where \texttt{marker} has been set by a \texttt{\label{marker}} command.

\texttt{\pagestyle{style}} [p] \hfill (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{\path{directory}} \hfill (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{\paperheight} \hfill (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{\paperwidth} and \texttt{\paperheight}.

\texttt{\paperwidth} \hfill (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{\paperwidth} and \texttt{\paperheight}.

\texttt{\par} \hfill (2.5.1) – 23

Ends the current paragraph and begins a new one. This command is equivalent to a blank line.

\texttt{\paragraph[short title]{title}} \hfill (3.3.3) – 55

The second last command in the sectioning hierarchy, coming between \texttt{\subsubsection} and \texttt{\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 || ................................. (5.3.4) – 126
\parbox[\textwidth] {\textwidth} {text} (4.7.3), (4.7.5) – 88, 90
   Produces a vertical box of width \textwidth 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 \textwidth: t for alignment with its top line, b with its bottom
line, and centered with no argument. The two additional optional
arguments: \textwidth to give the total height, and \textwidth 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
\textwidth option. The \textwidth argument may contain the parameters \textwidth,\textwidth, \textwidth, and \textwidth.

\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 as-
signed 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 \hfill (4.4.2) – 75
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:
\setlength{\partopsep}{2pt plus1pt minus1pt}

\PassOptionsToClass{options}{class} [p] \hfill (D.2.3) – 443
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] \hfill (D.2.3) – 443
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 $\perp$ \hfill (5.3.4) – 126
\Phi [m] produces $\Phi$ \hfill (5.3.1) – 125
\phi [m] produces $\phi$ \hfill (5.3.1) – 125
\Pi [m] produces $\Pi$ \hfill (5.3.1) – 125
\pi [m] produces $\pi$ \hfill (5.3.1) – 125
\pm [m] produces $\pm$ \hfill (5.3.4) – 126
\pmb{symbol} [m][a] \hfill (12.2.1) – 258

When one of the packages amsmath or amssymb has been loaded, this
command prints symbol in simulated bold face. This is done by
printing it several times slightly displaced.

\pmod{arg} [m] \hfill (5.3.8), (12.2.5) – 129, 268
Command to produce the function name ‘mod’ in formulas in the
form:
$y \pmod{a+b} = y \pmod{a + b}$

\pod{arg} [m][a] \hfill (12.2.5) – 268
With the amsopn or amsmath packages, command to produce the
function name ‘mod’ in formulas in the form:
$y \pod{a+b} = y (a + b)$

\poptabs \hfill (4.6.4) – 83
Restores the last set of tabular stops in the tabbing environment
that has been saved with \pushtabs.

\pounds produces £ \hfill (2.5.5) – 23
\Pr [m] \hfill (5.3.8) – 128
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 makeidx.sty file that generates the index environment after the program MakeIndex has processed the .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 \hfill (5.3.7) - 128
\texttt{\textbackslash propto} [m] produces \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:

\begin{verbatim}
\section{The \protect\pounds{} Sign}
\end{verbatim}

\texttt{\textbackslash providecommand}\{\texttt{\textbackslash com\_name}\}[\texttt{narg}][\texttt{opt}]{\texttt{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{\textbackslash com\_name}\}[\texttt{narg}][\texttt{opt}]{\texttt{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}\{\texttt{class}\}[\texttt{version}][\texttt{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:

\begin{verbatim}
\ProvidesClass{thesis}[1995/01/25 v3.8 U of Saigon]
\end{verbatim}

\texttt{\textbackslash ProvidesFile}\{\texttt{file\_name}\}[\texttt{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.
\ProvidesPackage{\textit{class}}[\textit{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 \textit{version} specification, if present, consists of three parts: date, version number, and additional information. Example:
\ProvidesPackage{\textit{notes}}[1995/02/13 1.2 G. Smith]
\ProvidesPackage{\textit{class}}[\textit{version}] [p] .................................. (D.2.1) – 441

\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 \textit{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{\textit{cmd}}{\textit{code}}{\textit{narg}}[\textit{opt}]{\textit{def}} [p]
.................................................................... (A.3.7) – 379
Defines \textit{cmd} in the same way as \providecommand except the definition is only valid when encoding \textit{code} is active. If the command \textit{cmd} is already defined for that encoding, it is not redefined.

\ProvideTextCommandDefault{\textit{cmd}}{\textit{code}}{\textit{narg}}[\textit{opt}]{\textit{def}} [p]
.................................................................... (A.3.7) – 380
Is the same as \DeclareTextCommandDefault except that if a default encoding definition already exists for \textit{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 \( \Psi \) .............................................. (5.3.1) – 125
\psi [m] produces \( \psi \) .............................................. (5.3.1) – 125

\psstabs ........................................................................ (4.6.4) – 83
Saves the current set of tabulator stops in the tabbing environment. It may be recalled with the \poptabs command.

\put(\textit{x},\textit{y}){\textit{pic.elem}} ........................................ (13.1.3) – 289
The positioning command within a picture environment. The picture element \textit{pic.elem} is placed with its reference point at the location \((\textit{x},\textit{y})\).

\qbezier[\textit{num}](\textit{x_1},\textit{y_1})(\textit{x_2},\textit{y_2})(\textit{x_3},\textit{y_3}) ................................ (13.1.4) – 299
This command can be given within the picture environment to draw a quadratic Bézier curve from point \((\textit{x_1},\textit{y_1})\) to \((\textit{x_3},\textit{y_3})\), using \((\textit{x_2},\textit{y_2})\) as the extra Bézier point. The curve is drawn as \textit{num} + 1 dots if the optional argument \textit{num} is given, otherwise \textit{num} is calculated automatically to produce a solid line. It is the same as \bezier except that \textit{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

\texttt{\textbackslash quotedblbase} \hspace{1em} \textbullet \hspace{1em} (G.4.5) – 503

When T1 encoding is active, prints the symbol “.”

\texttt{\textbackslash quotesinglbase} \hspace{1em} \textbullet \hspace{1em} (G.4.5) – 503

When T1 encoding is active, prints the symbol ‘.’

\texttt{\textbackslash r\{x\}} \hspace{1em} \textbullet \hspace{1em} (2.5.7) – 24

Produces a circle accent: \texttt{\textbackslash r\{o\} = ˚o}.

\texttt{\textbackslash raggedbottom} \hspace{1em} \textbullet \hspace{1em} (3.2.5) – 48

The standard page formatting for \texttt{article}, \texttt{report}, and \texttt{letter} document classes when the \texttt{twoside} option has \texttt{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{1em} \textbullet \hspace{1em} (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 \textbackslash}. See also \texttt{\begin{flushright}}.

\texttt{\textbackslash raggedright} \hspace{1em} \textbullet \hspace{1em} (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 \textbackslash}. See also \texttt{\begin{flushleft}}.

\texttt{\textbackslash raisebox\{lift\}[height][depth]\{text\}} \hspace{1em} \textbullet \hspace{1em} (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{1em} \textbullet \hspace{1em} (12.2.6) – 271

Within one of the \textit{AMS-\LaTeX} alignment environments, raises the equation number or marker by \texttt{len} above its normal position.

\texttt{\textbackslash rangle [m] produces )} \hspace{1em} \textbullet \hspace{1em} (5.4.1) – 132
\texttt{\textbackslash rceil [m] produces ]} \hspace{1em} \textbullet \hspace{1em} (5.4.1) – 132
\texttt{\textbackslash Re [m] produces \textbf{R}} \hspace{1em} \textbullet \hspace{1em} (5.3.6) – 127
\texttt{\textbackslash ref\{marker\}} \hspace{1em} \textbullet \hspace{1em} (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{1em} \textbullet \hspace{1em} (6.1.2) – 156

A command made available with the \texttt{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 \renewenvironment 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 yyyy/mm/dd, as for example 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 graphics package that scales the contents text as an LR box such that the horizontal size becomes h_length and the vertical size v_length. If either size is given as !, the one scale factor is applied to both dimensions.
H.1. Brief description of the \LaTeX\ commands 575

\resizebox*{\textwidth}{\vphantom{text}}{\text{. . . . . . . . . (6.1.2) – 156}}

The same as \texttt{\resizebox} except that the vertical size \texttt{v.length} refers to the total height plus depth of the LR box.

\reversemarginpar {\text{. . . . . . . . . (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 \texttt{\normalmarginpar}.

\texttt{\textbackslash rfloor} [m] produces \textbackslash ] \text{. . . . . . . . . (5.4.1) – 132}

\texttt{\textbackslash rhd} [m] produces \textbackslash \textgreater \text{. . . . . . . . . (5.3.3) – 125}

\texttt{\textbackslash rho} [m] produces \textbackslash \rho \text{. . . . . . . . . (5.3.1) – 125}

\texttt{\textbackslash rightbrack} [m] \text{. . . . . . . . . (5.4.1) – 131}

Adjusts the size of the bracket symbol \texttt{rbrack} to fit the height of the formula between the \texttt{\left ... \right} pair. For example, \texttt{\right}. If there is to be no matching bracket, the \texttt{\left} and \texttt{\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, \texttt{\left.}).

\texttt{\textbackslash Rightarrow} [m] produces \textbackslash \Rightarrow \text{. . . . . . . . . (5.3.5) – 127}

\texttt{\textbackslash rightarrow} [m] produces \textbackslash \rightarrow \text{. . . . . . . . . (5.3.5) – 127}

\texttt{\textbackslash rightharpoonup} [m] produces \textbackslash \rightharpoonup \text{. . . . . . . . . (5.3.5) – 127}

\texttt{\textbackslash rightharpoondown} [m] produces \textbackslash \rightharpoondown \text{. . . . . . . . . (5.3.5) – 127}

\texttt{\textbackslash rightleftharpoons} [m] produces \textbackslash \rightleftharpoons \text{. . . . . . . . . (5.3.5) – 127}

\texttt{\textbackslash rightmargin} \text{. . . . . . . . . . . . . . (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 \texttt{\setlength} command:

\texttt{\setlength{\textbackslash rightmargin}{0.5cm}}

\texttt{\textbackslash rmdefault} \text{. . . . . . . . . . . . . . . . . (A.3.1) – 372}

This command defines the family attribute that is selected with the \texttt{\rmfamily} command. It may be redefined with \texttt{\renewcommand}:

\texttt{\renewcommand{\textbackslash rmdefault}{ptm}}

\texttt{\textbackslash rmfamily} \text{. . . . . . . . . . . . . . . . . (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.

\texttt{\textbackslash Roman{\textbackslash counter}} \text{. . . . . . . . . . . . . . (8.1.4) – 183}

Prints the current value of the \texttt{\textbackslash counter} as an upper case Roman numeral.

\texttt{\textbackslash roman{\textbackslash counter}} \text{. . . . . . . . . . . . . . (8.1.4) – 183}

Prints the current value of the \texttt{\textbackslash counter} as a lower case Roman numeral.
\rotatebox{angle}{text} \hspace{156}\text{(6.1.2)}

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} \hspace{91}\text{(4.7.6)}

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 \text{[m][a]} produces | (right delimiter) \hspace{270}\text{(12.2.5)}

\rVert \text{[m][a]} produces \| (right delimiter) \hspace{270}\text{(12.2.5)}

\S \text{produces §} \hspace{23}\text{(2.5.5)}

\SS \text{produces SS, the upper case version of \ss}, ß \hspace{24}\text{(2.5.6)}

\savebox{\boxname}{[width][pos]{text}} \hspace{87}\text{(4.7.1)}

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}} \hspace{300}\text{(13.1.4)}

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 \text{produces a subscript, identical to the _ symbol.}

\sbox{\boxname}{text} \hspace{87}\text{(4.7.1)}

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} \hspace{156}\text{(6.1.2)}

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 \hspace{372}\text{(A.3.1)}

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

\texttt{\textbackslash scriptscriptstyle[m]} .................................. (5.5.2) – 146
Switches to font size \texttt{\textbackslash scriptscriptstyle} as the active font inside a math formula.

\texttt{\textbackslash scriptsize} ............................................. (4.1.2) – 62
Switches to the font size \texttt{\textbackslash scriptsize}, which is smaller than \texttt{\textbackslash footnotesize} but larger than \texttt{\textbackslash tiny}.

\texttt{\textbackslash scriptstyle[m]} ..................................... (5.5.2) – 146
Switches to font size \texttt{\textbackslash scriptstyle} as the active font inside a math formula.

\texttt{\textbackslash scshape} ............................................. (4.1.3), (A.2) – 64, 371
This declaration switches to a font in the current family and series, but with the \texttt{CAPS AND SMALL CAPS} shape attribute.

\texttt{\textbackslash searrow[m]} produces \textbullet{ } ...................... (5.3.5) – 127
\texttt{\textbackslash sec[m]} .................................................. (5.3.8) – 128
Command to produce the function name 'sec' in formulas.

\texttt{\textbackslash section[\textit{short title}]{\textit{title}}} .................. (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.

\texttt{\textbackslash section*{\textit{title}}} .................................. (3.3.3) – 55
The same as \texttt{\textbackslash section} but without a number or an entry in the table of contents.

\texttt{\textbackslash see{\textit{reference}}} .................................. (9.4.2) – 227
A command defined in the makeidx package for use with the MakeIndex program. It is called within an \texttt{\textbackslash index} command to refer to another entry in the keyword index as 'see \textit{reference}'. Given in the form:

\texttt{\textbackslash index{\textit{entry}|see{\textit{reference}}}}

Note: the above text is correct with | in place of \ for |see.

\texttt{\textbackslashseealso{\textit{reference}}} ............................. (9.4.2) – 227
Like \texttt{\textbackslash see}, but will print the text 'see also \textit{reference}' in the index. More precisely, it prints the text stored in the command \texttt{\textbackslash alsoname}. Requires the makeidx package.

