LaTeX2e: An unofficial reference manual

March 2018

http://puszcza.gnu.org.ua/software/latexrefman/

This document is an unofficial reference manual for LaTeX, a document preparation system, version of March 2018.

This manual was originally translated from `LATEX.HLP` v1.0a in the VMS Help Library. The pre-translation version was written by George D. Greenwade of Sam Houston State University. The LaTeX 2.09 version was written by Stephen Gilmore. The LaTeX2e version was adapted from this by Torsten Martinsen. Karl Berry made further updates and additions, and gratefully acknowledges using *Hypertext Help with LaTeX*, by Sheldon Green, and *LaTeX Command Summary* (for LaTeX 2.09) by L. Botway and C. Biemesderfer (published by the TeX Users Group as *TeXniques* number 10), as reference material (no text was directly copied).

Copyright 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018 Karl Berry.
Copyright 1988, 1994, 2007 Stephen Gilmore.
Copyright 1994, 1995, 1996 Torsten Martinsen.

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions.

Short Contents

LaTeX2e: An unofficial reference manual 1
1 About this document 2
2 Overview of LaTeX 3
3 Document classes 8
4 Fonts 17
5 Layout 23
6 Sectioning 31
7 Cross references 35
8 Environments 37
9 Line breaking 71
10 Page breaking 73
11 Footnotes 74
12 Definitions 78
13 Counters 87
14 Lengths 90
15 Making paragraphs 92
16 Math formulas 94
17 Modes 110
18 Page styles 112
19 Spaces 114
20 Boxes 119
21 Color 122
22 Graphics 126
23 Special insertions 137
24 Splitting the input 145
25 Front/back matter 146
26 Letters 148
27 Terminal input/output 152
28 Command line 153
A Document templates 154
Concept Index 157
Command Index 164

Table of Contents

LaTeX2e: An unofficial reference manual 1

1 About this document 2

2 Overview of LaTeX 3

2.1 Starting and ending 3
2.2 Output files 3
2.3 TeX engines 4
2.4 LaTeX command syntax 5
2.4.1 Environments 5
2.4.2 Command declarations 6
2.4.3 `\makeatletter` and `\makeatother` 6
2.4.3.1 `\@ifstar` 7

3 Document classes 8

3.1 Document class options 8
3.2 Additional packages 9
3.3 Class and package construction 10
3.3.1 Class and package structure 10
3.3.2 Class and package commands 11

4 Fonts 17

4.1 Font styles 17
4.2 Font sizes 19
4.3 Low-level font commands 19

5 Layout 23

5.1 `\onecolumn` 23
5.2 `\twocolumn` 23
5.3 `\flushbottom` 25
5.4 `\raggedbottom` 25
5.5 Page layout parameters 25
5.6 Floats 27

6 Sectioning 31

6.1 `\@startsection` 31

7 Cross references 35

7.1 `\label` 35
7.2 `\pageref{`*key*`}` 36
7.3 `\ref{`*key*`}` 36

8 Environments ... 37

8.1 abstract ... 37
8.2 array ... 38
8.3 center ... 39
8.3.1 \centering ... 39
8.4 description ... 40
8.5 displaymath ... 41
8.6 document ... 41
8.6.1 \AtBeginDocument ... 41
8.6.2 \AtEndDocument ... 42
8.7 enumerate ... 42
8.8 eqnarray ... 43
8.9 equation ... 44
8.10 figure ... 44
8.11 filecontents: Write an external file ... 45
8.12 flushleft ... 46
8.12.1 \raggedright ... 46
8.13 flushright ... 46
8.13.1 \raggedleft ... 46
8.14 itemize ... 47
8.15 letter environment: writing letters ... 48
8.16 list ... 48
8.16.1 \item: An entry in a list ... 53
8.16.2 trivlist: A restricted form of list ... 53
8.17 math ... 54
8.18 minipage ... 54
8.19 picture ... 54
8.19.1 \circle ... 55
8.19.2 \makebox ... 56
8.19.3 \framebox ... 56
8.19.4 \dashbox ... 56
8.19.5 \frame ... 56
8.19.6 \line ... 56
8.19.7 \linethickness ... 57
8.19.8 \thicklines ... 57
8.19.9 \thinlines ... 57
8.19.10 \multiput ... 57
8.19.11 \oval ... 57
8.19.12 \put ... 57
8.19.13 \shortstack ... 58
8.19.14 \vector ... 58
8.20 quotation and quote ... 58
8.21 tabbing ... 59
8.22 table ... 61
8.23 tabular ... 62
8.23.1 \multicolumn ... 65
8.23.2 \vline ... 66
8.23.3 \cline ... 66

8.23.4 \hline ... 67
8.24 thebibliography ... 67
8.24.1 \bibitem ... 67
8.24.2 \cite ... 68
8.24.3 \nocite ... 68
8.24.4 Using BibTEX ... 68
8.25 theorem ... 69
8.26 titlepage ... 69
8.27 verbatim ... 69
8.27.1 \verb ... 70
8.28 verse ... 70

9 Line breaking ... 71
9.1 \\ ... 71
9.2 \obeycr & \restorecr ... 71
9.3 \newline ... 71
9.4 \- (discretionary hyphen) ... 72
9.5 \discretionary (generalized hyphenation point) ... 72
9.6 \fussy ... 72
9.7 \sloppy ... 72
9.8 \hyphenation ... 72
9.9 \linebreak & \nolinebreak ... 72

10 Page breaking ... 73
10.1 \cleardoublepage ... 73
10.2 \clearpage ... 73
10.3 \newpage ... 73
10.4 \enlargethispage ... 73
10.5 \pagebreak & \nopagebreak ... 73

11 Footnotes ... 74
11.1 \footnote ... 74
11.2 \footnotemark ... 75
11.3 \footnotetext ... 75
11.4 Footnotes in a table ... 75
11.5 Footnotes in section headings ... 76
11.6 Footnotes of footnotes ... 76
11.7 Multiple references to footnotes ... 77
11.8 Footnote parameters ... 77

12 Definitions ... 78
12.1 \newcommand & \renewcommand ... 78
12.2 \providecommand ... 79
12.3 \newcounter: Allocating a counter ... 79
12.4 \newlength: Allocating a length ... 80
12.5 \newsavebox: Allocating a box ... 80
12.6 \newenvironment & \renewenvironment ... 80
12.7 \newtheorem ... 82
12.8 \newfont: Define a new font (obsolete) ... 84
12.9 \protect ... 84
12.10 \ignorespaces & \ignorespacesafterend ... 85

13 Counters ... 87
13.1 \alph \Alph \arabic \roman \Roman \fnsymbol: Printing counters ... 87
13.2 \usecounter{*counter*} ... 88
13.3 \value{*counter*} ... 88
13.4 \setcounter{*counter*}{*value*} ... 89
13.5 \addtocounter{*counter*}{*value*} ... 89
13.6 \refstepcounter{*counter*} ... 89
13.7 \stepcounter{*counter*} ... 89
13.8 \day \month \year: Predefined counters ... 89

14 Lengths ... 90
14.1 Units of length ... 90
14.2 \setlength ... 91
14.3 \addtolength ... 91
14.4 \settodepth ... 91
14.5 \settoheight ... 91
14.6 \settowidth{*len*}{*text*} ... 91
14.7 Predefined lengths ... 91

15 Making paragraphs ... 92
15.1 \indent ... 92
15.2 \noindent ... 92
15.3 \parskip ... 92
15.4 Marginal notes ... 92

16 Math formulas ... 94
16.1 Subscripts & superscripts ... 94
16.2 Math symbols ... 95
16.3 Math functions ... 105
16.4 Math accents ... 106
16.5 Spacing in math mode ... 107
16.6 Math miscellany ... 107

17 Modes ... 110
17.1 \ensuremath ... 110

18 Page styles ... 112
18.1 \maketitle ... 112
18.2 \pagenumbering ... 112
18.3 \pagestyle ... 112
18.4 \thispagestyle{*style*} ... 113