\texttt{\textbackslash seename} .............................................. (D.4.1) – 460
A command defined in the package makeidx containing the text for the command \texttt{\textbackslash 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 \langle true \rangle or \langle false \rangle, 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{\lengthcmd}{length_spec} ............... (8.2) - 184
The length command with the name \lengthcmd 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{\symfnt}{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 579

\settodepth{\length.cmd}{text} \hspace{1cm} (8.2) – 184

The length command with the name \length.cmd is assigned a value equal to the depth of text below the baseline.

\settodepth{\length.cmd}{text} \hspace{1cm} (8.2) – 184

The length command with the name \length.cmd is assigned a value equal to the height of text above the baseline.

\settowidth{\length.cmd}{text} \hspace{1cm} (8.2) – 184

The length command with the name \length.cmd is assigned a value equal to the length of text as it would be set in an LR box.

\sfdefault \hspace{1cm} (A.3.1) – 372

This command defines the family attribute that is selected with the \sffamily command. It may be redefined with \renewcommand:

\renewcommand{\sfdefault}{phv}

\sffamily \hspace{1cm} (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} \hspace{1cm} (4.7.9) – 94

With the fancybox package, is a variant of \fbox, drawing a shadowed box around text; the thickness of the shadow is given by the length \shadowsize.

\sharp[m] produces ♯ \hspace{1cm} (5.3.6) – 127

\shortstack[pos]{text} \hspace{1cm} (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:

\shortstack{aa\bbb\ccc\x\yy\zzz}

\sideset{pre}{post}\symbol[m][a] \hspace{1cm} (12.2.2) – 262

With the amsmath package, places superscripts and subscripts snugly before (\pre) and after (\post) the math symbol \symbol. Example:

$\sideset{-}_{\dag^*}\prod^\dag$\ yields $\prod^\dag$

\Sigma[m] produces Σ \hspace{1cm} (5.3.1) – 125

\sigma[m] produces σ \hspace{1cm} (5.3.1) – 125

\signature{name} \hspace{1cm} (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{\textbackslash sim}\texttt{} \texttt{[m]} \texttt{} produces \textasciitilde . . . . . . . . . . . . . . . . . . . . . . (5.3.4) – 126

\texttt{\textbackslash sineq}\texttt{} \texttt{[m]} \texttt{} produces \textasciitilde \textasciitilde . . . . . . . . . . . . . . . . . . . . . . (5.3.4) – 126

\texttt{\textbackslash sin}\texttt{} \texttt{[m]} \texttt{} produces \texttt{\textbackslash sin}. . . . . . . . . . . . . . . . . . . . . . (5.3.8) – 128

Command to produce the function name ‘sin’ in formulas.

\texttt{\textbackslash sinh}\texttt{} \texttt{[m]} \texttt{} produces \texttt{\textbackslash sinh}. . . . . . . . . . . . . . . . . . . . . . (5.3.8) – 128

Command to produce the function name ‘sinh’ in formulas.

\texttt{\textbackslash sldefault} \texttt{} . . . . . . . . . . . . . . . . . . . . . . (A.3.1) – 372

This command defines the shape attribute that is selected with the \texttt{\textbackslash slshape} command. It may be redefined with \texttt{\textbackslash renewcommand}:

\texttt{\textbackslash renewcommand}\texttt{\{}\texttt{\textbackslash sldefault}\texttt{\}{\texttt{s}l} \texttt{\}}

\texttt{\textbackslash sloppy} \texttt{} . . . . . . . . . . . . . . . . . . . . . . (2.8.3) – 36

After this command has been given, word spacings are allowed to stretch more generously than usual so that paragraphs are broken up into lines with fewer word divisions. It is countermanded by \texttt{\textbackslash fussy}. See also \texttt{\textbackslash begin\{}\texttt{sloppypar}\texttt{\}}.

\texttt{\textbackslash slshape} \texttt{} . . . . . . . . . . . . . . . . . . . . . . (4.1.3), (A.2) – 64, 371

This declaration switches to a font in the current family and series, but with the \textit{slanted} shape attribute.

\texttt{\textbackslash small} \texttt{} . . . . . . . . . . . . . . . . . . . . . . (4.1.2) – 62

Switches to the font size \texttt{\textbackslash small}, which is smaller than \texttt{\textbackslash normalsize} but larger than \texttt{\textbackslash footnotesize}.

\texttt{\textbackslash smallskip} \texttt{} . . . . . . . . . . . . . . . . . . . . . . (2.7.3) – 32

Inserts large vertical spacing of amount \texttt{\textbackslash smallskipamount}. See also \texttt{\textbackslash medskip} and \texttt{\textbackslash bigskip}.

\texttt{\textbackslash smallskipamount}

Standard value for the amount of vertical spacing that is inserted with the command \texttt{\textbackslash smallskip}. May be changed with the \texttt{\textbackslash setlength} command:

\texttt{\textbackslash setlength}\texttt{\}{\texttt{\textbackslash smallskipamount}\texttt{\}{\texttt{1ex plus0.5ex minus0.3ex}}}

\texttt{\textbackslash smash}\texttt{\}{\texttt{\textbackslash pos}\texttt{\}}\texttt{\}{\texttt{text}\texttt{\}}\texttt{\}{\texttt{\textbackslash m}\texttt{\}}\texttt{\}{\texttt{\textbackslash a]} \texttt{} . . . . . . . . . . . . . . . . . . . . . . (12.2.5) – 269

With the amsmath package, this \TeX{} command acquires an optional argument \textit{pos} that may be \texttt{b} or \texttt{t}, to effectively zero the depth or height of the \textit{text}. With no \textit{pos}, both height and depth are zeroed.

\texttt{\textbackslash smile}\texttt{} \texttt{[m]} \texttt{} produces \textcircled{\textbullet}. . . . . . . . . . . . . . . . . . . . . . (5.3.4) – 126

\texttt{\textbackslash sp} \texttt{} produces a superscript, identical to the \textasciicircum{} symbol.

\texttt{\textbackslash spadesuit}\texttt{} \texttt{[m]} \texttt{} produces ♠ . . . . . . . . . . . . . . . . . . . . . . (5.3.6) – 127

\texttt{\textbackslash sqcap}\texttt{} \texttt{[m]} \texttt{} produces □ . . . . . . . . . . . . . . . . . . . . . . (5.3.3) – 125

\texttt{\textbackslash sqcup}\texttt{} \texttt{[m]} \texttt{} produces △ . . . . . . . . . . . . . . . . . . . . . . (5.3.3) – 125

\texttt{\textbackslash sqrt}\texttt{[n]}\texttt{\}{\texttt{arg}\texttt{\}}\texttt{\}{\texttt{\textbackslash m]} \texttt{} . . . . . . . . . . . . . . . . . . . . . . (5.2.4) – 122

Basic math command to produce a root sign. The height and length of the sign are made to fit the contents \textit{arg}. The optional argument \textit{n} is the degree of the root: \texttt{\textbackslash sqrt}\texttt{[3]}\texttt{\}{\texttt{2\}} = \sqrt[3]{2}, \texttt{\textbackslash sqrt}\texttt{[2]}\texttt{\}{\texttt{2\}} = \sqrt{2}. 
H.1. Brief description of the \LaTeX\ commands 581

\sqsubseteq [m] produces ⊑ ........................................ (5.3.4) – 126
\sqsupseteq [m] produces ⊒ ........................................ (5.3.4) – 126
\ss produces ß ......................................................... (2.5.6) – 24
\stackrel{\text{upper}}{\text{lower}} .................................. (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 * .................................................. (5.3.3) – 125
\stepcounter{\text{counter}} ........................................ (8.1.3) – 182
Increases the value of the number stored in counter by one.
\stretch{\text{decimal.num}} ....................................... (8.2) – 184
A rubber length with a natural value of 0 pt but with a stretchability that is \text{decimal.num} times that of \fill.

\subitem{\text{sub.entry}} .......................................... (9.4.1) – 225
In the index environment, a command to produce a second-level entry after an \item command.

\subparagraph[\text{short title}]{\text{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*{\text{title}} ........................................... (3.3.3) – 55
The same as \subparagraph but without a number or an entry in the table of contents.

\subsection[\text{short title}]{\text{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*{\text{title}} ................................................. (3.3.3) – 55
The same as \subsection but without a number or an entry in the table of contents.

\substack{1st line\ldots\ldots\ldots 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 { }.
\texttt{\subsubitem{sub\_sub\_entry}} \hspace{1em} (9.4.1) - 225

In the index environment, a command to produce a third-level entry under a \texttt{\subitem} command.

\texttt{\subsubsection[short form]{title}} \hspace{1em} (3.3.3) - 55

The command in the sectioning hierarchy coming between \texttt{\subsection} and \texttt{\paragraph}. It formats \textit{title} with the current subsection number and an automatic sequential sub-subsection number. If the optional \textit{short title} is given, it appears in place of \textit{title} in the table of contents.

\texttt{\subsubsection*{title}} \hspace{1em} (3.3.3) - 55

The same as \texttt{\subsubsection} but without a number or an entry in the table of contents.

\texttt{\subset [m]} produces \( \subset \) \hspace{1em} (5.3.4) - 126
\texttt{\subseteq [m]} produces \( \subseteq \) \hspace{1em} (5.3.4) - 126
\texttt{\succ [m]} produces \( \succ \) \hspace{1em} (5.3.4) - 126
\texttt{\succeq [m]} produces \( \succeq \) \hspace{1em} (5.3.4) - 126
\texttt{\sum [m]} produces \( \sum \) \hspace{1em} (5.2.5) - 123
\texttt{\sup [m]} \hspace{1em} (5.3.8) - 128

Command to produce the function name ‘sup’ in formulas. A lower limit may be set as a subscript.

\texttt{\supset [m]} produces \( \supset \) \hspace{1em} (5.3.4) - 126
\texttt{\supseteq [m]} produces \( \supseteq \) \hspace{1em} (5.3.4) - 126
\texttt{\surd [m]} produces \( \sqrt{} \) \hspace{1em} (5.3.6) - 127
\texttt{\swarrow [m]} produces \( \swarrow \) \hspace{1em} (5.3.5) - 127
\texttt{\symbol{n}} \hspace{1em} (4.1.6) - 66

Produces the symbol in the current character font that is stored under the internal number \textit{n}.

\texttt{\t{xy}} \hspace{1em} (2.5.7) - 24

Produces a ‘tie-after’ accent over two letters: \texttt{\t{oo}} = \( \ddot{o} \).

\texttt{\tabbingsep} \hspace{1em} (4.6.4) - 83

Determines the spacing between the text \textit{ltext} and the current tabular stop when \textit{ltext\textbackslash'} is given in a \texttt{tabbing} environment. A new value may be assigned with the \texttt{\setlength} command.
H.1. Brief description of the LaTeX commands 583

\tabcolsep .............................. (4.8.2) – 98
Determine the half-column spacing in the tabular environment. A
new value may be assigned with the \setlength command:
\setlength{\tabcolsep}{3mm}

\tableofcontents ........................... (3.4.2) – 58
Prints the table of contents from information in the sectioning com-
mands and additional entries.

\tablename ................................. (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.

\tabularnewline[len] ............................ (4.8.1) – 98
Terminates a row in the tabular or array environments, adding
vertical spacing \textit{len} if it is specified. This is equivalent to \,
\textbackslash{[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{\raggedright} is given in the last column, then this command
\textit{must} be used in place of \textbackslash{.}

\tag{mark} \textit{[m][a]} .......................... (12.2.6) – 271
Within one of the \texttt{\textbackslash{}{\texttt{AMS-L\textbackslash{}{}\LaTeX}}} alignment environments, prints \textit{mark}
in place of the equation number, in parentheses. The \texttt{*}-form prints
it without parentheses.

\tan \textit{[m]} ................................ (5.3.8) – 128
Command to produce the function name ‘\texttt{tan}’ in formulas.

\tanh \textit{[m]} ............................... (5.3.8) – 128
Command to produce the function name ‘\texttt{tanh}’ in formulas.

\tau \textit{[m]} produces $\tau$ ........................ (5.3.1) – 125
\tbinom{over}{under} \textit{[m][a]} ............... (12.2.3) – 264
With the \texttt{amsmath} package, produces a binomial as \texttt{\binom} does, but
in \texttt{textstyle} size.

\telephone{\textit{number}} .................... (16.1), (16.2) – 352, 356
In the \texttt{\textbackslash{}letter} document class, enters the sender’s telephone number.
In the standard \texttt{\textbackslash{}L\textbackslash{}\LaTeX}\texttt{\textbackslash{}letter.sty}, \textit{number} is only output if \texttt{\address}
has not been called. It is intended to be used in company letter styles
such as \texttt{\textbackslash{}mpletter}.

\TeX \textit{produces \texttt{\textbackslash{}L\textbackslash{}\LaTeX}} .......................... (2.1) – 19
\textit{[m][a]} ................................ (12.2.1) – 259
When one of the packages \texttt{amsmath} or \texttt{amstext} has been loaded, this
command prints \textit{short\texttt{.text}} as normal text within a math formula. If
used in subscripts or superscripts, automatic sizing takes place.
\textsym{name} \hspace{1cm} (D.6) - 462

An alternative means to produce certain special symbols that otherwise are only available in math mode or through ligature combinations:
\textbullet \hspace{0.5cm} \textemdash \hspace{0.5cm} \textendash \hspace{0.5cm} \textexclamdown \hspace{0.5cm} \textperiodcentered \hspace{0.5cm} \textquestiondown \hspace{0.5cm} \textquotedblleft \hspace{0.5cm} \textquotedblright \hspace{0.5cm} \textquoteleft \hspace{0.5cm} \textquoteright \hspace{0.5cm} \textvisiblespace \hspace{0.5cm} \textasciicircum \hspace{0.5cm} \textasciitilde \hspace{0.5cm} \textbackslash \hspace{0.5cm} \textbar \hspace{0.5cm} \textgreater \hspace{0.5cm} \textless

\textbf{\text{text}} \hspace{1cm} (4.1.4) - 65

This command sets its argument in a font in the current family and shape, but with the \textbf{bold} series attribute. It is equivalent to \textbf{\text{bfseries text}}.

\textcircled{\text{char}} \hspace{1cm} (D.6) - 462

Produces the specified character in a circle: \textcircled{s} = 8.