19 Spaces ... 114
19.1 \hspace ... 114
19.2 \hfill ... 114
19.3 \spacefactor ... 114
19.3.1 \(SPACE) and \@ ... 115
19.3.2 \frenchspacing ... 115
19.3.3 \normalsfcodes ... 116
19.4 \ after control sequence ... 116
19.5 \thinspace: Insert 1/6 em ... 116
19.6 \/: Insert italic correction ... 116
19.7 \hrulefill \dotfill ... 116
19.8 \addvspace ... 117
19.9 \bigskip \medskip \smallskip ... 117
19.10 \vfill ... 117
19.11 \vspace{*length*} ... 118

20 Boxes ... 119
20.1 \mbox{*text*} ... 119
20.2 \fbox and \framebox ... 119
20.3 lrbox ... 119
20.4 \makebox ... 119
20.5 \parbox ... 120
20.6 \raisebox ... 120
20.7 \savebox ... 121
20.8 \sbox{*boxcmd*}{*text*} ... 121
20.9 \usebox{*boxcmd*} ... 121

21 Color ... 122
21.1 Color package options ... 122
21.2 Color models ... 122
21.3 Commands for color ... 123
21.3.1 Define colors ... 123
21.3.2 Colored text ... 123
21.3.3 Colored boxes ... 124
21.3.4 Colored pages ... 125

22 Graphics ... 126
22.1 Graphics package options ... 126
22.2 Graphics package configuration ... 127
22.2.1 `\graphicspath` ... 127
22.2.2 `\DeclareGraphicsExtensions` ... 128
22.2.3 `\DeclareGraphicsRule` ... 129
22.3 Commands for graphics ... 130
22.3.1 `\includegraphics` ... 130
22.3.2 `\rotatebox` ... 135
22.3.3 `\scalebox` ... 136
22.3.4 `\resizebox` ... 136

23 Special insertions ... 137
23.1 Reserved characters ... 137
23.2 Upper and lower case ... 137
23.3 Symbols by font position ... 138
23.4 Text symbols ... 138
23.5 Accents ... 141
23.6 Additional Latin letters ... 142
23.7 `\rule` ... 143
23.8 `\today` ... 143

24 Splitting the input ... 145
24.1 `\include` ... 145
24.2 `\includeonly` ... 145
24.3 `\input` ... 145

25 Front/back matter ... 146
25.1 Tables of contents ... 146
25.1.1 `\addcontentsline` ... 146
25.1.2 `\addtocontents` ... 146
25.2 Glossaries ... 147
25.3 Indexes ... 147

26 Letters ... 148
26.1 `\address` ... 149
26.2 `\cc` ... 149
26.3 `\closing` ... 150
26.4 `\encl` ... 150
26.5 `\location` ... 150
26.6 `\makelabels` ... 150
26.7 `\name` ... 150
26.8 `\opening` ... 151
26.9 `\ps` ... 151
26.10 `\signature` ... 151
26.11 `\telephone` ... 151

27 Terminal input/output 152
27.1 `\typein[`*cmd*`]{`*msg*`}` 152
27.2 `\typeout{`*msg*`}` 152

28 Command line 153

Appendix A Document templates 154
A.1 `beamer` template 154
A.2 `book` template 154
A.3 `tugboat` template 155

Concept Index 157

Command Index 164

LaTeX2e: An unofficial reference manual

This document is an unofficial reference manual (version of March 2018) for LaTeX2e, a document preparation system.

1 About this document

This is an unofficial reference manual for the LaTeX2e document preparation system, which is a macro package for the TeX typesetting program (see Chapter 2 [Overview], page 3). This document's home page is `http://puszcza.gnu.org.ua/software/latexrefman/`. That page has links to the current output in various formats, sources, mailing list archives and subscriptions, and other infrastructure.

In this document, we will mostly just use 'LaTeX' rather than 'LaTeX2e', since the previous version of LaTeX (2.09) was frozen decades ago.

LaTeX is currently maintained by a group of volunteers (`http://latex-project.org`). The official documentation written by the LaTeX project is available from their web site. This document is completely unofficial and has not been reviewed by the LaTeX maintainers. Do not send bug reports or anything else about this document to them. Instead, please send all comments to `latexrefman@tug.org`.

This document is a reference. There is a vast array of other sources of information about LaTeX, at all levels. Here are a few introductions.

`http://ctan.org/pkg/latex-doc-ptr`
: Two pages of recommended references to LaTeX documentation.

`http://ctan.org/pkg/first-latex-doc`
: Writing your first document, with a bit of both text and math.

`http://ctan.org/pkg/usrguide`
: The guide for document authors that is maintained as part of LaTeX. Many other guides by many other people are also available, independent of LaTeX itself; one such is the next item:

`http://ctan.org/pkg/lshort`
: A short introduction to LaTeX, translated to many languages.

`http://tug.org/begin.html`
: Introduction to the TeX system, including LaTeX, with further references.

2 Overview of LaTeX

LaTeX is a system for typesetting documents. It was originally created by Leslie Lamport and is now maintained by a group of volunteers (`http://latex-project.org`). It is widely used, particularly for complex and technical documents, such as those involving mathematics.

A LaTeX user writes an input file containing text along with interspersed commands, for instance commands describing how the text should be formatted. It is implemented as a set of related commands that interface with Donald E. Knuth's TeX typesetting program (the technical term is that LaTeX is a *macro package* for the TeX engine). The user produces the output document by giving that input file to the TeX engine.

The term LaTeX is also sometimes used to mean the language in which the document is marked up, that is, to mean the set of commands available to a LaTeX user.

The name LaTeX is short for "Lamport TeX". It is pronounced LAH-teck or LAY-teck, or sometimes LAY-tecks. Inside a document, produce the logo with `\LaTeX`. Where use of the logo is not sensible, such as in plain text, write it as '`LaTeX`'.

2.1 Starting and ending

LaTeX files have a simple global structure, with a standard beginning and ending. Here is a "hello, world" example:

```
\documentclass{article}
\begin{document}
Hello, \LaTeX\ world.
\end{document}
```

Here, the '`article`' is the so-called *document class*, implemented in a file `article.cls`. Any document class can be used. A few document classes are defined by LaTeX itself, and vast array of others are widely available. See Chapter 3 [Document classes], page 8.

You can include other LaTeX commands between the `\documentclass` and the `\begin{document}` commands. This area is called the *preamble*.

The `\begin{document} ... \end{document}` is a so-called *environment*; the '`document`' environment (and no others) is required in all LaTeX documents (see Section 8.6 [document], page 41). LaTeX provides many environments itself, and many more are defined separately. See Chapter 8 [Environments], page 37.

The following sections discuss how to produce PDF or other output from a LaTeX input file.

2.2 Output files

LaTeX produces a main output file and at least two accessory files. The main output file's name ends in either `.dvi` or `.pdf`.

`.dvi` If LaTeX is invoked with the system command `latex` then it produces a DeVice Independent file, with extension `.dvi`. You can view this file with a command such as `xdvi`, or convert it to a PostScript `.ps` file with `dvips` or to a Portable Document Format `.pdf` file with `dvipdfmx`. The contents of the file can be

dumped in human-readable form with `dvitype`. A vast array of other DVI utility programs are available (`http://mirror.ctan.org/dviware`).

`.pdf` If LaTeX is invoked via the system command `pdflatex`, among other commands (see Section 2.3 [TeX engines], page 4), then the main output is a Portable Document Format (PDF) file. Typically this is a self-contained file, with all fonts and images included.

LaTeX also produces at least two additional files.

`.log` This transcript file contains summary information such as a list of loaded packages. It also includes diagnostic messages and perhaps additional information for any errors.

`.aux` Auxiliary information is used by LaTeX for things such as cross references. For example, the first time that LaTeX finds a forward reference—a cross reference to something that has not yet appeared in the source—it will appear in the output as a doubled question mark `??`. When the referred-to spot does eventually appear in the source then LaTeX writes its location information to this `.aux` file. On the next invocation, LaTeX reads the location information from this file and uses it to resolve the reference, replacing the double question mark with the remembered location.

LaTeX may produce yet more files, characterized by the filename ending. These include a `.lof` file that is used to make a list of figures, a `.lot` file used to make a list of tables, and a `.toc` file used to make a table of contents. A particular class may create others; the list is open-ended.