\textcolor{col spec}{\text{text}} \hspace{1cm} (6.2) - 167

A command made available with the color package. The text is set in the specified color. The \textcolor{col spec}{col spec} is the same as for \textcolor{col spec}{color}.

\textcompwordmark \hspace{1cm} (D.6) - 462

Prints an invisible character that may be used to break ligatures:
\textcompwordmark = fi

\textfloatsep \hspace{1cm} (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:
\textfloatsep = 20pt plus 2pt minus 4pt

\textfraction \hspace{1cm} (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 = decimal \textfraction

\textheight \hspace{1cm} (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 \textheight command:
\textheight = 45\baselineskip
\textit{text} 
\hspace{1cm} (4.1.4) – 65

This command sets its argument in a font in the current family and series, but with the italic shape attribute. It is equivalent to \{\itshape text\}.

\textmd{text} 
\hspace{1cm} (4.1.4) – 65

This command sets its argument in a font in the current family and shape, \textbf{but with the medium series attribute}. It is equivalent to \{\mdseries text\}.

\textnormal{text} 
\hspace{1cm} (4.1.3) – 65

This command sets its argument in the font with the default family, series, and shape attributes. It is equivalent to \{\normalfont text\}.

\textquotedbl . 
\hspace{1cm} (G.4.5) – 503

When T1 encoding is active, prints the symbol ".

\textregistered produces ® 
\hspace{1cm} (D.6) – 462

\textsuperscript{char} 
\hspace{1cm} (D.6) – 462

Produces a superscript in the current text, rather than math, font: \textsuperscript{12} = 12.

\texttrademark produces ™ 
\hspace{1cm} (D.6) – 462

\texttt{text} 
\hspace{1cm} (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] 
\hspace{1cm} (5.5.2) – 146

Switches to font size \textstyle inside a math formula.

\textsuperscript{char} 
\hspace{1cm} (D.6) – 462

Sets text in a font in the current series and shape, but with the slanted shape attribute. It is equivalent to \{\slshape text\}.

\textup{text} 
\hspace{1cm} (4.1.4) – 65

Sets text in a font in the current family and series, \textbf{but with the upright shape attribute}. It is equivalent to \{\upshape text\}. 
Appendix H. Command Summary

\textwidth ............................... (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}} [m][a] ........................ (12.2.3) - 264
With the amsmath package, produces a fraction as \frac does, but in textstyle size.

\TH .................................. (G.4.5) - 503
When T1 encoding is active, prints the character Þ.

\th .................................. (G.4.5) - 503
When T1 encoding is active, prints the character þ.

\thanks{\text{footnote text}} ............................... (3.3.1) - 54
Produces a footnote to an author's name on the title page when \maketitle is called.

\thecounter ................................ (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}{\text{def}}.

\Theta [m] produces Θ ................................ (5.3.1) - 125
\theta [m] produces θ ................................ (5.3.1) - 125

\thicklines ................................ (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] ............................... (12.2.5) - 269
With the amsmath package, this is an alias for \; ; a thick space in a math formula.

\thinlines ................................ (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 .................................. (12.2.5) - 269
With the amsmath package, this is an alias for \,, a thin space in a math formula.

\thisfancypage{cmds1}{cmds2} .......................... (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} .................................. (3.2) - 42

Changes the page style for the current page only. Possible values for \textit{style} are: plain, empty, headings, and myheadings.

\tilde{x} \text{ [m]} ........................................ (5.3.9) - 129

Produces a tilde (squiggle) over the math variable $x$: $\tilde{a}$.

$\tilde{T}$ \text{ [m][a]} ........................................ (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.

\texttt{\tiny} ........................................... (4.1.2) - 62

Switches to the smallest font size available \texttt{\tiny}, smaller than \texttt{\scriptsize}.

\texttt{\times [m]} produces $\times$ .................................. (5.3.3) - 125

\texttt{\title{text}} ......................................... (3.3.1) - 52

Enters the \texttt{text} for the title page that is produced by \texttt{\maketitle}.

\texttt{\to [m]} produces $\to$ .................................. (5.3.5) - 127

\texttt{\today} ........................................... (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 \LaTeX{} commands \texttt{\day}, \texttt{\month}, and \texttt{\year}.

\texttt{\top [m]} produces $\top$ .................................. (5.3.6) - 127

\texttt{\topfigrule} ........................................ (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.

\texttt{\renewcommand{\topfigrule}}{\vspace*{-.4pt} \\
\hspace{\columnwidth}{.4pt}}

\texttt{\topfraction} ........................................ (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}}{\textit{decimalfrac}}

\texttt{\topmargin} ........................................ (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:

\texttt{\setlength{\topmargin}{0.5in}}
Appendix H. Command Summary

\texttt{topnumber} ............................. (7.3) - 171
The maximum number of floats that may appear at the top of a page.
A new value is assigned with:
\texttt{\setcounter{topnumber}\{num\}}
\texttt{\topsep} .............................. (4.4.2), (5.5.4) - 75, 149
The extra vertical spacing, in addition to \texttt{\parskip}, inserted at the
beginning and end of a listing environment. When document class
option \texttt{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 \texttt{\setlength} command:
\texttt{\setlength{\topsep}\{4pt plus2pt minus2pt\}}
\texttt{\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
\texttt{\setlength} command:
\texttt{\setlength{\topskip}\{12pt\}}
\texttt{\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 \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[6\totalheight]{text}}
\texttt{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:
\texttt{\setcounter{totalnumber}\{num\}}
\texttt{\triangle} [m] produces \(\triangle\) .......................... (5.3.6) - 127
\texttt{\triangledownleft} [m] produces \(<\) .......................... (5.3.3) - 125
\texttt{\triangleright} [m] produces \(\triangleleft\) ............................ (5.3.3) - 125
\texttt{ttdefault} ................................. (A.3.1) - 372
This command defines the family attribute that is selected with the
\texttt{\ttfamily} command. It may be redefined with \texttt{\renewcommand}:
\texttt{\renewcommand{\ttfamily}{pcr}}
\texttt{\ttfamily} ................................. (4.1.3), A.2) - 64, 371
This declaration switches to a font in the current series and shape,
but with the \texttt{typewriter} family attribute.
\texttt{\twocolumn[text]} .......................... (3.2.7) - 51
Begins a new page and switches to two-column page format. The
optional \texttt{text} is set in one column extending over the two columns.
\typein[\texttt{cmd}]{message} \hspace{1em} (9.1.3) \hspace{1em} 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} \hspace{1em} (9.1.3) \hspace{1em} 211

Prints the \textit{message} to the monitor and continues the processing. The \textit{message} is also written to the \texttt{.log} file.

\texttt{u\{x\}} \hspace{1em} (2.5.7) \hspace{1em} 24

Produces a breve accent: \texttt{u\{o\} = ô}.

\unboldmath \hspace{1em} (5.4.9) \hspace{1em} 143

Countermands the \texttt{\boldmath} command. It must be given outside of the math mode. Afterwards, formulas are set in standard ‘\textit{math italics}’ once more.

\underbrace{\textit{sub\_form}} [\texttt{m}] \hspace{1em} (5.4.4) \hspace{1em} 136

Produces a horizontal curly brace beneath the math formula \textit{sub\_form}. Any following subscript will be placed centered below the horizontal brace.

\texttt{underbrace{a+b}: \underbrace{a+b}}

\texttt{underbrace{x+y+z}_{\xi\eta\zeta}: \underbrace{x+y+z}_{\xi\eta\zeta}}

\underleftarrow{expr} [\texttt{m}[a]] \hspace{1em} (12.2.2) \hspace{1em} 262

With the \texttt{amsmath} package, places a long leftwards pointing arrow beneath the mathematical expression \textit{expr}.

\underrightarrow{expr} [\texttt{m}[a]] \hspace{1em} (12.2.2) \hspace{1em} 262

With the \texttt{amsmath} package, places a long rightwards pointing arrow beneath the mathematical expression \textit{expr}.

\underline{\textit{text}} \hspace{1em} (5.4.4) \hspace{1em} 136

Underlines the \textit{text} in both math and normal text modes:
\texttt{underline{Text} = Text}.

\underrightarrow{expr} [\texttt{m}[a]] \hspace{1em} (12.2.2) \hspace{1em} 262

With the \texttt{amsmath} package, places a long rightwards pointing arrow beneath the mathematical expression \textit{expr}.
\underset{\text{char}}{\text{symbol}} \text{[m][a]} \ldots \ldots \ldots \ldots \ldots \ldots \ldots (12.2.2) - 262

With the amsmath package, places \textit{char} below the \textit{math symbol} \textbackslash symbol in subscript size.

\unitlength \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (13.1.1) - 288

Defines the unit of length for the following \textit{picture environments}.
A value is assigned with the \texttt{\setlength} command:
\begin{verbatim}
\setlength{\unitlength}{1.2cm}
\end{verbatim}

\unlhd [m] produces \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (5.3.3) - 125

\unrhd [m] produces \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (5.3.3) - 125

\Uparrow [m] produces \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (5.3.5) - 127

\uparrow [m] produces \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (5.3.5) - 127

\updefault \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (A.3.1) - 372

This command defines the shape attribute that is selected with the \texttt{\upshape} command. It may be redefined with \texttt{\renewcommand}:
\begin{verbatim}
\renewcommand{\updefault}{n}
\end{verbatim}

\Updownarrow [m] produces \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (5.3.5) - 127

\updownarrow [m] produces \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (5.3.5) - 127

\uplus [m] produces \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (5.3.3) - 125

\uproot{\text{shift}} [m][a] \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (12.2.5) - 269

With the amsmath package, used in the index to a \texttt{\sqrt} command to shift it slightly upwards. The \textit{shift} is a number specifying how many units to move it. Example:
\begin{verbatim}
\sqrt[\leftroot{-1}\uproot{3}\beta]{k}
\end{verbatim}

\url{\text{address}} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (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 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{\verb}. Examples:
\begin{verbatim}
\url{w_smith@xyz.com} or \url|w_smith@xyz.com|
\end{verbatim}

\urlstyle{\text{style}} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (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.
\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
example 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} = Ĉ.

\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 \textquoteleft lim\textquoteright in formulas  . . . (12.2.5) – 268
\varliminf[m][a] produces \textquoteleft lim\textquoteright in formulas  . . . (12.2.5) – 268
\varlimsup[m][a] produces \textquoteleft lim\textquoteright 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 \textquoteleft lim\textquoteright in formulas  . . . (12.2.5) – 268
\varkappa[m] produces $\varkappa$  . . . . . . . . . . . . . . . . . . . . (5.3.1) – 125
\varsigma[m] produces $\varsigma$  . . . . . . . . . . . . . . . . . . . . (5.3.1) – 125
\vartheta [m] produces $\vartheta$ . . . . . . . . . . . . . . . . . . (5.3.1) – 125
\vdash [m] produces $\vdash$ . . . . . . . . . . . . . . . . . . . . (5.3.6) – 127
\vdots [m] produces $\ldots$ . . . . . . . . . . . . . . . . . . . . . (5.2.6) – 123
\vec{\text{x}} [m] . . . . . . . . . . . . . . . . . . . . . . . . . . . (5.3.9) – 129
A vector symbol over the variable x: $\vec{a}$.
\Vec{\text{x}} [m][a] . . . . . . . . . . . . . . . . (12.2.2) – 263
With the amsmath package, can be used like \vec, but with multiple \LaTeX math accents the positioning will be correct.
\vector(\Delta x,\Delta y){length} . . . . . . . . . . . . . . . (13.1.4) – 295
A picture element command within a 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 length argument is the actual length in units of \unitlength. For slanted arrows, 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 \put or \multiput command.
\vee [m] produces $\vee$ . . . . . . . . . . . . . . . . . . (5.3.3) – 125
\verb|source_text| . . . . . . . . . . . . . . . . . . . . . . (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 source_text.
\verb*|source_text| . . . . . . . . . . . . . . . . . . . . . . (4.9) – 110
The same as \verb except that blanks are made visible with the symbol $\omega$.
\vfill . . . . . . . . . . . . . . . . . . . . . . . . . . . . . (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 \vspace{\fill}.
\visible . . . . . . . . . . . . . . . . . . . . . . . . . . . (15.1.2) – 326
In slides class, a declaration that countermands a previous \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 \invisible is given. It is used for making overlays.
\vline . . . . . . . . . . . . . . . . . . . . . . . . . . . (4.8.1) – 97
Prints a vertical rule within the column entry of a table in the tabular environment.
\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 \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 \setlength command:
\setlength{\voffset}{-1in}

\vpageref[current][non-current]{key} .................. (9.2.4) – 215
With the varioref package, this is equivalent to on page
\pageref{key}, unless \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.

\vref{key} ........................................ (9.2.4) – 215
With the varioref package, this is equivalent to \ref{key} on page
\pageref{key}, unless \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'.

\vspace{height} ................................. (2.7.3) – 32
Produces vertical spacing of length height. It is ignored if it occurs
at the beginning or end of a page.

\vspace*[height] ................................. (2.7.3) – 32
Produces vertical spacing of length height even at the beginning or
end of a page.

\wedge [m] produces ^ ................................. (5.3.3) – 125
\whiledo{test}{do_text} ............................. (8.3.5) – 193
A conditional command available when the standard package ifthen
has been loaded. The do_text is inserted repeatedly as long as
the logical statement test evaluates to \(true\). The logical state-
ment may be relational (two numbers with one of < = > be-
tween them), an even–odd test (\isodd{number}), a comparison
of two texts (\equal{text1}{text2}), a comparison of two lengths
\lengthtest{length1}{length2}, op is one of < = >), or a test
of a boolean switch (\bool{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 \( and
\).

\widehat{xyz} [m] ................................. (5.3.9) – 129
Produces a wide \hat symbol over several characters:
\widehat{xyz} = \hat{x\hat{y}z}. 
\texttt{\textbackslash \textit{widetilde}{xyz}} [m] \hspace{1cm} \textit{...} \hspace{1cm} (5.3.9) \hspace{1cm} 129

Produces a wide \textit{\textit{tilde}} symbol over several characters:
\texttt{\textit{\textbackslash \textit{widetilde}{xyz} = x\overline{yz}}.}

\texttt{\textbackslash \textit{width}} \hspace{1cm} \textit{...} \hspace{1cm} (4.7.1) \hspace{1cm} 86

A length parameter equal to the natural width of a box; it may only be used in the \textit{width} specification of \texttt{\textbackslash \textit{makebox}}, \texttt{\textbackslash \textit{framebox}}, or \texttt{\textbackslash \textit{savebox}}, or in the \textit{height} specification of a \texttt{\textbackslash \textit{parbox}} or a \texttt{\textit{minipage}} environment.

\texttt{\textit{\textbackslash \textit{framebox}[2\textit{\textbackslash \textit{width}}\texttt{\textit{\{text}}}}

\texttt{\wp} [m] produces $\wp$. \hspace{1cm} (5.3.6) \hspace{1cm} 127

\texttt{\wr} [m] produces $\wr$. \hspace{1cm} (5.3.3) \hspace{1cm} 125

\texttt{\Xi} [m] produces $\Xi$. \hspace{1cm} (5.3.1) \hspace{1cm} 125

\texttt{\xi} [m] produces $\xi$. \hspace{1cm} (5.3.1) \hspace{1cm} 125

\texttt{\textit{xleftarrow[below] \{above\} [m][a] \textit{...} \textit{...} \textit{...} \textit{...} \textit{(12.2.2)} \textit{- 262}}

With the \texttt{\textit{amsmath}} package, draws leftward pointing arrow with \texttt{\textit{above}} printed over it in superscript size, and optionally \texttt{\textit{below}} beneath it in subscript size.

\texttt{\textit{xrightarrow[below] \{above\} [m][a] \textit{...} \textit{...} \textit{...} \textit{...} \textit{(12.2.2)} \textit{- 262}}

With the \texttt{\textit{amsmath}} package, draws rightward pointing arrow with \texttt{\textit{above}} printed over it in superscript size, and optionally \texttt{\textit{below}} beneath it in subscript size.

\texttt{\texttt{\textit{\textbackslash \textit{zeta}} \textit{[m] \textit{produces \zeta}} \hspace{1cm} \textit{...} \hspace{1cm} (5.3.1) \hspace{1cm} 125}
H.2 Summary tables and figures

Table H.1  Font attribute commands  (4.1.3) – p. 64.
\rmfamily \textrm{text}  Roman
\sffamily \textsf{text}  sans serif
\ttfamily \texttt{text}  typewriter
\upshape \textup{text}  upright
\itshape \textit{text} italic
\slshape \textsl{text} slanted
\scshape \textsc{text} SMALL CAPS
\mdseries \textmd{text} medium
\bfseries \textbf{text} bold face

Table H.2  Math alphabet commands  (5.4.2) – p. 133.
\mathrm{text} Roman
\mathsf{text} sansserif
\mathnormal{text} normal
\mathit{text} italic
\mathbf{text} boldface
\mathcal{text} CAL

Table H.3  Font sizes  (4.1.2) – p. 62.
\tiny smallest
\scriptsize very small
\footnotesize smaller
\small small
\normalsize normal
\large large
\Large larger
\LARGE even larger
\huge still larger
\Huge largest

Table H.4  \TeX{} 2.09 font declarations  (F.2.1) – p. 485.
\textup{text}  \Gamma \Pi \Phi 
\textsc{text}  \textsc{cal} CAL
\textbf{text}  \textbf{cal} CAL
\textit{text}  \textit{cal} CAL
\texttt{text}  \texttt{cal} CAL
\textsl{text}  \textsl{cal} CAL
### Table H.5 Dimensions (2.4.1) – p. 21.

<table>
<thead>
<tr>
<th>Unit</th>
<th>Description</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>Accent</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>ö</td>
<td>ö</td>
</tr>
<tr>
<td>ȯ</td>
<td>ȯ</td>
</tr>
<tr>
<td>ŏ</td>
<td>ŏ</td>
</tr>
<tr>
<td>ō</td>
<td>ō</td>
</tr>
<tr>
<td>ô</td>
<td>ô</td>
</tr>
<tr>
<td>õ</td>
<td>õ</td>
</tr>
<tr>
<td>o̅</td>
<td>o̅</td>
</tr>
<tr>
<td>o˚</td>
<td>o˚</td>
</tr>
<tr>
<td>o̬</td>
<td>o̬</td>
</tr>
<tr>
<td>o̝</td>
<td>o̝</td>
</tr>
</tbody>
</table>

### Table H.7 Special letters from other languages (2.5.6) – p. 24.

<table>
<thead>
<tr>
<th>Letter</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>œ</td>
<td>œ</td>
</tr>
<tr>
<td>Œ</td>
<td>Œ</td>
</tr>
<tr>
<td>æ</td>
<td>æ</td>
</tr>
<tr>
<td>Æ</td>
<td>Æ</td>
</tr>
<tr>
<td>˚a</td>
<td>˚a</td>
</tr>
<tr>
<td>˚A</td>
<td>˚A</td>
</tr>
<tr>
<td>¡</td>
<td>¡</td>
</tr>
<tr>
<td>ø</td>
<td>ø</td>
</tr>
<tr>
<td>Ø</td>
<td>Ø</td>
</tr>
<tr>
<td>ł</td>
<td>ł</td>
</tr>
<tr>
<td>Ł</td>
<td>Ł</td>
</tr>
<tr>
<td>ß</td>
<td>ß</td>
</tr>
<tr>
<td>SS</td>
<td>SS</td>
</tr>
<tr>
<td>¿</td>
<td>¿</td>
</tr>
</tbody>
</table>

### Table H.8 Special symbols (2.5.5) – 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>
<tr>
<td>¶</td>
<td>¶</td>
</tr>
<tr>
<td>©</td>
<td>©</td>
</tr>
<tr>
<td>℃</td>
<td>℃</td>
</tr>
<tr>
<td>℉</td>
<td>℉</td>
</tr>
</tbody>
</table>

### Table H.9 Command symbols (2.5.4) – 23.

<table>
<thead>
<tr>
<th>Command</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>$</td>
</tr>
<tr>
<td>%</td>
<td>%</td>
</tr>
<tr>
<td>{</td>
<td>{</td>
</tr>
<tr>
<td>_</td>
<td>_</td>
</tr>
<tr>
<td>&amp;</td>
<td>&amp;</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>Letter</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>α</td>
</tr>
<tr>
<td>β</td>
<td>β</td>
</tr>
<tr>
<td>γ</td>
<td>γ</td>
</tr>
<tr>
<td>δ</td>
<td>δ</td>
</tr>
<tr>
<td>ε</td>
<td>ε</td>
</tr>
<tr>
<td>ζ</td>
<td>ζ</td>
</tr>
<tr>
<td>η</td>
<td>η</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>ζ</td>
<td>\zeta</td>
</tr>
<tr>
<td>ξ</td>
<td>\xi</td>
</tr>
</tbody>
</table>

#### Upper case letters

<table>
<thead>
<tr>
<th>Letter</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Γ</td>
<td>Γ</td>
</tr>
<tr>
<td>Δ</td>
<td>Δ</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>\pm</th>
<th>\cap</th>
<th>\circ</th>
<th>\bigcirc</th>
<th>\mp</th>
<th>\cup</th>
<th>\bullet</th>
<th>\Box</th>
<th>\times</th>
<th>\uplus</th>
<th>\diamond</th>
<th>\Diamond</th>
<th>\div</th>
<th>\sqcap</th>
<th>\lefttriangleleft</th>
<th>\bigtriangleup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>±</td>
<td>∩</td>
<td>◦</td>
<td>ⓐ</td>
<td>⊓</td>
<td>∪</td>
<td>•</td>
<td>□</td>
<td>×</td>
<td>⊎</td>
<td>⋄</td>
<td>♦</td>
<td>÷</td>
<td>⊓</td>
<td>♦</td>
<td>△</td>
</tr>
</tbody>
</table>

### Table H.12  Relational symbols  (5.3.4) - p. 126.

| Symbol | \leq | \geq | \neq | \sim | \ll | \gg | \doteq | \simeq | \subset | \supset | \approx | \asymp | \equiv | \equiv | \preceq | \succeq | \smile | \frown |
|--------|------|------|-----|-----|-----|-----|-------|------|--------|--------|-------|------|------|------|------|-------|-------|-------|-------|
|        | ≤   | ≥   | ≠   | ~   | ≪  | ≫  | ≃     | ≍    | ⊂      | ⊃      | ≈     | ≍    | ≊    | ≃    | ≪    | ≃     | ≪    | ≍     | ≋    |

### Table H.13  Negated relational symbols  (5.3.4) - p. 127.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>\not&lt;</th>
<th>\not\leq</th>
<th>\not\geq</th>
<th>\not\equiv</th>
<th>\not\preceq</th>
<th>\not\succeq</th>
<th>\not\approx</th>
<th>\not\cong</th>
<th>\not\subseteq</th>
<th>\not\supseteq</th>
<th>\not\asymp</th>
<th>\not\bowtie</th>
<th>\not\prec</th>
<th>\not\succ</th>
<th>\not\sim</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>≠</td>
<td>≠</td>
<td>≠</td>
<td>≠</td>
<td>≠</td>
<td>≠</td>
<td>≠</td>
<td>≠</td>
<td>≠</td>
<td>≠</td>
<td>≠</td>
<td>≠</td>
<td>≠</td>
<td>≠</td>
<td>≠</td>
</tr>
</tbody>
</table>

### Table H.14  Brackets  (5.4.1) - p. 132.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>\lfloor</th>
<th>\rfloor</th>
<th>\lceil</th>
<th>\rceil</th>
<th>\langle</th>
<th>\rangle</th>
<th>\uparrow</th>
<th>\Downarrow</th>
<th>\updownarrow</th>
<th>\Uparrow</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(</td>
<td>)</td>
<td>[</td>
<td>]</td>
<td>{</td>
<td>}</td>
<td>|</td>
<td>|</td>
<td></td>
<td>|</td>
</tr>
</tbody>
</table>
Table H.15  Arrows  (5.3.5) - p. 127.

<table>
<thead>
<tr>
<th>Command</th>
<th>\leftarrow</th>
<th>\gets</th>
<th>\longrightarrow</th>
<th>\Uparrow</th>
</tr>
</thead>
<tbody>
<tr>
<td>← \leftarrow</td>
<td>← \gets</td>
<td>← \longrightarrow</td>
<td>↑ \Uparrow</td>
<td></td>
</tr>
<tr>
<td>⇐ \Leftarrow</td>
<td>⇐ \Leftarrow</td>
<td>← \longrightarrow</td>
<td>↑ \Uparrow</td>
<td></td>
</tr>
<tr>
<td>⇑ \Uparrow</td>
<td>□ \Box</td>
<td>□ \Box</td>
<td>□ \Box</td>
<td></td>
</tr>
</tbody>
</table>

The underlined commands in Tables H.11, H.12, H.15, and H.16 can only be used in \LaTeX\textsuperscript{2} if one of the packages latexsym or amsfonts has been loaded.

Table H.16  Miscellaneous symbols  (5.3.6) - p. 127.

<table>
<thead>
<tr>
<th>Command</th>
<th>\aleph</th>
<th>\prime</th>
<th>\forall</th>
<th>\Box</th>
</tr>
</thead>
<tbody>
<tr>
<td>\hbar</td>
<td>\emptyset</td>
<td>\exists</td>
<td>\Diamond</td>
<td></td>
</tr>
<tr>
<td>\imath</td>
<td>\nabla</td>
<td>\neg</td>
<td>\triangle</td>
<td></td>
</tr>
<tr>
<td>\jmath</td>
<td>\flat</td>
<td>\clubsuit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>\ell</td>
<td>\natural</td>
<td>\diamondsuit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>\wp</td>
<td>\top</td>
<td>\sharp</td>
<td>\heartsuit</td>
<td></td>
</tr>
<tr>
<td>\Re</td>
<td>\bot</td>
<td>\natural</td>
<td>\diamondsuit</td>
<td></td>
</tr>
<tr>
<td>\Im</td>
<td>\vdash</td>
<td>\angle</td>
<td>\Join</td>
<td></td>
</tr>
<tr>
<td>\mho</td>
<td>\backslash</td>
<td>\infty</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table H.17  Mathematical symbols in two sizes  (5.3.7) - p. 128.

<table>
<thead>
<tr>
<th>Command</th>
<th>\sum</th>
<th>\bigcap</th>
<th>\bigcup</th>
<th>\bigdot</th>
</tr>
</thead>
<tbody>
<tr>
<td>\int</td>
<td>\bigcup</td>
<td>\bigcup</td>
<td>\bigcup</td>
<td></td>
</tr>
<tr>
<td>\oint</td>
<td>\bigcap</td>
<td>\bigcap</td>
<td>\bigcap</td>
<td></td>
</tr>
<tr>
<td>\prod</td>
<td>\bigvee</td>
<td>\bigvee</td>
<td>\bigvee</td>
<td></td>
</tr>
<tr>
<td>\coprod</td>
<td>\bigwedge</td>
<td>\bigwedge</td>
<td>\bigwedge</td>
<td></td>
</tr>
</tbody>
</table>

Table H.18  Function names  (5.3.8) - p. 128.