2.3 TeX engines

LaTeX is defined to be a set of commands that are run by a TeX implementation (see Chapter 2 [Overview], page 3). This section gives a terse overview of the main programs.

`latex`

`pdflatex` In TeX Live (`http://tug.org/texlive`), if LaTeX is invoked via either the system command `latex` or `pdflatex`, then the pdfTeX engine is run (`http://ctan.org/pkg/pdftex`). When invoked as `latex`, the main output is a `.dvi` file; as `pdflatex`, the main output is a `.pdf` file.

pdfTeX incorporates the e-TeX extensions to Knuth's original program (`http://ctan.org/pkg/etex`), including additional programming features and bi-directional typesetting, and has plenty of extensions of its own. e-TeX is available on its own as the system command `etex`, but this is plain TeX (and produces `.dvi`).

In other TeX distributions, `latex` may invoke e-TeX rather than pdfTeX. In any case, the e-TeX extensions can be assumed to be available in LaTeX.

`lualatex` If LaTeX is invoked via the system command `lualatex`, the LuaTeX engine is run (`http://ctan.org/pkg/luatex`). This program allows code written in the scripting language Lua (`http://luatex.org`) to interact with TeX's typesetting. LuaTeX handles UTF-8 Unicode input natively, can handle OpenType and TrueType fonts, and produces a `.pdf` file by default. There is also `dvilualatex` to produce a `.dvi` file, but this is rarely used.

`xelatex` If LaTeX is invoked with the system command `xelatex`, the XeTeX engine is run (`http://tug.org/xetex`). Like LuaTeX, XeTeX natively supports UTF-8 Unicode and TrueType and OpenType fonts, though the implementation is completely different, mainly using external libraries instead of internal code. XeTeX produces a `.pdf` file as output; it does not support DVI output.

Internally, XeTeX creates an `.xdv` file, a variant of DVI, and translates that to PDF using the (`x`)`dvipdfmx` program, but this process is automatic. The `.xdv` file is only useful for debugging.

Other variants of LaTeX and TeX exist, e.g., to provide additional support for Japanese and other languages ([u]pTeX, `http://ctan.org/pkg/ptex`, `http://ctan.org/pkg/uptex`).

2.4 LaTeX command syntax

In the LaTeX input file, a command name starts with a backslash character, `\`. The name itself then consists of either (a) a string of letters or (b) a single non-letter.

LaTeX commands names are case sensitive so that `\pagebreak` differs from `\Pagebreak` (the latter is not a standard command). Most commands are lowercase, but in any event you must enter all commands in the same case as they are defined.

A command may be followed by zero, one, or more arguments. These arguments may be either required or optional. Required arguments are contained in curly braces, `{...}`. Optional arguments are contained in square brackets, `[...]`. Generally, but not universally, if the command accepts an optional argument, it comes first, before any required arguments.

Inside of an optional argument, to use the character close square bracket (`]`) hide it inside curly braces, as in `\item[closing bracket {]}]`. Similarly, if an optional argument comes last, with no required argument after it, then to make the first character of the following text be an open square bracket, hide it inside curly braces.

LaTeX has the convention that some commands have a `*` form that is related to the form without a `*`, such as `\chapter` and `\chapter*`. The exact difference in behavior varies from command to command.

This manual describes all accepted options and `*`-forms for the commands it covers (barring unintentional omissions, a.k.a. bugs).

2.4.1 Environments

Synopsis:

```
\begin{environment name}
  ...
\end{environment name}
```

An area of LaTeX source, inside of which there is a distinct behavior. For instance, for poetry in LaTeX put the lines between `\begin{verse}` and `\end{verse}`.

```
\begin{verse}
  There once was a man from Nantucket \\
  ...
\end{verse}
```

See Chapter 8 [Environments], page 37, for a list of environments.

The *environment name* at the beginning must exactly match that at the end. This includes the case where *environment name* ends in a star (`*`); both the `\begin` and `\end` texts must include the star.

Environments may have arguments, including optional arguments. This example produces a table. The first argument is optional (and causes the table to be aligned on its top row) while the second argument is required (it specifies the formatting of columns).

```
\begin{tabular}[t]{r|l}
  ... rows of table ...
\end{tabular}
```

2.4.2 Command declarations

A command that changes the value, or changes the meaning, of some other command or parameter. For instance, the `\mainmatter` command changes the setting of page numbers from roman numerals to arabic.

2.4.3 `\makeatletter` and `\makeatother`

Synopsis:

```
\makeatletter
  ... definition of commands with @ in their name ..
\makeatother
```

Used to redefine internal LaTeX commands. `\makeatletter` makes the at-sign character `@` have the category code of a letter, 11. `\makeatother` sets the category code of `@` to 12, its original value.

As each character is read by TeX for LaTeX, it is assigned a character category code, or *catcode* for short. For instance, the backslash `\` is assigned the catcode 0, for characters that start a command. These two commands alter the catcode assigned to `@`.

The alteration is needed because many of LaTeX's commands use `@` in their name, to prevent users from accidentally defining a command that replaces one of LaTeX's own. Command names consist of a category 0 character, ordinarily backslash, followed by letters, category 11 characters (except that a command name can also consist of a category 0 character followed by a single non-letter symbol). So under the default category codes, user-defined commands cannot contain an `@`. But `\makeatletter` and `\makeatother` allow users to define or redefine commands named with `@`.

Use these two commands inside a `.tex` file, in the preamble, when defining or redefining a command with `@` in its name. Don't use them inside `.sty` or `.cls` files since the `\usepackage` and `\documentclass` commands set the at sign to have the character code of a letter.

For a comprehensive list of macros with an at-sign in their names see `http://ctan.org/pkg/macros2e`. These macros are mainly intended to package or class authors.

The example below is typical. In the user's class file is a command `\thesis@universityname`. The user wants to change the definition. These three lines should go in the preamble, before the `\begin{document}`.

```
\makeatletter
\renewcommand{\thesis@universityname}{Saint Michael's College}
\makeatother
```

2.4.3.1 \@ifstar

Synopsis:

```
\newcommand{\mycmd}{\@ifstar{\mycmd@star}{\mycmd@nostar}}
\newcommand{\mycmd@nostar}[non-starred command number of args]{body of non-starred command}
\newcommand{\mycmd@star}[starred command number of args]{body of starred command}
```

Many standard LaTeX environments or commands have a variant with the same name but ending with a star character `*`, an asterisk. Examples are the `table` and `table*` environments and the `\section` and `\section*` commands.

When defining environments, following this pattern is straightforward because `\newenvironment` and `\renewenvironment` allow the environment name to contain a star. For commands the situation is more complex. As in the synopsis above, there will be a user-called command, given above as `\mycmd`, which peeks ahead to see if it is followed by a star. For instance, LaTeX does not really have a `\section*` command; instead, the `\section` command peeks ahead. This command does not accept arguments but instead expands to one of two commands that do accept arguments. In the synopsis these two are `\mycmd@nostar` and `\mycmd@star`. They could take the same number of arguments or a different number, or no arguments at all. As always, in a LaTeX document a command using at-sign `@` must be enclosed inside a `\makeatletter ... \makeatother` block (see Section 2.4.3 [\makeatletter and \makeatother], page 6).

This example of `\@ifstar` defines the command `\ciel` and a variant `\ciel*`. Both have one required argument. A call to `\ciel{night}` will return "starry night sky" while `\ciel*{blue}` will return "starry not blue sky".

```
\newcommand*{\ciel@unstarred}[1]{starry #1 sky}
\newcommand*{\ciel@starred}[1]{starry not #1 sky}
\newcommand*{\ciel}{\@ifstar{\ciel@starred}{\ciel@unstarred}}
```

In the next example, the starred variant takes a different number of arguments than does the unstarred one. With this definition, Agent 007's ``My name is \agentsecret*{Bond}, \agentsecret{James}{Bond}.'' is equivalent to ``My name is \textsc{Bond}, \textit{James} textsc{Bond}.''

```
\newcommand*{\agentsecret@unstarred}[2]{\textit{#1} \textsc{#2}}
\newcommand*{\agentsecret@starred}[1]{\textsc{#1}}
\newcommand*{\agentsecret}{\@ifstar{\agentsecret@starred}{\agentsecret@unstarred}}
```

There are two sometimes more convenient ways to accomplish the work of `\@ifstar`. The `suffix` package allows the construct `\newcommand\mycommand{`*unstarred version*`}` followed by `\WithSuffix\newcommand\mycommand*{`*starred version*`}`. And LaTeX3 has the `xparse` package that allows this code.

```
\NewDocumentCommand\foo{s}{\IfBooleanTF#1
  {starred version}%
  {unstarred version}%
  }
```

3 Document classes

The document's overall class is defined with this command, which is normally the first command in a LaTeX source file.

```
\documentclass[options]{class}
```

The following document *class* names are built into LaTeX. (Many other document classes are available as separate packages; see Chapter 2 [Overview], page 3.)

`article` For a journal article, a presentation, and miscellaneous general use.

`book` Full-length books, including chapters and possibly including front matter, such as a preface, and back matter, such as an appendix (see Chapter 25 [Front/back matter], page 146).

`letter` Mail, optionally including mailing labels (see Chapter 26 [Letters], page 148).

`report` For documents of length between an `article` and a `book`, such as technical reports or theses, which may contain several chapters.

`slides` For slide presentations—rarely used today. In its place the `beamer` package is perhaps the most prevalent (see Section A.1 [beamer template], page 154).

Standard *options* are described in the next section.

3.1 Document class options

You can specify so-called *global options* or *class options* to the `\documentclass` command by enclosing them in square brackets. To specify more than one *option*, separate them with a comma, as in:

```
\documentclass[option1,option2,...]{class}
```

Here is the list of the standard class options.

All of the standard classes except `slides` accept the following options for selecting the typeface size (default is `10pt`):

```
10pt  11pt  12pt
```

All of the standard classes accept these options for selecting the paper size (these show height by width):

`a4paper` 210 by 297 mm (about 8.25 by 11.75 inches)

`a5paper` 148 by 210 mm (about 5.8 by 8.3 inches)

`b5paper` 176 by 250 mm (about 6.9 by 9.8 inches)

`executivepaper`
7.25 by 10.5 inches

`legalpaper`
8.5 by 14 inches

`letterpaper`
8.5 by 11 inches (the default)

When using one of the engines pdfLaTeX, LuaLaTeX, or XeLaTeX (see Section 2.3 [TeX engines], page 4), options other than `letterpaper` set the print area but you must also set the physical paper size. One way to do that is to put `\pdfpagewidth=\paperwidth` and `\pdfpageheight=\paperheight` in your document's preamble. The `geometry` package provides flexible ways of setting the print area and physical page size.

Miscellaneous other options:

`draft`
`final` Mark (`draft`) or do not mark (`final`) overfull boxes with a black box in the margin; default is `final`.

`fleqn` Put displayed formulas flush left; default is centered.

`landscape`
Selects landscape format; default is portrait.

`leqno` Put equation numbers on the left side of equations; default is the right side.

`openbib` Use "open" bibliography format.

`titlepage`
`notitlepage`
Specifies whether there is a separate page for the title information and for the abstract also, if there is one. The default for the `report` class is `titlepage`, for the other classes it is `notitlepage`.

The following options are not available with the `slides` class.

`onecolumn`
`twocolumn`
Typeset in one or two columns; default is `onecolumn`.

`oneside`
`twoside` Selects one- or two-sided layout; default is `oneside`, except that in the `book` class the default is `twoside`.

For one-sided printing, the text is centered on the page. For two-sided printing, the `\evensidemargin` (`\oddsidemargin`) parameter determines the distance on even (odd) numbered pages between the left side of the page and the text's left margin, with `\oddsidemargin` being 40% of the difference between `\paperwidth` and `\textwidth`, and `\evensidemargin` is the remainder.

`openright`
`openany` Determines if a chapter should start on a right-hand page; default is `openright` for `book`, and `openany` for `report`.

The `slides` class offers the option `clock` for printing the time at the bottom of each note.

3.2 Additional packages

Load a package *pkg*, with the package options given in the comma-separated list *options*, as here.

```
\usepackage[options]{pkg}.
```

To specify more than one package you can separate them with a comma, as in `\usepackage{pkg1,pkg2,...}`, or use multiple `\usepackage` commands.

Any options given in the `\documentclass` command that are unknown to the selected document class are passed on to the packages loaded with `\usepackage`.

3.3 Class and package construction

You can create new document classes and new packages. For instance, if your memos must satisfy some local requirements, such as a standard header for each page, then you could create a new class `smcmemo.cls` and begin your documents with `\documentclass{smcmemo}`.

What separates a package from a document class is that the commands in a package are useful across classes while those in a document class are specific to that class. Thus, a command to set page headers is for a package while a command to make the page headers say `Memo from the SMC Math Department` is for a class.

Inside of a class or package file you can use the at-sign `@` as a character in command names without having to surround the code containing that command with `\makeatletter` and `\makeatother`. See Section 2.4.3 [\makeatletter and \makeatother], page 6. This allow you to create commands that users will not accidentally redefine. Another technique is to preface class- or package-specific commands with some string to prevent your class or package from interfering with others. For instance, the class `smcmemo` might have commands `\smc@tolist`, `\smc@fromlist`, etc.

3.3.1 Class and package structure

A class file or package file typically has four parts.

In the *identification part*, the file says that it is a LaTeX package or class and describes itself, using the `\NeedsTeXFormat` and `\ProvidesClass` or `\ProvidesPackage` commands.

1. The *preliminary declarations part* declares some commands and can also load other files. Usually these commands will be those needed for the code used in the next part. For example, an `smcmemo` class might be called with an option to read in a file with a list of people for the to-head, as `\documentclass[mathto]{smcmemo}`, and therefore needs to define a command `\newcommand{\setto}[1]{\def\@tolist{#1}}` used in that file.
2. In the *handle options part* the class or package declares and processes its options. Class options allow a user to start their document as `\documentclass[option list]{class name}`, to modify the behavior of the class. An example is when you declare `\documentclass[11pt]{article}` to set the default document font size.
3. Finally, in the *more declarations part* the class or package usually does most of its work: declaring new variables, commands and fonts, and loading other files.

Here is a starting class file, which should be saved as `stub.cls` where LaTeX can find it, for example in the same directory as the `.tex` file.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{stub}[2017/07/06 stub to start building classes from]
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions\relax
\LoadClass{article}
```

It identifies itself, handles the class options via the default of passing them all to the `article` class, and then loads the `article` class to provide the basis for this class's code.

For more, see the official guide for class and package writers, the Class Guide, at `http://www.latex-project.org/help/documentation/clsguide.pdf` (much of the descriptions here derive from this document), or the tutorial `https://www.tug.org/TUGboat/tb26-3/tb84heff.pdf`.

3.3.2 Class and package commands

These are the commands designed to help writers of classes or packages.

`\AtBeginDvi{specials}`

Save in a box register things that are written to the `.dvi` file at the beginning of the shipout of the first page of the document.

`\AtEndOfClass{`*code*`}`
`\AtEndOfPackage{`*code*`}`

Hook to insert *code* to be executed when LaTeX finishes processing the current class or package. You can use these hooks multiple times; the `code` will be executed in the order that you called it. See also Section 8.6.1 [\AtBeginDocument], page 41.

`\CheckCommand{`*cmd*`}[`*num*`][`*default*`]{`*definition*`}`
`\CheckCommand*{`*cmd*`}[`*num*`][`*default*`]{`*definition*`}`

Like `\newcommand` (see Section 12.1 [\newcommand & \renewcommand], page 78) but does not define *cmd*; instead it checks that the current definition of *cmd* is exactly as given by *definition* and is or is not *long* as expected. A long command is a command that accepts `\par` within an argument. The *cmd* command is expected to be long with the unstarred version of `\CheckCommand`. Raises an error when the check fails. This allows you to check before you start redefining `cmd` yourself that no other package has already redefined this command.

`\ClassError{`*class name*`}{`*error text*`}{`*help text*`}`
`\PackageError{`*package name*`}{`*error text*`}{`*help text*`}`
`\ClassWarning{`*class name*`}{`*warning text*`}`
`\PackageWarning{`*package name*`}{`*warning text*`}`
`\ClassWarningNoLine{`*class name*`}{`*warning text*`}`
`\PackageWarningNoLine{`*package name*`}{`*warning text*`}`
`\ClassInfo{`*class name*`}{`*info text*`}`
`\PackageInfo{`*package name*`}{`*info text*`}`
`\ClassInfoNoLine{`*class name*`}{`*info text*`}`
`\PackageInfoNoLine{`*package name*`}{`*info text*`}`

Produce an error message, or warning or informational messages.

For `\ClassError` and `\PackageError` the message is *error text*, followed by TeX's `?` error prompt. If the user then asks for help by typing `h`, they see the *help text*.

The four warning commands are similar except that they write *warning text* on the screen with no error prompt. The four info commands write *info text*

only in the transcript file. The `NoLine` versions do not show the number of the line generating the message, while the other versions do show that number.

To format the messages, including the *help text*: use `\protect` to stop a command from expanding, get a line break with `\MessageBreak`, and get a space with `\space` when a space character does not allow it, like after a command. Note that LaTeX appends a period to the messages.

`\CurrentOption`

Expands to the name of the currently-being-processed option. Can only be used within the *code* argument of either `\DeclareOption` or `\DeclareOption*`.

`\DeclareOption{`*option*`}{`*code*`}`
`\DeclareOption*{`*code*`}`

Make an option available to a user, for invoking in their `\documentclass` command. For example, the `smcmemo` class could have an option allowing users to put the institutional logo on the first page with `\documentclass[logo]{smcmemo}`. The class file must contain `\DeclareOption{logo}{`*code*`}` (and later, `\ProcessOptions`).

If you request an option that has not been declared, by default this will produce a warning like `Unused global option(s): [badoption].` Change this behaviour with the starred version `\DeclareOption*{`*code*`}`. For example, many classes extend an existing class, using a declaration such as `\LoadClass{article}`, and for passing extra options to the underlying class use code such as this.