<table>
<thead>
<tr>
<th>Command</th>
<th>\arccos</th>
<th>\cosh</th>
<th>\det</th>
<th>\inf</th>
<th>\limsup</th>
<th>\Pr</th>
<th>\tan</th>
</tr>
</thead>
<tbody>
<tr>
<td>\arcsin</td>
<td>\cot</td>
<td>\dim</td>
<td>\ker</td>
<td>\ln</td>
<td>\sec</td>
<td>\tanh</td>
<td></td>
</tr>
<tr>
<td>\arctan</td>
<td>\coth</td>
<td>\exp</td>
<td>\lg</td>
<td>\log</td>
<td>\sin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>\arg</td>
<td>\csc</td>
<td>\gcd</td>
<td>\lim</td>
<td>\max</td>
<td>\sinh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>\cos</td>
<td>\deg</td>
<td>\hom</td>
<td>\liminf</td>
<td>\min</td>
<td>\sup</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
H.2. Summary tables and figures

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 H.20  \( \LaTeX \) arrows
\[ \rightarrow \quad \leftarrow \quad \leftrightarrow \quad \Lleftarrow \quad \Rightarrow \quad \Leftarrow \quad \Lsh \quad \upuparrows \quad \upharpoonleft \quad \downharpoonleft \quad \multimap \quad \Rightarrow \quad \Leftrightarrow \quad \nrightarrow \quad \nleftarrow \quad \nRightarrow \quad \nLeftrightarrow \]

Negated arrows
\[ \nrightarrow \quad \nLeftarrow \quad \nRightarrow \quad \nLeftrightarrow \]

Table H.21  \( \LaTeX \) binary operation symbols
\[ \dotplus \quad \smallsetminus \quad \Cap \quad \Cap \quad \rtimes \quad \leftthreetimes \quad \forall \quad \Box \quad \Box \quad \divideontimes \quad \rtimes \quad \leftthreetimes \quad \forall \quad \forall \quad \centerdot \quad \intercal \]
Table H.22 \( \mathcal{AMS} \) Greek and Hebrew letters

\begin{verbatim}
F \digamma \kappa1 \varkappa \\
\beth \daleth \gimel
\end{verbatim}

Table H.23 \( \mathcal{AMS} \) delimiters

\begin{verbatim}
\ulcorner \urcorner \llcorner \lrcorner
\end{verbatim}

Table H.24 \( \mathcal{AMS} \) relational symbols

\begin{verbatim}
\leqq \leqslant \leqssim \lesseqgtr \\
\lessim \leqq \leqslant \leqssim \\
\lessgtr \lesseqgtr \leqq \leqslant \\
\lesseq \lesseqgtr \leqq \leqslant \\
\prec \preccurlyeq \preceq \precapprox \\
\preceq \preccurlyeq \preceq \precapprox \\
\approx \approxeq \approx \approxeq \\
\approx \approxeq \approx \approxeq \\
\equiv \equiv \equiv \equiv \\
\equiv \equiv \equiv \equiv \\
\geqq \geqslant \geqslant \geqslant \\
\geq \geqq \geqslant \geqslant \\
\ggg \gtrdot \gtrsim \gtrapprox \\
\gtrdot \gtrdot \gtrsim \gtrapprox \\
\bump \bump \bump \bump \\
\bump \bump \bump \bump \\
\vartriangleright \vartriangleright \vartriangleright \vartriangleright \\
\vartriangleright \vartriangleright \vartriangleright \vartriangleright \\
\shortmid \shortparallel \shortparallel \shortmid \\
\shortmid \shortmid \shortparallel \shortparallel \\
\backepsilon \blacktriangleleft \because \therefore \\
\backepsilon \blacktriangleleft \because \therefore \\
\blacktriangleright \blacktriangleright \blacktriangleright \blacktriangleright \\
\blacktriangleright \blacktriangleright \blacktriangleright \blacktriangleright \\
\blacktriangleleft \blacktriangleleft \blacktriangleleft \blacktriangleleft \\
\blacktriangleleft \blacktriangleleft \blacktriangleleft \blacktriangleleft \\
\because \because \because \because \\
\because \because \because \because \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \\
\therefore \therefore \therefore \therefore \
\end{verbatim}
## Table H.25  \( \mathcal{A}_M \)S negated relational symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>LaTeX Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>\lessneg</td>
<td>\nless</td>
</tr>
<tr>
<td>\leqslantneg</td>
<td>\nleqslant</td>
</tr>
<tr>
<td>\npreceqneg</td>
<td>\npreceq</td>
</tr>
<tr>
<td>\precnapproxneg</td>
<td>\precnapproxneg</td>
</tr>
<tr>
<td>\nshortmidneg</td>
<td>\nshortmid</td>
</tr>
<tr>
<td>\nvdashneg</td>
<td>\nvdash</td>
</tr>
<tr>
<td>\ntrianglerightneg</td>
<td>\ntrianglerightneg</td>
</tr>
<tr>
<td>\nsucceqneg</td>
<td>\nsucceq</td>
</tr>
<tr>
<td>\nshortparallelneg</td>
<td>\nshortparallel</td>
</tr>
<tr>
<td>\complementneg</td>
<td>\complement</td>
</tr>
<tr>
<td>\diagupneg</td>
<td>\diagup</td>
</tr>
</tbody>
</table>

## Table H.26  Miscellaneous \( \mathcal{A}_M \)S symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>LaTeX 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>\blacksquare</td>
<td>\blacksquare</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>
</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 \LaTeXe; it has been dropped from \LaTeXe.

---

**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: \textsc{\TeX} Users Group

Eijkhout V. (1992). \textit{\TeX by Topic, a \TeXnician’s Reference}. Harlow: Addison-Wesley


setting, Vol. E. Reading MA: Addison-Wesley


METAFONT. Bonn: Addison-Wesley


Addison-Wesley

\LaTeX\2ε. Reading MA: Addison-Wesley

Marchal, B. *XML by Example*. Indianapolis IN: Que-Programming


Reckdahl, K. (1996a). Using EPS graphics in \LaTeX\2ε documents, part 1: 
The graphics and graphicx packages. *TUGboat*, 17(1), 43–53

Reckdahl K. (1996b). Using EPS graphics in \LaTeX\2ε documents, part 2: 
Floating figures, boxed figures, captions, and math in figures. *TUGboat*, 
17(3), 288–310

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\2PDF—Acrobatics with 
an alternative to DVI format. *TUGboat*, 17(3), 244–251