```
\DeclareOption*{%
\PassOptionsToClass{\CurrentOption}{article}%
}
```

Another example is that the class `smcmemo` may allow users to keep lists of memo recipients in external files. Then the user could invoke `\documentclass[math]{smcmemo}` and it will read the file `math.memo`. This code handles the file if it exists and otherwise passes the option to the `article` class.

```
\DeclareOption*{\InputIfFileExists{\CurrentOption.memo}{}{%
    \PassOptionsToClass{\CurrentOption}{article}}}
```

`\DeclareRobustCommand{`*cmd*`}[`*num*`][`*default*`]{`*definition*`}`
`\DeclareRobustCommand*{`*cmd*`}[`*num*`][`*default*`]{`*definition*`}`

Like `\newcommand` and `\newcommand*` (see Section 12.1 [\newcommand & \renewcommand], page 78) but these declare a robust command, even if some code within the *definition* is fragile. (For a discussion of robust and fragile commands see Section 12.9 [\protect], page 84.) Use this command to define new robust commands or to redefine existing commands and make them robust. Unlike `\newcommand` these do not give an error if macro *cmd* already exists; instead, a log message is put into the transcript file if a command is redefined.

Commands defined this way are a bit less efficient than those defined using `\newcommand` so unless the command's data is fragile and the command is used within a moving argument, use `\newcommand`.

The `etoolbox` package offers commands `\newrobustcmd`, `\newrobustcmd*`, `\renewrobustcmd`, `\renewrobustcmd*`, `\providerobustcmd`, and `\providerobustcmd*` which are similar to `\newcommand`, `\newcommand*`, `\renewcommand`, `\renewcommand*`, `\providecommand`, and `\providecommand*`, but define a robust *cmd* with two advantages as compared to `\DeclareRobustCommand`:

1. They use the low-level e-TEX protection mechanism rather than the higher level LATEX `\protect` mechanism, so they do not incur the slight loss of performance mentioned above, and
2. They make the same distinction between `\new...`, `\renew...`, and `\provide...`, as the standard commands, so they do not just make a log message when you redefine *cmd* that already exists, in that case you need to use either `\renew...` or `\provide...` or you get an error.

`\IfFileExists{`*file name*`}{`*true code*`}{`*false code*`}`
`\InputIfFileExists{`*file name*`}{`*true code*`}{`*false code*`}`

Execute *true code* if LATEX can find the file *file name* and *false code* otherwise. In the second case it inputs the file immediately after executing *true code*. Thus `\IfFileExists{img.pdf}{\includegraphics{img.pdf}}{\typeout{WARNING: img.pdf not found}}` will include the graphic `img.pdf` if it is found but otherwise just give a warning.

This command looks for the file in all search paths that LATEX uses, not only in the current directory. To look only in the current directory do something like `\IfFileExists{./filename}{`*true code*`}{`*false code*`}`. If you ask for a filename without a `.tex` extension then LATEX will first look for the file by appending the `.tex`; for more on how LATEX handles file extensions see Section 24.3 [\input], page 145.

`\LoadClass[`*options list*`]{`*class name*`}[`*release date*`]`
`\LoadClassWithOptions{`*class name*`}[`*release date*`]`

Load a class, as with `\documentclass[`*options list*`]{`*class name*`}[`*release info*`]`. An example is `\LoadClass[twoside]{article}`.

The *options list*, if present, is a comma-separated list. The *release date* is optional. If present it must have the form *YYYY/MM/DD*.

If you request a *release date* and the date of the package installed on your system is earlier, then you get a warning on the screen and in the log like `You have requested, on input line 4, version '2038/01/19' of document class article, but only version '2014/09/29 v1.4h Standard LaTeX document class' is available.`

The command version `\LoadClassWithOptions` uses the list of options for the current class. This means it ignores any options passed to it via `\PassOptionsToClass`. This is a convenience command that lets you build classes on existing ones, such as the standard `article` class, without having to track which options were passed.

`\ExecuteOptions{`*`options-list`*`}`

For each option *option* in the *options-list*, in order, this command executes the command `\ds@`*`option`*. If this command is not defined then that option is silently ignored.

It can be used to provide a default option list before `\ProcessOptions`. For example, if in a class file you want the default to be 11pt fonts then you could specify `\ExecuteOptions{11pt}\ProcessOptions\relax`.

`\NeedsTeXFormat{`*`format`*`}[`*`format date`*`]`

Specifies the format that this class must be run under. Often issued as the first line of a class file, and most often used as: `\NeedsTeXFormat{LaTeX2e}`. When a document using that class is processed, the format name given here must match the format that is actually being run (including that the *format* string is case sensitive). If it does not match then execution stops with an error like '`This file needs format 'LaTeX2e' but this is 'xxx'.`'

To specify a version of the format that you know to have certain features, include the optional *format date* on which those features were implemented. If present it must be in the form `YYYY/MM/DD`. If the format version installed on your system is earlier than *format date* then you get a warning like '`You have requested release '2038/01/20' of LaTeX, but only release '2016/02/01' is available.`'

`\OptionNotUsed`

Adds the current option to the list of unused options. Can only be used within the *code* argument of either `\DeclareOption` or `\DeclareOption*`.

`\PassOptionsToClass{`*`option list`*`}{`*`class name`*`}`
`\PassOptionsToPackage{`*`option list`*`}{`*`package name`*`}`

Adds the options in the comma-separated list *option list* to the options used by any future `\RequirePackage` or `\usepackage` command for package *package name* or the class *class name*.

The reason for these commands is: you may load a package any number of times with no options but if you want options then you may only supply them when you first load the package. Loading a package with options more than once will get you an error like `Option clash for package foo.` (LaTeX throws an error even if there is no conflict between the options.)

If your own code is bringing in a package twice then you can collapse that to once, for example replacing the two `\RequirePackage[landscape]{geometry}\RequirePackage[` with the single `\RequirePackage[landscape,margins=1in]{geometry}`. But if you are loading a package that in turn loads another package then you need to queue up the options you desire for this other package. For instance, suppose the package `foo` loads the package `geometry`. Instead of `\RequirePackage{foo}\RequirePackage[draft]{graphics}` you must write `\PassOptionsToPackage{draft}{graphics} \RequirePackage{foo}`. (If `foo.sty` loads an option in conflict with what you want then you may have to look into altering its source.)

These commands are useful for general users as well as class and package writers. For instance, suppose a user wants to load the `graphicx` package with

the option `draft` and also wants to use a class `foo` that loads the `graphicx` package, but without that option. The user could start their LaTeX file with `\PassOptionsToPackage{draft}{graphicx}\documentclass{foo}`.

`\ProcessOptions`
`\ProcessOptions*\@options`