Society

trends in scientific publication. *TUGboat*, 17(4), 367–381

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.
\= (\textbackslash accent), 24, 510
\> (tabbing), 81, 82-4, 420, 510
?`, 24, 510
@\textbackslash Bin\textbackslash {\textbackslash P}x, 312-14, 510
@\textbackslash Mak\textbackslash e\textbackslash Index, 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
@seccontformat, 457
@startsection, 456
@warning, 439
[1, 120, 124, 416, 425, 510, 549
[1, 18, 132, 426, 510
\[, 31, 32, 53, 67, 68, 81, 84, 97, 99,
104, 271, 297, 352, 419,
425, 434, 510
\[*, 31, 279, 510
\[, 120, 124, 416, 510
\[, 18, 132, 510
\], 23
\+, 17, 121, 123, 424, 511
\+ (` accent), 24, 511
\-, 23, 414, 511
\-, 17, 121, 123, 413, 424, 511
\l (tabbing), 83, 511
\l, 23
\l, 23
\l (` accent), 24, 511
\{, 17, 18, 312, 425, 511
\}, 23, 120, 132, 511
\{, 126, 132, 270, 511
\} (\textbackslash Mak\textbackslash e\textbackslash Index), 227, 511
\}, 126, 127, 132, 270, 511
\{ (table), 96
\}, 17, 18, 312, 425, 426, 428, 511
\}, 23, 120, 132, 511
\}, 17, 22, 28, 511
\ \textbackslash accent), 24, 511
10pt option, 38, 61
11pt option, 38, 61
12pt option, 38, 61, 456
8r.\textbackslash enc, 235, 236
\., 22, 28, 507
\a (tabbing), 83, 512
a4paper option, 38, 359, 447, 452, 456
a5paper option, 38
\a= (tabbing), 83, 512
\a` (tabbing), 83, 512
\AA, 24, 512
\aa, 24, 512
abbrv bibliography style, 311
abbrvnat bibliography style, 221, 311
\above\displayshortskip, 149, 512
\above\displayskip, 149, 512
Abrahams, P. W., 605
abstract, 55
abstract environment, 55, 516
\abstractname environment, 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 (\textbackslash A\textbackslash MS), 263, 512
\addButton (pdfscreen), 342
\addcontentsline, 59, 60, 174,
427, 436, 512
\adddefault (babel), 255
\addheading (babel), 255
\address, 351, 352, 353, 356, 512
\add longtime (slides), 327, 512
\addtoarlength (seminar), 332
\addtocounter, 21, 110, 115, 182,
183, 418, 513
\addtocontents, 59, 60, 174, 427,
436, 513
\addtocounter (seminar), 332
\addtoslidereset (seminar), 337
\addvspace, 185, 513
Adobe Systems Inc., 25, 231, 236, 249, 387, 504
\AE, 24, 513
\ae, 24, 513
Afrikaans, 252
afterpage package, 171, 394
\aleph, 127, 513
align environment (\AMS), 270, 274, 516
align page style (seminar), 335
alignat environment (\AMS), 270, 275, 516
aligned environment (\AMS), 276, 516
\allinethickness (eepic), 306
\allowdisplaybreaks (\AMS), 278, 513
\alltt environment, 111, 392, 517
\alltt package, 111, 392
allversions* environment (seminar), 336
\Alph, 73, 114, 183, 278, 513
\alph, 73, 114, 183, 191, 192, 513
alpha bibliography style, 311
\alpha, 125, 513
alphabetic page numbering, 45
\alsoname, 460, 513
\amalg, 126, 513
American Mathematical Society, 151, 257
American spelling, 195
amsart class, 258
amsbook class, 258
amsbsy package, 144, 192, 258–9
amscd package, 258, 263, 282
amsfndoc.tex, 257
amsfonts package, 126, 284–5, 422, 494
\AMS-\LaTeX, 257–85
\Acute, 263, 512
align environment, 270, 274, 516
alignat environment, 270, 275, 516
aligned environment, 276, 516
\allowdisplaybreaks, 278, 513
arrows, extended, 262
\Bar, 263, 515
\binom, 264, 525
binomials, 264
Bmatrix environment, 266, 517
bmatrix environment, 266, 517
\boldsymbol, 144, 258, 526
boxed, 142, 270, 526
\Breve, 263, 526
cases environment, 277, 517
CD environment, 282
\cfrac, 265, 527
\Check, 263, 528
CM fonts, 283
commutative diagrams, 282
Cyrillic fonts, 283, 497
\dbinom, 265, 531
\dddot, 263, 532
\ddddot, 263, 532
\Ddot, 263, 532
\DeclareMathOperator, 268, 282, 534
\dfrac, 264, 537
displaybreak, 278, 538
\Dot, 263, 539
dots, 263
\doteq, 263, 539
\doteqdot, 264, 539
\doteqdot, 264, 539
\doteqdot, 264, 539
\doteqdot, 264, 539
\doteqdot, 278, 541
equation* environment, 272
falign* environment, 272
\font, 283–5
fractions, 264
continued, 265
user-defined, 265
function names, 267
defining, 268
gather environment, 270, 273, 519
gathered environment, 276, 519
\genfrac, 265, 545
\Grave, 263, 546
\Hat, 263, 546
\hdotsfor, 267
\idotsint, 260, 548
\iintint, 260, 551
\iint, 260, 551
\int, 260, 551
\intertext, 259, 550
\leftroot, 269, 560
\limits, multiline, 261
special, 261
\lVert, 270, 557
\lvert, 270, 557
math symbols, 285
\mathbb, 284
matrix, 266
\matrixenvironment, 266, 519
\medspace, 269, 560
\mod, 268, 560
\mspace, 269, 560
multiline equations, 270–7
page breaks, 278
multiple integrals, 260
\multlineenvironment, 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
\overleftrightarrown, 262, 566
\overrightarrow, 262, 567
\overset, 262, 567
\pmb, 258, 570
\pod, 268, 570
\raisetag, 271
roots, 269
\rVert, 270, 576
\rvert, 270, 576
\shoveleft, 271
\shoveright, 271
\sideset, 262, 579
smallmatrixenvironment, 267
\smash, 269, 580
\splitenvironment, 270, 272, 276, 521
\subarrayenvironment, 261, 522
\subequationsenvironment, 277, 522
\substack, 261, 581
\tag, 271, 573, 583
\tbinom, 265, 583
\text, 259, 583
\tfrac, 264, 586
theorems, 280
\theoremstyle, 281
\thicksepacement, 269, 586
\thinspace, 269, 586
\Tilde, 263, 587
\underleftarrow, 262, 589
\underleftrightarrow, 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
\Vmatrixenvironment, 266, 524
\vmatrixenvironment, 266, 524
\xleftarrow, 263, 594
\xrightarrow, 263, 594
\amsldoctex, 257, 282
\amsmathpackage, 192, 258
\amsopnpackage, 258, 267-8
\amsprocclass, 258
\amssymbpackage, 284, 285
\AMS-T\LaTeX, 257
\amstexformat, 257
\amstexpackage, 258-9
\amsthmpackage, 81, 258, 280-2, 395
\and, 53, 513
\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
\arc (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 (A\_{\text{M}}S), 262
Arsenau, 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 (L\_\text{T}_{\text{E}}X 2.09), 430
\articlemag (seminar), 332
ASCII, 5, 14, 17, 228, 231, 382, 487
\askforoverwritelfalse (DocStrip), 465
\ast, 126, 514
\asymp, 126, 514
\AtBeginDocument, 444, 472, 514
\AtEndDocument, 444, 514
\AtEndOfClass, 444, 515
\AtEndOfPackage, 444, 515
\atop (\_\text{T}_{\text{E}}X), 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 L\_\text{T}_{\text{E}}X, 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 (A\_{\text{M}}S), 263, 515
Barroca, Leonor, 159
baseline, 46, 63
\baselineskip, 63, 362, 369, 515
\baselinestretch, 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 (L\_\text{T}_{\text{E}}X 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
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
\texttt{BIB}\TeX, 217, 219, 224, 309–21
customizing, 223, 321–2
type, 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
\texttt{Bib}\TeX, 15, 217, 219, 224, 309–21, 382, 388, 397, 430
writing styles for, 321
Biemesderfer, C., 605
\texttt{Big} (\TeX), 148, 525
\texttt{big} (\TeX), 148, 525
\texttt{bigcap}, 128, 525
\texttt{bigcirc}, 126, 525
\texttt{bigcup}, 128, 525
\texttt{Bigg} (\TeX), 148, 525
\texttt{bigg} (\TeX), 148, 525
\texttt{bigdot}, 128, 525
\texttt{bigplus}, 128, 525
\texttt{bigotimes}, 128, 525
\texttt{bigskip}, 33, 525
\texttt{bigskipamount}, 525
\texttt{bigscup}, 128, 525
\texttt{bigtriangleleft}, 126, 525
\texttt{bigtriangledown}, 126, 525
\texttt{biguplus}, 128, 525
\texttt{bigvee}, 128, 525
\texttt{bigwedge}, 128, 525
binary operator symbols, 125
\texttt{binom} (\AMS), 264, 525
binomial coefficient, 137, 264
blank, 11, 22
after command, 19, 22, 27, 186
at beginning of line, 22, 29
forced, 22
multiple, 11, 22
protected, 22, 29
rubber, 30
suppression of, 22
blank line for new paragraph, 22
.blg file, 397
Blue Sky Research, 382, 506
bm package, 144, 394
\texttt{bm}, 144, 394
\texttt{Bmatrix} environment (\AMS), 266, 517
\texttt{bmatrix} environment (\AMS), 266, 517
\texttt{bmod}, 129, 268, 526
body, 47
bold face in formulas, 143, 394
\texttt{boldmath}, 143, 144, 150, 151, 373, 394, 433, 526
\texttt{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
\texttt{boolean} (ifthen), 194, 195, 360, 451, 453
\texttt{bot}, 127, 526
\texttt{botfigrule}, 173, 526
bottom margin, 48
\texttt{bottombuttons (pdfscreen)}, 342
\texttt{bottomfraction}, 172, 526
\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, 305, 309, 431, 528
\circle*(eepic), 305
\circle*(picture), 295, 300, 431, 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
\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
\colm, 97, 102, 106, 529
.clo 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, 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
genral 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
PostScript version, 235, 505
conditional text, 193
config.cms, 235
config.ps, 233, 235
configuration file, local
color, 166
graphics, 164
hyphen, 253, 385
\LaTeX{} installation, 384
\ltxdoc, 397, 447, 472
pdfscreen, 343
seminar, 339, 340
configuring \LaTeX{}, 384
\cong, 126
constant, see formula
\contentsline, 530
\contentsline, 459, 530
continued fraction, 146, 149, 265
continuing dots, 123, 263
\coprod, 128, 530
copyright sign, 23, 503
\copyright, 23, 503, 530
\copyrightspace, 391
\cornersize (fancybox), 94
\cos, 128, 530
\cosh, 128, 530
\cot, 128, 530
\coth, 128, 530
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, 530
CTAN, 381, 389
\cup, 126, 530
\CurrentOption, 359, 443, 530
cyracc.def, 284
Cyrillic fonts, 6, 283, 379, 497
transliteration, 284
Czech, 252
\d (. accent), 24, 531
\dag, 23, 531
dagger, 23
\dagger, 126, 531
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, 531
database, see bibliographic
database
\date, 52, 54, 69, 353, 358, 363, 531
\datelang (non-standard), 255, 461, 531
.dat file, 253
\day (\TeX{}), 27, 460
\dbinom (\AMS{}), 265, 531
\dblfloatrule, 173, 531
\dblfloatpagefraction, 172, 531
\dblfloatsep, 172, 532
\dbltextfloatsep, 172, 532
\dbltopfraction, 172, 532
dbltopnumber, 172, 532
DC fonts, see EC fonts
dcolumn package, 107, 394
dd (ddot), 21
\ddag, 23, 532
\dagger, 126, 532
\dddot (AMS), 263, 532
\dddot (AMS), 263, 532
\ddots, 123, 532
de Boer, Ingo H., 15
DEC Multinational coding, 462
declaration, 20
local, 20
scope of, 20
\DeclareBoldMathCommand (bm), 394
\DeclareErrorFont, 376, 423, 424, 532
\DeclareFixedFont, 66, 361, 373, 533
\DeclareFontEncoding, 376, 378, 379, 533
\DeclareFontEncodingDefaults, 376, 533
\DeclareFontFamily, 376, 378, 423, 533
\DeclareFontShape, 377, 378, 423, 533
\DeclareFontSubstitution, 376, 533
\DeclareGraphicsExtensions (graphics), 165, 533
\DeclareGraphicsRule (graphics), 165, 533
\DeclareInputMath (inputenc), 461
\DeclareInputText (inputenc), 461
\DeclareMathAccent, 375, 423, 534
\DeclareMathAlphabet, 191, 374, 375, 416, 423, 534
\DeclareMathDelimiter, 375, 534
\DeclareMathOperator (AMS), 268, 282, 534
\DeclareMathRadical, 375, 534
\DeclareMathSizes, 375, 534
\DeclareMathSymbol, 375, 534
\DeclareMathVersion, 374, 422, 423, 535
\DeclareOldFontCommand, 373, 535
\DeclareOption, 359, 442, 444, 535
\DeclareOption*, 359, 443, 535
\declarepostamble (DocStrip), 465
\declarepreamble (DocStrip), 465
\DeclareRobustCommand, 445, 535
\DeclareRobustCommand*, 445, 535
\DeclareSymbolFont, 375, 423, 424, 535
\DeclareSymbolFontAlphabet, 375, 424, 536
\DeclareTextAccent, 379, 536
\DeclareTextAccentDefault, 380, 536
\DeclareTextAccentCommand, 379, 417, 536
\DeclareTextAccentDefault, 380, 536
\DeclareTextComposite, 379, 422, 536
\DeclareTextCompositeCommand, 379, 536
\DeclareTextFontCommand, 373, 536
\DeclareTextSymbol, 379, 536
\DeclareTextSymbolDefault, 380, 537
\def (TeX), 439, 445, 450
.deffile, 153, 253, 376, 378, 379, 385, 392, 398
\definecolor (color), 167, 345, 537
\deg, 128, 537
dehyphn.tex, 256
dehypht.tex, 256
delarray package, 108, 394
\DeleteShortVerb (shortvrb), 111, 393, 468, 537
\Delta, 125, 537
\Delta, 125, 537
\depth, 86, 90, 537
\DescribeEnvironment (doc), 468
\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{A}_M S), 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 (AMS), 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 (\LaTeX 2.09), 418, 422, 483
dollar sign, 23, 27
\DoNotIndex (doc), 470
\dot, 129, 263, 539
\Dot (\mathcal{A}_M S), 263, 539
\doteq, 126, 539
\dotfill, 30, 84, 86, 135, 539
\dots, 124, 539
\dots (\mathcal{A}_M S), 263, 539
\dotsb (AMS), 264, 539
\dotsi (AMS), 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
dvipsone, 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\ell, 127, 539
\ellipse (eepic), 306
ellipsis, 123
\else (\TeX{}), 451
e m dash, 23
\em, 20, 62, 65, 433, 485, 539
em, 21
emacs, 382
email addresses, 112
\emblema (pdfscreen), 341
\emph, 62, 65, 372, 540
empty page style, 42, 327, 335, 352, 452
\emptyset, 127, 540
e\TeX{} for DOS, 382
etex, graphics option, 154
en dash, 23
\EnableCrossrefs (doc), 470
capsulated 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.ldf, 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
numerate package, 74, 394
enunm counter, 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
everypage (\TeX{}), 450
\epsf file, 155
\epsf package, 159
\epsf (epsfig), 159
\epsfig package, 159
\epsfig (epsfig), 159
\epsfxsize (epsfig), 159
\epsfysize (epsfig), 159
\epsilon, 125, 541
\texttt{eqnarray} environment, 120, 138, 142, 213, 270, 420, 517
\texttt{eqnarray*} environment, 120, 138, 142, 270, 518
\texttt{eqref (\texttt{AMS})}, 278, 541
\texttt{equal (ifthen)}, 194, 363
equation, see formula
equation counter, 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
equation* environment (\texttt{AMS}), 272
equations, set of, 133–6
\texttt{equiv}, 126, 541
error messages
  basic structure for \TeX, 403
  continue program, 402
  emergency stop, 413
  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 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{errorcontextlines} counter, 404
  \texttt{errorcontextlines} \texttt{counter}, 404
  \texttt{eso-pic} package, 346
  Esperanto, 252, 462
  Esser, Thomas, 381
  Estonian, 252
\exists, 125, 541
eucal package, 284–5
eufrak package, 285\texttt{\EUR (eurosym)}, 25, 541
\texttt{\EUR (eurosym)}, 25, 28, 541\texttt{\EURofc (eurosym)}, 25, 541
\texttt{\EUR (eursym)}, 25, 28, 541
\texttt{\EURofc (eursym)}, 25, 28, 541
\texttt{\EURofc (eursym)}, 25, 28, 541
\texttt{\evensidemargin}, 48, 49, 361, 453, 541
\texttt{ex}, 21
\texttt{\ExecuteOptions}, 165, 443, 453, 541
\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
\exists, 127, 541
\exp, 128, 541
\expandafter (\TeX), 451
explicit names, changing, 459
exponents, 121
exscale package, 392
Extended Computer fonts, see EC fonts
\externaldocument (xr), 215, 243
\extracolsep, 96, 110, 455, 541
\extrarowheight (array), 107
\extraslang (non-standard), 255
\extraslideheight (seminar), 336
\f@baselineskip, 376
\f@encoding, 376
\f@family, 376
\f@series, 376
\f@shape, 376
\f@size, 376
Fairbairns, Robin, 47
falign environment (\AMS), 270, 275, 518
family, font attribute, 64, 368
\familydefault, 372
fancy page style, 43
fancybox package, 94–5, 334, 338, 345
\fancyfoot (fancyhdr), 44
fancyhdr package, 43–5, 249, 335, 454
\fancyhead (fancyhdr), 44
\fancyhf (fancyhdr), 44
\fancyheadpage (fancybox), 95, 542
\fancyheadpagestyle (fancyhdr), 45
\fancypageput (fancybox), 338
\fbox, 86, 92–4, 115, 142, 334, 542
\fbox (picture), 298
\fboxrule, 93, 94, 164, 334, 542
\fboxsep, 93, 94, 164, 298, 334, 542
\fcolorbox (color), 167, 335, 542
\fdd file, 398
\fd file, 233, 234, 283, 376, 378, 385, 398, 424, 490
\fi (\TeX), 451
figure, see float
figure counter, 181
figure caption, see float, caption
figure environment, 60, 108, 169, 173, 182, 213, 417, 419, 430, 518
\figurename, 459, 542
figures, list of, 59
file
including, 448
inputting safely, 447
listing, 447
transcript, 14, 50, 208, 211, 399, 401, 447
file transfer protocol, see FTP
file types
\.aux, 209, 210, 214, 396, 397, 418, 427, 430, 436
\.bbl, 219, 224, 310, 397
\.bib, 309, 397
\.blg, 397
\.bst, 220, 221, 310, 321, 397
\.cfg, 253, 384, 385, 397, 472
\.clo, 384, 385, 397
\.cls, 384, 385, 398, 484
\.dat, 253
\.def, 153, 253, 376, 378, 379, 385, 392, 398
\.drv, 463, 464
\.dtx, 13, 384, 389, 398, 463, 465
\.dvi, 14, 27, 153, 161, 236, 305, 393, 403, 477
\.eps, 155
\.fdd, 398
\.fd, 233, 234, 283, 376, 378, 385, 398, 424, 490
\.fmt, 398
\.gif, 476
\.glo, 230, 398, 470
\.gls, 399, 470
\.gz, 166
\.html, 476
\.idv, 478
\.idx, 226–8, 399, 470
\.ilg, 399
\.ind, 228, 399
\.ins, 389, 399, 463
\.ist, 385, 399, 470
\.jpeg, 162
\.jpg, 162
\.lfd, 253
\.lof, 59, 179, 399, 436
.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
.tiff, 162
.tif, 162
.toc, 59, 400, 436
.txt, 384
.vf, 234, 236, 400, 488, 504
.zip, 166
\file (DocStrip), 465
\filecontents environment, 432, 448, 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
cmcscl10 (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
wncy10 (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
fontsmppl 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
\footrulewidth (fancyhdr), 44
\footskip, 48, 50, 453, 544
\foraill, 127, 544
\foreignlanguage (babel), 254, 544
format file, 6, 384, 398, 440
INDEX 623

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 \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
INDEX

Gaulle, Bernard, 251
\gcd, 128, 129, 280, 545
\gdef (\TeX), 450
\ge, 126, 545
\generate (DocStrip), 465
\genfrac (\AMS), 265, 545
geometry package, 49–51, 452
\geq, 126, 545
German, 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
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.tex, 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
\hbox (\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, 365, 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
\hpagecolor (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
Icelandic, 252
\idotsint (\AMS), 260, 548
.idv file, 478
.idx file, 227, 385, 391
.idx file, 226–8, 399, 470
\if (\TeX), 451
\ifcase (\TeX), 451, 461
\iff, 127, 548
\iffalse (\TeX), 468
\IfFileExists, 447, 548
\iflanguage (babel), 254, 548
ifthen package, 193–5, 359, 392, 439, 452
ifthenelse (ifthen), 193, 195, 360, 363, 439, 451, 453
\iiiint (\AMS), 260, 551
\iiint (\AMS), 260, 551
\iint (\AMS), 260, 551
.ilg file, 399
\Im, 127, 549
\imageButton (pdfscreen), 342
\imath, 127, 129, 549
importing graphics files, 154–66
compressed, 166
configuring, 164
problems, 161
\in, 126, 549
in (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
including one file in another, 448
\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
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
Inuit, 462
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 (LaTeX 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 (acute), 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 (landscape), 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
\language (babel), 254
languages
accents and special letters, 24
babel, 252
Lappish, 462
\LARGE, 63, 325, 371, 553
\large, 63, 367, 371, 553
\lasthline (array), 107
\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
left margin, 48, 50
two-sided, 48
\left, 108, 131, 134, 137, 140, 553
\Leftarrow, 127, 554
\leftrightarrow, 127, 554
\leftarrow, 127, 554
\leftharpoonup, 127, 554
\leftmargin, 76, 77, 79, 80, 554, 604
\leftmargini, 80, 554
\leftmark, 44
\Leftarrow, 127, 554
\leq, 126, 554
\leqno option, 39, 120, 131, 271, 272, 280
\let (\TeX), 451
letter, 351–64
closing, 353
commands
forbidden, 351
local, 356
standard, 351–6
distribution list, 353
enclosures, 353
house style, 356–64

\LaTeXe, 19, 418, 553

\LaTeX \ Version 3, 8
latex209.def, 398
\LaTeX \ 2.09, 7, 9, 46, 65, 153, 159,
324, 330, 373, 374, 392,
422, 430, 440, 443, 445,
483–6
\LaTeX\ 209.def, 398
\LaTeXX2HTML translator, 476, 480
\LaTeXX Project, 3, 8
latexbug.tex, 385

\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
\LaTeX 209.def, 398
\LaTeXx2html, 476, 480
\LaTeX \ X Project, 3, 8
\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
\texttt{letter} class, 37, 351, 391, 460
\texttt{letter} environment, 352, 356, 519
\texttt{letter}.cls, 359
letterhead, 352, 356
\texttt{letterpaper} option, 38
\texttt{\lfloor}, 132, 554
\texttt{\lfoot} (fancyhdr), 44
\texttt{\lg}, 128, 554
\texttt{\lhd}, 126, 554
\texttt{\lhead} (fancyhdr), 44
ligatures, 26, 488
breaking of, 26
Cyrillic, 284
\texttt{\lim}, 128, 129, 268, 280, 554
\texttt{\liminf}, 128, 129, 280, 554
\texttt{\limits}, 123, 128, 260, 280, 555
\texttt{\limsup}, 128, 129, 280, 555
line breaking
discouraged, 32
encouraged, 31
forced, 31
with justification, 31
suppressed, 29, 32
with centered text, 67
with extra space, 31
with left justification, 67
with right justification, 67
line width, 48, 50
\texttt{\line} (eepic), 305
\texttt{\line} (picture), 293, 294, 295, 416, 431, 555
line, black, see rules
\texttt{\linebreak}, 31, 33, 434, 555
\texttt{\linethickness}, 300, 555
\texttt{\linewidth}, 47, 51, 158, 161, 332, 555
Lingnau, Anselm, 179
list environment, 74, 77–80, 230, 419, 420, 519
list labels, 69
changing style, 72
in lists, 74
list markers, see list labels
list of references, see bibliography
\texttt{\listfigurename}, 459, 555
\texttt{\listfiles}, 208, 441, 442, 447, 555
listings, 69–74
\texttt{\listof} (float), 179
\texttt{\listoffigures}, 59, 210, 399, 427, 556
\texttt{\listoftables}, 59, 210, 399, 427, 556
\texttt{\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
\texttt{\listtablename}, 459, 556
literal text, printing, 110, 393
live.pdf, 383
\texttt{\ll}, 126, 556
\texttt{\llap} (\LaTeX), 348
\texttt{\ln}, 128, 556
\texttt{\LoadClass}, 359, 398, 421, 422, 442, 443, 556
\texttt{\LoadClassWithOptions}, 442, 454, 556
\texttt{\location}, 352, 556
\texttt{.lof} file, 59, 179, 399, 436
\texttt{\log}, 128, 556
\texttt{.log} file, 14, 246, 399, 401, 403, 414, 429
\texttt{\Longleftarrow}, 127, 557
\texttt{\longleftarrow}, 127, 557
\texttt{\Longlefttrightarrow}, 127, 557
\texttt{\longleftrightarrow}, 127, 557
\texttt{\longmapsto}, 127, 557
\texttt{\Longmapsto}, 127, 557
\texttt{\longrightarrow}, 127, 557
\texttt{\longrightarrow}, 127, 557
longtable environment
(longtable), 108, 519
longtable package, 108, 395
\texttt{.lot} file, 59, 399, 436
Lower Sorbian, 252
lowering text, see text
lowering, in formulas, see formula
\texttt{\textbackslash q (\TeX )}, 557
LR box, 85, 87
\texttt{lrbox} environment, 87, 418, 519
\texttt{lscape} package, 159
\texttt{ltxdoc class}, 392, 397, 447, 471
\texttt{ltxdoc.cfg}, 472
\texttt{ltxdoc.dtx}, 472
\texttt{\textbackslash lVert (\AMS )}, 270, 557
\texttt{\textbackslash lvert (\AMS )}, 270, 557
\texttt{\textbackslash L_{\text{X}}}, 15

Macintosh, see Apple
macro environment (doc), 469
macrocode environment (doc), 469
\texttt{\textbackslash mainmatter}, 57, 210, 557
\texttt{\textbackslash makebox}, 86, 90, 364, 557
\texttt{\textbackslash makebox (picture)}, 161, 291, 292, 299, 300, 557
\texttt{makebst.tex}, 321
\texttt{\textbackslash makeglossary}, 230, 398, 416, 557
\texttt{makeidx package}, 228, 392, 399, 460
\texttt{MakeIndex}, 15, 228-9, 382, 389, 392, 399, 470, 550
\texttt{options}, 229
\texttt{style file}, 229
\texttt{\textbackslash makeindex}, 226, 228, 396, 399, 416, 557
\texttt{makeindex.tex}, 229
\texttt{\textbackslash makelabel}, 75, 77, 557
\texttt{\textbackslash makelabels}, 355, 557
\texttt{\textbackslash MakeLowercase}, 455, 557
\texttt{\textbackslash MakeShortVerb (\texttt{shortvrb})}, 111, 393, 468, 558
\texttt{\textbackslash maketitle}, 52-4, 54, 69, 419, 431, 558
\texttt{\textbackslash MakeUppercase}, 455, 457, 458, 558
\texttt{Maltese}, 462
\texttt{Malyshev, Basil}, 506
\texttt{.map file}, 234
\texttt{\textbackslash mapsto}, 127, 558
\texttt{Marchal, B.}, 606
\texttt{margin}, 48-50
\texttt{margin mark}, 116
\texttt{marginal notes}, 116-18
\texttt{\textbackslash marginpar}, 116, 117, 417, 419, 420, 430, 558
\texttt{\textbackslash marginparpush}, 118, 558
\texttt{\textbackslash marginparsep}, 50, 118, 558
\texttt{\textbackslash marginparwidth}, 50, 118, 228, 558
\texttt{\textbackslash margins (\texttt{pdfscreen})}, 342
\texttt{\textbackslash markboth}, 43, 454, 558
\texttt{\textbackslash markright}, 43, 454, 558
\texttt{makeup}, 4-5, 11, 14, 17, 197, 207
\texttt{logical}, 5, 7, 61, 112, 456, 479
typographical, 4, 28, 479
\texttt{markup languages}, 4
\texttt{math accents}, 129
\texttt{math alphabets}, 133, 258, 373
\texttt{math environment}, 119, 519
\texttt{math equation}, see formula
\texttt{math fine-tuning}, 145-8
\texttt{bracket sizing}, 148
\texttt{font size}, 146-7
\texttt{horizontal spacing}, 145, 269
\texttt{with \AMS }, 269
\texttt{math formula}, see formula
\texttt{math mode}, 119
\texttt{math standards}, 144
\texttt{math style parameters}, 148
\texttt{math symbols}, see symbols
\texttt{math variable}, see formula
\texttt{math version}, 373
\texttt{\textbackslash mathalpha}, 375
\texttt{\textbackslash mathbb (\AMS )}, 284
\texttt{\textbackslash mathbf}, 133, 143, 258, 373, 416, 422, 558
\texttt{\textbackslash mathbin}, 375
\texttt{\textbackslash mathcal}, 125, 133, 258, 284, 373, 559
\texttt{\textbackslash mathclose}, 375
\texttt{\textbackslash mathfrak}, 285
\texttt{\textbackslash mathindent}, 40, 120, 124, 142, 149, 280, 559
\texttt{\textbackslash mathit}, 133, 258, 373, 416, 559
\texttt{MathML}, 480
\texttt{\textbackslash mathnormal}, 125, 133, 373, 559
\texttt{\textbackslash mathop}, 375
\texttt{\textbackslash mathopen}, 375
\texttt{\textbackslash mathord}, 375
\texttt{\textbackslash mathpunct}, 375
\texttt{\textbackslash mathrel}, 375
\texttt{\textbackslash mathring}, 129, 559
\texttt{\textbackslash mathrm}, 123, 133, 143, 144, 258, 373, 559
\texttt{mathscr option}, 284
\textbackslash mathscr, 284
\textbackslash mathsf, 133, 144, 258, 373, 559
\textbackslash mathttf, 133, 258, 373, 559
\textbackslash mathversion, 373, 433, 559
matrix, 133–6
  small in text formulas, 138
  with \textbackslash AMS, 266
\textbackslash matrix environment (\textbackslash AMS), 266, 519
\textbackslash matrixput (epic), 302, 303
Mattes, Eberhardt, 382
\textbackslash max, 128, 129, 280, 559
\textbackslash MaxMatrixCols counter (\textbackslash AMS), 266
\textbackslash mbox, 32, 86, 87, 90, 133, 144, 150, 151, 426, 439, 559
.mbs file, 321, 322
\textbackslash mddefault, 372, 559
\textbackslash mdseries, 64, 371, 560
\textbackslash medskip, 33, 560
\textbackslash medskipamount, 560
\textbackslash medspace (\textbackslash AMS), 269, 560
merlin.mbs, 321
Merz, Thomas, 606
\textbackslash MessageBreak, 446, 447, 560
messages, error, see error messages
  processing, see
  processing messages
\textbackslash METAFONT, 6, 8, 233, 389, 488, 494, 497–9, 501
.mf file, 389, 400, 488, 498, 506
\textbackslash mho, 127, 560
\textbackslash mid, 126, 560
\textbackslash MikTeX for Windows, 381, 386
\textbackslash miktex.txt, 384
\textbackslash min, 128, 129, 280, 560
minimal class, 392
\textbackslash minipage environment, 88, 89, 90, 92, 93, 113, 142, 200, 212, 290, 292, 417, 419, 455, 520
minus sign, 23
\textbackslash minus, 22, 47, 77
\textbackslash Mittelbach, Frank, 8, 51, 178, 215, 257, 282, 367, 464, 467, 605
\textbackslash mm (millimeter), 21
\textbackslash mod (\textbackslash AMS), 268, 560
mode