Execute the code for each option that the user has invoked. Include it in the class file as `\ProcessOptions\relax` (because of the existence of the starred command).

Options come in two types. *Local options* have been specified for this particular package in the *options* argument of `\PassOptionsToPackage{options}`, `\usepackage[options]`, or `\RequirePackage[options]`. *Global options* are those given by the class user in `\documentclass[options]` (If an option is specified both locally and globally then it is local.)

When `\ProcessOptions` is called for a package `pkg.sty`, the following happens:

1. For each option *option* so far declared with `\DeclareOption`, it looks to see if that option is either a global or a local option for `pkg`. If so then it executes the declared code. This is done in the order in which these options were given in `pkg.sty`.
2. For each remaining local option, it executes the command `\ds@`*option* if it has been defined somewhere (other than by a `\DeclareOption`); otherwise, it executes the default option code given in `\DeclareOption*`. If no default option code has been declared then it gives an error message. This is done in the order in which these options were specified.

When `\ProcessOptions` is called for a class it works in the same way except that all options are local, and the default *code* for `\DeclareOption*` is `\OptionNotUsed` rather than an error.

The starred version `\ProcessOptions*` executes the options in the order specified in the calling commands, rather than in the order of declaration in the class or package. For a package this means that the global options are processed first.

`\ProvidesClass{`*class name*`}[`*release date brief additional information*`]`
`\ProvidesClass{`*class name*`}[`*release date*`]`
`\ProvidesPackage{`*package name*`}[`*release date brief additional information*`]`
`\ProvidesPackage{`*package name*`}[`*release date*`]`

Identifies the class or package, printing a message to the screen and the log file.

When a user writes `\documentclass{smcmemo}` then LaTeX loads the file `smcmemo.cls`. Similarly, a user writing `\usepackage{test}` prompts LaTeX to load the file `test.sty`. If the name of the file does not match the declared class or package name then you get a warning. Thus, if you invoke `\documentclass{smcmemo}`, and the file `smcmemo.cls` has the statement `\ProvidesClass{xxx}` then you get a warning like `You have requested document class ‘smcmemo’, but the document class provides ’xxx’.`
This warning does not prevent LaTeX from processing the rest of the class file normally.

If you include the optional argument, then you must include the date, before the first space if any, and it must have the form `YYYY/MM/DD`. The rest of the

optional argument is free-form, although it traditionally identifies the class, and is written to the screen during compilation and to the log file. Thus, if your file `smcmemo.cls` contains the line `\ProvidesClass{smcmemo}[2008/06/01 v1.0 SMC memo class]` and your document's first line is `\documentclass{smcmemo}` then you will see `Document Class: smcmemo 2008/06/01 v1.0 SMC memo class`.

The date in the optional argument allows class and package users to ask to be warned if the version of the class or package installed on their system is earlier than *release date*, by using the optional arguments such as `\documentclass{smcmemo}[2018/10/12]` or `\usepackage{foo}[[2017/07/07]]`. (Note that package users only rarely include a date, and class users almost never do.)

`\ProvidesFile{`*file name*`}[`*additional information*`]`

Declare a file other than the main class and package files, such as configuration files or font definition files. Put this command in that file and you get in the log a string like `File: test.config 2017/10/12 config file for test.cls` for *file name* equal to '`test.config`' and *additional information* equal to '`2017/10/12 config file for test.cls`'.

`\RequirePackage[`*option list*`]{`*package name*`}[`*release date*`]`
`\RequirePackageWithOptions{`*package name*`}[`*release date*`]`

Load a package, like the document author command `\usepackage`. See Section 3.2 [Additional packages], page 9. An example is `\RequirePackage[landscape,margin=1in]{geometry}`. Note that the LaTeX development team strongly recommends use of these commands over Plain TeX's `\input`; see the Class Guide.

The *option list*, if present, is a comma-separated list. The *release date*, if present, must have the form *YYYY/MM/DD*. If the release date of the package as installed on your system is earlier than *release date* then you get a warning like `You have requested, on input line 9, version '2017/07/03' of package jhtest, but only version '2000/01/01' is available`.

The `\RequirePackageWithOptions` version uses the list of options for the current class. This means it ignores any options passed to it via `\PassOptionsToClass`. This is a convenience command to allow easily building classes on existing ones without having to track which options were passed.

The difference between `\usepackage` and `\RequirePackage` is small. The `\usepackage` command is intended for the document file while `\RequirePackage` is intended for package and class files. Thus, using `\usepackage` before the `\documentclass` command causes LaTeX to give error like `\usepackage before \documentclass`, but you can use `\RequirePackage` there.

4 Fonts

Two important aspects of selecting a *font* are specifying a size and a style. The LaTeX commands for doing this are described here.

4.1 Font styles

The following type style commands are supported by LaTeX.

This first group of commands is typically used with an argument, as in `\textit{text}`. In the table below, the corresponding command in parenthesis is the "declaration form", which takes no arguments, as in `{\itshape text}`. The scope of the declaration form lasts until the next type style command or the end of the current group.

These commands, in both the argument form and the declaration form, are cumulative; e.g., you can say either `\sffamily\bfseries` or `\bfseries\sffamily` to get bold sans serif.

You can alternatively use an environment form of the declarations; for instance, `\begin{ttfamily}...\end{ttfamily}`.

These font-switching commands automatically insert italic corrections if needed. (See Section 19.6 [\/], page 116, for the details of italic corrections.) Specifically, they insert the italic correction unless the following character is in the list `\nocorrlist`, which by default consists of a period and a comma. To suppress the automatic insertion of italic correction, use `\nocorr` at the start or end of the command argument, such as `\textit{\nocorr text}` or `\textsc{text \nocorr}`.

`\textrm (\rmfamily)`
: Roman.

`\textit (\itshape)`
: Italics.

`\textmd (\mdseries)`
: Medium weight (default).

`\textbf (\bfseries)`
: Boldface.

`\textup (\upshape)`
: Upright (default).

`\textsl (\slshape)`
: Slanted.

`\textsf (\sffamily)`
: Sans serif.

`\textsc (\scshape)`
: Small caps.

`\texttt (\ttfamily)`
: Typewriter.

`\textnormal (\normalfont)`
: Main document font.

Although it also changes fonts, the `\emph{`*text*`}` command is semantic, for text to be emphasized, and should not be used as a substitute for `\textit`. For example, `\emph{`*start text* `\emph{`*middle text*`}` *end text*`}` will result in the *start text* and *end text* in italics, but *middle text* will be in roman.