\textbackslash left-to-right, 86
\textbackslash math, 119
\textbackslash paragraph, 88
\textbackslash models, 126, 560
\textbackslash monitor input/output, 211
\textbackslash month (\textbackslash TI\TeX), 27, 460
Moore, Ross, 307, 605
\textbackslash mp, 126, 560
\textbackslash mpfootnote counter, 115, 181
\textbackslash mspace (\textbackslash AMS), 269, 560
\textbackslash mu, 125, 560
multi-file text, 207
merging, 207
selective processing, 208
multicol package, 51–2, 395
multicols environment, 51, 395, 520
\textbackslash multicol* environment, 52
\textbackslash multicolcolumn, 97, 101–3, 106, 151, 417, 418, 560
multiline formula, 120, 138, 270–7
multilingual \textbackslash TI\TeX, 251–6
\textbackslash multiput (picture), 289, 290, 291, 293, 295, 428, 561
\textbackslash multiputlist (epic), 302
\textbackslash multiline environment (\textbackslash AMS), 270, 271, 520
\textbackslash multlinegap (\textbackslash AMS), 271, 561
\textbackslash myheadings page style, 42, 45, 58, 335, 452, 454
\textbackslash nabla, 127, 561
name commands, 459
\textbackslash name, 351, 352, 353, 355, 356, 358, 561
nameless environment, 20, 62, 63
\textbackslash namelimits option, 280
\textbackslash natbib package, 219–24, 476
\textbackslash natnotes.tex, 224
\textbackslash natural, 127, 561
\textbackslash nearrow, 127, 561
\textbackslash NeedsTeXFormat, 359, 421, 432, 438, 440, 454, 561
\textbackslash neg, 127, 561
negation symbols, 126
\textbackslash negmedspace (\textbackslash AMS), 269, 561
\textbackslash negthickspace (\textbackslash AMS), 269, 561
\textbackslash negthinspace (\textbackslash AMS), 269, 561
\textbackslash neg, 126, 561
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, 190, 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
\newslidewidth (seminar), 334
\newtheorem, 80, 81, 214, 395, 416, 563
\newtheorem (amsthm), 281
\newtheorem* (amsthm), 281
\newtheorempost (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
\nobreak, 364
\nocite, 310, 563
\noexpand (\TeX), 451
\noextras (\LaTeXe), 255
\nofiles, 396, 397–400, 416, 436, 563
\noindent, 46, 563
\nointlimits option, 279
\nolimits, 128, 280, 563
\nolinebreak, 33, 564
\nolimits, 128, 563
\nonameoptions, 280
\nofrenchspacing, 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
note environment (seminar), 337
note environment (slides), 326, 520
\notin, 126, 565
\notitlepage option, 39
\nocomment (seminar), 337
\nu, 125, 565
null.tex, 413
\nomerit 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
OML encoding, 491, 495
OMS encoding, 491, 495
OMX encoding, 491, 496
one-column page formatting for single pages, 51
onecolumn option, 38
oneside option, 38
\OnlyDescription (doc), 468, 472
\onlyslides (seminar), 336
\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
\originalTeX (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
\overleftrightarrows (\AMS), 262, 566
\overline, 136, 567
overlining, 136
\overrightarrow (\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
enumerate, 74, 394
epic, 302–5
epsf, 159
epsfig, 159
eso-pic, 346
eucal, 284–5
eufrak, 285
eurps, 25, 387
eurosans, 25, 387
eurosym, 25, 28
exscale, 392
fancybox, 94–5, 334, 338, 345
fancyhdr, 43–5, 249, 335, 454
flafter, 171, 392
float, 179–80
floatfig, see floatflt
floatflt, 178–9
fontenc, 378–9, 392, 501
fontsmpl, 394
french, 251
ftnright, 394
fullpage, 452, 465
graphics, 153, 155–7
graphicx, 157–9
graphpap, 301, 392
harvard, 220
helvet, 234
here, 178
hhline, 395
html, 476
hyperref, 112, 161, 220,
    239–49, 330, 339, 340,
    344
ifthen, 193–5, 359, 392, 439,
    452
indentfirst, 46, 395
inputenc, 26, 392, 461–2
latexsym, 126, 285, 392, 422,
    494
layout, 47, 395
longtable, 108, 395
lscape, 159
makeidx, 228, 392, 399, 460
multicol, 51–2, 395
natbib, 219–24, 476
newcent, 234
palatino, 234
parskip, 46–7
pause, 346–9
pdfscreen, 340–3
pp4link, 349
pstricks, 308
rotating, 159
sempdftx, 340
shortvrb, 111, 393
showidx, 228, 392
showkeys, 395
somedefs, 395
standard, 392
syntonly, 393
tlenc, 393
tabularx, 107, 395
TeX4ht, 477
texcomp, 24–5, 393, 503
theorem, 81, 282, 395
times, 234
tracefnt, 393, 433
upref, 282
url, 112
varioref, 215–16, 253, 396
verbatim, 111, 118, 396
with options, 41
xr, 215, 243, 396
xspace, 186, 396
\PackageError, 439, 446, 567
\PackageInfo, 447, 567
packages, catalogue of, 13
\PackageWarning, 439, 446, 567
\PackageWarningNoLine, 446, 567
page breaking, 33
discouraged, 33
encouraged, 33
forced, 33
    with justification, 33
two columns, 34
two-sided, 34
with tabbing, 84
with figures and tables, 34
page counter, 45, 181, 183, 190,
    195
page format, 47
diagrams, 48
full page, 452–4
one-column, 48, 51
one-sided, 38
setting values, 47
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 PPowr4, 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 (\AMSL), 266, 521
\pmb (\AMSL), 258, 570
\pmod, 129, 268, 570
PNG images, 162, 237, 238
.png file, 162
\pod (\AMSL), 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
\prefacename, 460
preload.cfg, 385
preload.ltx, 385
presentation material, see slides
\prime, 127, 571
print environment (pdfscreen), 343
\PrintChanges (doc), 470
printer driver, 15, 153, 389, 391, 398, 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
\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
\put (picture), 161, 289, 290–9, 302, 572
\putfile (epic), 302, 304
\qbezier, 299, 572
\qed (amsthm), 282
\qedsymbol (amsthm), 282
\quad, 30, 124, 269, 572
\quad, 124, 269, 273, 572
\quote environment, 61, 67, 68, 79, 521
\put (picture), 161, 289, 290–9, 302, 572
\putfile (epic), 302, 304
\qbezier, 299, 572
\quad, 30, 124, 269, 572
\quote environment, 19, 61, 67, 68, 79, 197, 521
\quotedblbase, 503, 573
\quotesinglbase, 503, 573
\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
\raisebox, 87, 88, 92, 102, 104, 362, 573
\raisetag (M¿S), 271
raising text, see text
raising, in formulas, see formula
\rangle, 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
\reffstepcounter, 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
\renewpagemstyle (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
\rhd, 126, 575
\rhead (fancyhdr), 44
\rho, 125, 575
\right, 108, 131, 134, 137, 140, 575
\Rightarrow, 74, 127, 575
\rightarrow, 127, 575
\rightharpoonup, 127, 575
\rightharpoondown, 127, 575
\rightleftharpoons, 127, 575
\rightmargin, 76, 77, 79, 575, 604
\rightmark, 44
\rlap (\TeX), 348
\roman, 73, 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, 64, 371, 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
\addtoarclength, 332
\addtoslidecontent, 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
\ptsizer, 331
\raggedslides, 334
\renewpagestyle, 335
\rotateheaderstrue, 333
\secmcm, 332
\semin, 332
\setartlength, 332
\setslidelength, 332
\slidebottommargin, 334
\slidefooterfont, 336
\slideframe, 331, 334
\slideframenet, 334
\slideframewidth, 334
\slideheadfont, 336
\slideleftmargin, 334
\slidemargin, 331
\slidetopmargin, 334
\thenote, 335, 337
\theslide, 335, 337
\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
\setslidelength (seminar), 332
\SetSymbolFont, 374, 578
\settime (slides), 327, 578
\settodepth, 184, 579
\settowidth, 184, 579
\settime (slides), 327, 578
\shadowbox (fancybox), 94, 335, 579
\shadowsize (fancybox), 94, 579
\shapedefault, 372
\sharp, 127, 579
shortand.tex, 322
\shortstack, 297, 298, 299, 579
shortvrb package, 111, 393
\shoveleft (\AMS{}), 271
\shoveright (\AMS{}), 271
\showidx package, 228, 392
\showkeys package, 395
\sideset (\AMS{}), 262, 579
\sideways environment (rotating), 159
\\Sigma, 125, 579
\\sigma, 125, 579
\signature, 351, 352, 353, 358, 579
\sim, 126, 580
\sin, 126, 580
\sineq, 126, 580
Simonic, Aleksander, 15
\sin, 128, 580
\sinh, 128, 580
size, font attribute, 64, 368
\\size, 372
\slidesmag, 334
\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
slide environment (slides), 325, 521
slide* environment (seminar), 331, 332
\slidebottommargin (seminar), 334
\slidefootfont (seminar), 336
\slideframe (seminar), 331, 334
\slideframesep (seminar), 334
\slideheadfont (seminar), 336
\slideleftmargin (seminar), 334
\slidedapagestyle (seminar), 336
\sliderightmargin (seminar), 334
\sliderightmargin (seminar), 334
slides, 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
\sloppy, 36, 88, 434, 580
\sloppy par 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, 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
\ss, 24, 581
\ssf @ size, 376
\sstretch, 184, 581
\ssubset, 126, 581
\ssubseteq, 126, 581
\ssupset, 126, 581
\ssupseteq, 126, 581
square root sign, 122
\SS, 24, 576
\ss, 24, 581
\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
\sty file, 9, 12, 41, 384, 385, 463, 484
style files (\TEX 2.09), 483
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, 226, 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
\subset, 126, 582
\supset, 126, 582
\substack (\AMS), 261, 581
\subsubsection, 55, 582
\ssucc, 126, 582
\ssuccseq, 126, 582
\sum, 123, 128, 137, 582
\sumlimits option, 279
summation sign, 123
\sup, 128, 129, 280, 582
superscript, 121
\suppressfloats, 126, 582
\setcounter{equation}{0}
\textbackslash subequations environment
(\AMS), 277, 522
\textbackslash subitem, 225, 226, 581
\textbackslash subparagraph, 55, 581
\textbackslash subparagraph*, 55, 581
subscript, 121
subsection counter, 57, 181
\subsection, 44, 55, 181, 581
\subsection*, 55, 581
\subset, 126, 582
\supset, 126, 582
\substack (\AMS), 261, 581
\textbackslash subsubsection, 55, 582
\textbackslash subsubsection*, 55, 582
\textbackslash succ, 126, 582
\textbackslash succseq, 126, 582
\textbackslash sum, 123, 128, 137, 582
\textbackslash sumlimits option, 279
summation sign, 123
\textbackslash sup, 128, 129, 280, 582
superscript, 121
\textbackslash suppressfloats, 171, 175, 582
\textbackslash supset, 126, 582
\textbackslash supseteq, 126, 582
\textbackslash surd, 127, 582
Sutor, Robert, 605
\textbackslash swapnumbers (amsthm), 281
\textbackslash swarrow, 127, 582
Swedish, 252
\textbackslash switch \textbackslash def (babel), 253
\symbol, 66, 88, 582
symbols
above one another, 137, 261
alternative commands, 462
arrows, 127, 262, 282
binary operators, 125
brackets, 131
automatic size, 131
invisible, 132
manual sizing, 148
multiline equations, 140, 148, 276
direct input, 26, 461
integral, 123, 260
keyboard, 120
letters
calligraphic, 125, 284
Greek, 125
limits, 128, 261
miscellaneous, 127
negation, 126
pointers, 127
relational, 126
summation, 123
two sizes, 128
\textbackslash syntaxonly (syntonly), 393
\textbackslash syntonly package, 393
T1 encoding, 500, 501, 503, 506
t1enc package, 393
t1enc.def, 398, 501
\textbackslash t (~ accent), 24, 582
tab stop, 81
tabbing, 81-5
accents in, 83
additional commands, 83
backwards, 82
end of line, 81
left margin, 82
setting, 82
nesting, 84
page breaking, 84
sample line, 82
setting stops, 81
storing, 83
tabbing environment, 81, 83, 84, 415, 419, 420, 428, 522
\textbackslash tabbingsep, 83, 582
\textbackslash tabcolsep, 98, 106, 583
table, 95–110
column alignment, 101
on decimals, 107
construction, 95-8
\textbackslash @-expression, 96, 101, 110
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
  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
\bigg, 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
\.tex 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
\textcomp, 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
\textsf, 25, 65, 372, 585
\textsl, 44, 65, 372, 585
\textsuperscript, 137, 146, 264, 375, 585
\textstyle, 137, 146, 264, 375, 585
\textsuperscript, 463, 585
\texttrademark, 463, 585
\texttt{65, 88, 372, 585}
\texttt{65, 372, 585}
Textures, 382
\texttt{21, 48, 50, 184, 195, 332, 361, 452, 453, 455, 586}
\texttt{376}
\texttt{234, 400, 487, 498, 503, 506}
\texttt{264, 586}
\texttt{TH, 503, 586}
\texttt{th, 503, 586}
Thành, Hán Thé, 16, 238, 606
\texttt{54, 424, 586}
\texttt{586}
\texttt{183, 586}
\texttt{183, 586}
\texttt{217, 220, 221, 224, 228, 309, 311, 419, 523}
\texttt{187}
\texttt{191, 277}
\texttt{113, 116, 190}
Théling, Henrik, 25
\texttt{225, 228, 399, 523}
\texttt{335, 337}
\texttt{80, 523}
\texttt{81, 282, 395}
\texttt{80}
\texttt{80}
\texttt{280}
\texttt{281}
\texttt{44, 183, 335, 363, 455}
\texttt{183, 187, 277}
\texttt{335, 337}
\texttt{187}
\texttt{125, 586}
\texttt{125, 586}
\texttt{306}
\texttt{94, 289, 293, 300, 497, 586}
\texttt{269, 586}
\texttt{269, 586}
\texttt{95, 586}
\texttt{21, 43, 327, 335, 363, 587}
\texttt{281}
\texttt{162}
\texttt{238}
\texttt{162}
\texttt{129, 263, 587}
\texttt{263, 587}
\texttt{234}
\texttt{126, 587}
\texttt{379, 504}
\texttt{378}
\texttt{63, 371, 587}
title
of sections, 55
of table, see float
title page, 39, 52
free format, 54
standard format, 52
\texttt{52, 53, 54, 69, 419, 431, 587}
titlepage environment, 52, 54, 69, 210, 523
titlepage option, 39, 54, 55, 351
\texttt{127, 587}
toaddress, 359, 363
tocdepth counter, 58
.toc file, 59, 400, 436
\texttt{27, 28, 36, 255, 363, 460, 587}
toname, 359, 360, 363
top margin, 48, 50
\texttt{127, 587}
\texttt{173, 587}
\texttt{172, 587}
\texttt{171, 588}
\texttt{149, 588, 604}
\texttt{48, 588}
\texttt{349}
\texttt{86, 90, 588}
\texttt{172, 588}
\texttt{393, 433}
\texttt{14, 50, 399, 401}
\texttt{284}
\texttt{127, 588}
\texttt{126, 588}
\texttt{126, 588}
trivial lists, 79
\texttt{79, 523}
truetex, graphics option, 154
TrueType fonts, 238
TS1 encoding, 502, 506
\tt (\LaTeX 2.09), 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 \hbox warning, 434
Underfull \vbox 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
\usrt 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
\uplus, 126, 590
Upper Sorbian, 252
upref package, 282
\uroot (\AMS), 269, 590
\upshape, 64, 371, 590
\Upsilon, 125, 590
\upsilon, 125, 590
Urban, Michael, 606
\url, 112
url (url), 112
\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
\usepackage, 12, 24, 41, 42, 201, 222, 395, 412, 416, 418, 421, 422, 432, 441, 591
\usepreamble (DocStrip), 465
\usepostamble (DocStrip), 465
ushyphen.tex, 256
\v (ˇ accent), 24, 591
\varepsilon, 125, 591
\varinjlim (\AMS), 268, 591
\varliminf (\AMS), 268, 591
\varlimsup (\AMS), 268, 591
\varphi, 125, 591
\varpi, 125, 591
\varprojlim (\AMS), 268, 591
\varrho, 125, 591
\varsigma, 125, 591
\vartheta, 125, 591
\bbox (\TeX{}), 439
\vdash, 126, 127, 592
\vdots, 123, 592
\vec, 129, 144, 263, 592
\Vec (\textit{AMS}), 263, 592
\vector (\textit{picture}), 161, 295, 299, 300, 416, 592
\vee, 126, 592
\verb, 110, 393, 420, 592
\verb*, 110, 592
verbatim environment, 110, 467, 523
in verbatim package, 111, 396
verbatim package, 111, 118, 396
verbatim* environment, 110, 523
verbatiminput (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 (\textit{AMS}), 266, 524
vmatrix environment (\textit{AMS}), 266, 524
\voffset (\TeX{}), 593, 604
Volovich, Vladimir, 506
\vpagecolor (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
\vtex, graphics option, 154
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
Wooding, 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 (\textit{AMS}), 263, 594
XML, 3, 5, 10, 478-81
\xmltex format, 481
\xmltex.tex, 480
\xr package, 215, 243, 396
\xrightarrow (\textit{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