LaTeX also provides the following commands, which unconditionally switch to the given style, that is, are *not* cumulative. Also, they are used differently than the above commands: `{\`*cmd*`...}` instead of `\`*cmd*`{...}`. These are two unrelated constructs.

`\bf` Switch to bold face.

`\cal` Switch to calligraphic letters for math.

`\it` Italics.

`\rm` Roman.

`\sc` Small caps.

`\sf` Sans serif.

`\sl` Slanted (oblique).

`\tt` Typewriter (monospace, fixed-width).

The `\em` command is the unconditional version of `\emph`.

(Some people consider the unconditional font-switching commands, such as `\tt`, obsolete and that only the cumulative commands (`\texttt`) should be used. Others think that both sets of commands have their place and sometimes an unconditional font switch is precisely what you want; for one example, see Section 8.4 [`description`], page 40.)

The following commands are for use in math mode. They are not cumulative, so `\mathbf{\mathit{`*symbol*`}}` does not create a boldface and italic *symbol*; instead, it will just be in italics. This is because typically math symbols need consistent typographic treatment, regardless of the surrounding environment.

`\mathrm` Roman, for use in math mode.

`\mathbf` Boldface, for use in math mode.

`\mathsf` Sans serif, for use in math mode.

`\mathtt` Typewriter, for use in math mode.

`\mathit`
`(\mit)` Italics, for use in math mode.

`\mathnormal`
For use in math mode, e.g., inside another type style declaration.

`\mathcal` Calligraphic letters, for use in math mode.

In addition, the command `\mathversion{bold}` can be used for switching to bold letters and symbols in formulas. `\mathversion{normal}` restores the default.

Finally, the command `\oldstylenums{`*numerals*`}` will typeset so-called "old-style" numerals, which have differing heights and depths (and sometimes widths) from the standard "lining" numerals, which all have the same height as uppercase letters. LaTeX's default fonts

support this, and will respect `\textbf` (but not other styles; there are no italic old-style numerals in Computer Modern). Many other fonts have old-style numerals also; sometimes the `textcomp` package must be loaded, and sometimes package options are provided to make them the default. FAQ entry: `http://www.tex.ac.uk/cgi-bin/texfaq2html?label=osf`.

4.2 Font sizes

The following standard type size commands are supported by LaTeX. The table shows the command name and the corresponding actual font size used (in points) with the '`10pt`', '`11pt`', and '`12pt`' document size options, respectively (see Section 3.1 [Document class options], page 8).

Command	`10pt`	`11pt`	`12pt`
`\tiny`	5	6	6
`\scriptsize`	7	8	8
`\footnotesize`	8	9	10
`\small`	9	10	10.95
`\normalsize` (default)	10	10.95	12
`\large`	12	12	14.4
`\Large`	14.4	14.4	17.28
`\LARGE`	17.28	17.28	20.74
`\huge`	20.74	20.74	24.88
`\Huge`	24.88	24.88	24.88

The commands as listed here are "declaration forms". The scope of the declaration form lasts until the next type style command or the end of the current group. You can also use the environment form of these commands; for instance, `\begin{tiny}...\end{tiny}`.

4.3 Low-level font commands

These commands are primarily intended for writers of macros and packages. The commands listed here are only a subset of the available ones.

`\fontencoding{`*encoding*`}`

Select the font encoding, the encoding of the output font. There are a large number of valid encodings. The most common are `OT1`, Knuth's original encoding for Computer Modern (the default), and `T1`, also known as the Cork encoding, which has support for the accented characters used by the most widespread European languages (German, French, Italian, Polish and others), which allows TeX to hyphenate words containing accented letters. For more, see `https://ctan.org/pkg/encguide`.

`\fontfamily{`*family*`}`

Select the font family. The web page `http://www.tug.dk/FontCatalogue/` provides one way to browse through many of the fonts easily used with LaTeX. Here are examples of some common families:

`pag` Avant Garde
`fvs` Bitstream Vera Sans
`pbk` Bookman

`bch` Charter
`ccr` Computer Concrete
`cmr` Computer Modern
`cmss` Computer Modern Sans Serif
`cmtt` Computer Modern Typewriter
`pcr` Courier
`phv` Helvetica
`fi4` Inconsolata
`lmr` Latin Modern
`lmss` Latin Modern Sans
`lmtt` Latin Modern Typewriter
`pnc` New Century Schoolbook
`ppl` Palatino
`ptm` Times
`uncl` Uncial
`put` Utopia
`pzc` Zapf Chancery

`\fontseries{`*`series`*`}`

Select the font series. A *series* combines a *weight* and a *width*. Typically, a font supports only a few of the possible combinations. Some common combined series values include:

`m` Medium (normal)
`b` Bold
`c` Condensed
`bc` Bold condensed
`bx` Bold extended

The possible values for weight, individually, are:

`ul` Ultra light
`el` Extra light
`l` Light
`sl` Semi light
`m` Medium (normal)
`sb` Semi bold
`b` Bold
`eb` Extra bold
`ub` Ultra bold

The possible values for width, individually, are (the meaning and relationship of these terms varies with individual typefaces):

`uc` Ultra condensed
`ec` Extra condensed
`c` Condensed
`sc` Semi condensed
`m` Medium
`sx` Semi expanded
`x` Expanded

`ex` Extra expanded
`ux` Ultra expanded

When forming the *series* string from the weight and width, drop the `m` that stands for medium weight or medium width, unless both weight and width are `m`, in which case use just one ('`m`').

`\fontshape{`*`shape`*`}`
Select font shape. Valid shapes are:

`n` Upright (normal)
`it` Italic
`sl` Slanted (oblique)
`sc` Small caps
`ui` Upright italics
`ol` Outline

The two last shapes are not available for most font families, and small caps are often missing as well.

`\fontsize{`*`size`*`}{`*`skip`*`}`
Set the font size and the line spacing. The unit of both parameters defaults to points (`pt`). The line spacing is the nominal vertical space between lines, baseline to baseline. It is stored in the parameter `\baselineskip`. The default `\baselineskip` for the Computer Modern typeface is 1.2 times the `\fontsize`. Changing `\baselineskip` directly is inadvisable since its value is reset every time a size change happens; see `\baselinestretch`, next.

`\baselinestretch`
LaTeX multiplies the line spacing by the value of the `\baselinestretch` parameter; the default factor is 1. A change takes effect when `\selectfont` (see below) is called. You can make a line skip change happen for the entire document, for instance doubling it, by doing `\renewcommand{\baselinestretch}{2.0}` in the preamble.

However, the best way to double-space a document is to use the `setspace` package. In addition to offering a number of spacing options, this package keeps the line spacing single-spaced in places where that is typically desirable, such as footnotes and figure captions. See the package documentation.

`\linespread{`*`factor`*`}`
Equivalent to `\renewcommand{\baselinestretch}{`*`factor`*`}`, and therefore must be followed by `\selectfont` to have any effect. Best specified in the preamble, or use the `setspace` package, as just described.

`\selectfont`
The effects of the font commands described above do not happen until `\selectfont` is called, as in `\fontfamily{`*`familyname`*`}\selectfont`. It is often useful to put this in a macro:
`\newcommand*{\myfont}{\fontfamily{`*`familyname`*`}\selectfont}`
(see Section 12.1 [\newcommand & \renewcommand], page 78).

`\usefont{`*`enc`*`}{`*`family`*`}{`*`series`*`}{`*`shape`*`}`

The same as invoking `\fontencoding`, `\fontfamily`, `\fontseries` and `\fontshape` with the given parameters, followed by `\selectfont`. For example:

```
\usefont{ot1}{cmr}{m}{n}
```

5 Layout

Commands for controlling the general page layout.

5.1 \onecolumn

Start a new page and produce single-column output. If the document is given the class option `onecolumn` then this is the default behavior (see Section 3.1 [Document class options], page 8).

This command is fragile (see Section 12.9 [\protect], page 84).

5.2 \twocolumn

Synopses:

```
\twocolumn
\twocolumn[prelim one column text]
```

Start a new page and produce two-column output. If the document is given the class option `twocolumn` then this is the default (see Section 3.1 [Document class options], page 8).

If the optional *prelim one column text* argument is present, it is typeset in one-column mode before the two-column typesetting starts.

This command is fragile (see Section 12.9 [\protect], page 84).

These parameters control typesetting in two-column output:

`\columnsep`
: The distance between columns. The default is 35pt. Change it with a command such as `\setlength{\columnsep}{40pt}` You must change it before the two column environment starts; in the preamble is a good place.

`\columnseprule`
: The width of the rule between columns. The rule appears halfway between the two columns. The default is 0pt, meaning that there is no rule. Change it with a command such as `\setlength{\columnseprule}{0.4pt}`, before the two-column environment starts.

`\columnwidth`
: The width of a single column. In one-column mode this is equal to `\textwidth`. In two-column mode by default LaTeX sets the width of each of the two columns to be half of `\textwidth` minus `\columnsep`.

In a two-column document, the starred environments `table*` and `figure*` are two columns wide, whereas the unstarred environments `table` and `figure` take up only one column (see Section 8.10 [figure], page 44, and see Section 8.22 [table], page 61). LaTeX places starred floats at the top of a page. The following parameters control float behavior of two-column output.

`\dbltopfraction`
: The maximum fraction at the top of a two-column page that may be occupied by two-column wide floats. The default is 0.7, meaning that the height of a `table*` or `figure*` environment must not exceed `0.7\textheight` . If the

height of your starred float environment exceeds this then you can take one of the following actions to prevent it from floating all the way to the back of the document:

- Use the `[tp]` location specifier to tell LaTeX to try to put the bulky float on a page by itself, as well as at the top of a page.
- Use the `[t!]` location specifier to override the effect of `\dbltopfraction` for this particular float.
- Increase the value of `\dbltopfraction` to a suitably large number, to avoid going to float pages so soon.

You can redefine it, for instance with `\renewcommand{\dbltopfraction}{0.9}`.

`\dblfloatpagefraction`
: For a float page of two-column wide floats, this is the minimum fraction that must be occupied by floats, limiting the amount of blank space. LaTeX's default is `0.5`. Change it with `\renewcommand`.

`\dblfloatsep`
: On a float page of two-column wide floats, this length is the distance between floats, at both the top and bottom of the page. The default is `12pt plus2pt minus2pt` for a document set at `10pt` or `11pt`, and `14pt plus2pt minus4pt` for a document set at `12pt`.

`\dbltextfloatsep`
: This length is the distance between a multi-column float at the top or bottom of a page and the main text. The default is `20pt plus2pt minus4pt`.

`\dbltopnumber`
: On a float page of two-column wide floats, this counter gives the maximum number of floats allowed at the top of the page. The LaTeX default is `2`.

This example uses `\twocolumn`'s optional argument of to create a title that spans the two-column article:

```
\documentclass[twocolumn]{article}
\newcommand{\authormark}[1]{\textsuperscript{#1}}
\begin{document}
\twocolumn[{% inside this optional argument goes one-column text
  \centering
  \LARGE The Title \\[1.5em]
  \large Author One\authormark{1},
         Author Two\authormark{2},
         Author Three\authormark{1} \\[1em]
  \normalsize
  \begin{tabular}{p{.2\textwidth}@{\hspace{2em}}p{.2\textwidth}}
    \authormark{1}Department one  &\authormark{2}Department two \\
     School one                   &School two
  \end{tabular}\\[3em] % space below title part
  }]

Two column text here.
```

5.3 \flushbottom

Make all pages in the documents after this declaration have the same height, by stretching the vertical space where necessary to fill out the page. This is most often used when making two-sided documents since the differences in facing pages can be glaring.

If TEX cannot satisfactorily stretch the vertical space in a page then you get a message like '`Underfull \vbox (badness 10000) has occurred while \output is active`'. If you get that, one option is to change to `\raggedbottom` (see Section 5.4 [\raggedbottom], page 25). Alternatively, you can adjust the `textheight` to make compatible pages, or you can add some vertical stretch glue between lines or between paragraphs, as in `\setlength{\parskip}{0ex plus0.1ex}`. Your last option is to, in a final editing stage, adjust the height of individual pages (see Section 10.4 [\enlargethispage], page 73).

The `\flushbottom` state is the default only if you select the `twoside` document class option (see Section 3.1 [Document class options], page 8).

5.4 \raggedbottom

Make all later pages the natural height of the material on that page; no rubber vertical lengths will be stretched. Thus, in a two-sided document the facing pages may be different heights. This command can go at any point in the document body. See Section 5.3 [\flushbottom], page 25.

This is the default unless you select the `twoside` document class option (see Section 3.1 [Document class options], page 8).

5.5 Page layout parameters

`\columnsep`
`\columnseprule`
`\columnwidth`

The distance between the two columns, the width of a rule between the columns, and the width of the columns, when the document class option `twocolumn` is in effect (see Section 3.1 [Document class options], page 8). See Section 5.2 [\twocolumn], page 23.

`\headheight`

Height of the box that contains the running head. The default in the `article`, `report`, and `book` classes is '`12pt`', at all type sizes.

`\headsep` Vertical distance between the bottom of the header line and the top of the main text. The default in the `article` and `report` classes is '`25pt`'. In the `book` class the default is: if the document is set at 10pt then it is '`0.25in`', and at 11pt and 12pt it is '`0.275in`'.

`\footskip`

Distance from the baseline of the last line of text to the baseline of the page footer. The default in the `article` and `report` classes is '`30pt`'. In the `book` class the default is: when the type size is 10pt the default is '`0.35in`', while at 11pt it is '`0.38in`', and at 12pt it is '`30pt`'.

`\linewidth`
: Width of the current line, decreased for each nested `list` (see Section 8.16 [list], page 48). That is, the nominal value for `\linewidth` is to equal `\textwidth` but for each nested list the `\linewidth` is decreased by the sum of that list's `\leftmargin` and `\rightmargin` (see Section 8.14 [itemize], page 47).

`\marginparpush`
`\marginsep`
`\marginparwidth`
: The minimum vertical space between two marginal notes, the horizontal space between the text body and the marginal notes, and the horizontal width of the notes.

  Normally marginal notes appear on the outside of the page, but the declaration `\reversemarginpar` changes that (and `\normalmarginpar` changes it back).

  The defaults for `\marginparpush` in both `book` and `article` classes are: '`7pt`' if the document is set at 12pt, and '`5pt`' if the document is set at 11pt or 10pt.

  For `\marginsep`, in `article` class the default is '`10pt`' except if the document is set at 10pt and in two-column mode where the default is '`11pt`'.

  For `\marginsep` in `book` class the default is '`10pt`' in two-column mode and '`7pt`' in one-column mode.

  For `\marginparwidth` in both `book` and `article` classes, in two-column mode the default is 60% of `\paperwidth` − `\textwidth`, while in one-column mode it is 50% of that distance.

`\oddsidemargin`
`\evensidemargin`
: The `\oddsidemargin` is the extra distance between the left side of the page and the text's left margin, on odd-numbered pages when the document class option `twoside` is chosen and on all pages when `oneside` is in effect. When `twoside` is in effect, on even-numbered pages the extra distance on the left is `evensidemargin`.

  LaTeX's default is that `\oddsidemargin` is 40% of the difference between `\paperwidth` and `\textwidth`, and `\evensidemargin` is the remainder.

`\paperheight`
: The height of the paper, as distinct from the height of the print area. It is normally set with a document class option, as in `\documentclass[a4paper]{article}` (see Section 3.1 [Document class options], page 8).

`\paperwidth`
: The width of the paper, as distinct from the width of the print area. It is normally set with a document class option, as in `\documentclass[a4paper]{article}` (see Section 3.1 [Document class options], page 8).

`\textheight`
: The normal vertical height of the page body. If the document is set at a nominal type size of 10pt then for an `article` or `report` the default is

'43\baselineskip', while for a book it is '41\baselineskip'. At a type size of 11pt the default is '38\baselineskip' for all document classes. At 12pt it is '36\baselineskip' for all classes.

\textwidth
: The full horizontal width of the entire page body. For an article or report document, the default is '345pt' when the chosen type size is 10pt, the default is '360pt' at 11pt, and it is '390pt' at 12pt. For a book document, the default is '4.5in' at a type size of 10pt, and '5in' at 11pt or 12pt.

  In multi-column output, \textwidth remains the width of the entire page body, while \columnwidth is the width of one column (see Section 5.2 [\twocolumn], page 23).

  In lists (see Section 8.16 [list], page 48), \textwidth remains the width of the entire page body (and \columnwidth the width of the entire column), while \linewidth may decrease for nested lists.

  Inside a minipage (see Section 8.18 [minipage], page 54) or \parbox (see Section 20.5 [\parbox], page 120), all the width-related parameters are set to the specified width, and revert to their normal values at the end of the minipage or \parbox.

  This entry is included for completeness: \hsize is the TEX primitive parameter used when text is broken into lines. It should not be used in normal LATEX documents.

\topmargin
: Space between the top of the TEX page (one inch from the top of the paper, by default) and the top of the header. The value is computed based on many other parameters: \paperheight − 2in − \headheight − \headsep − \textheight − \footskip, and then divided by two.

\topskip
: Minimum distance between the top of the page body and the baseline of the first line of text. For the standard classes, the default is the same as the font size, e.g., '10pt' at a type size of 10pt.

5.6 Floats

Some typographic elements, such as figures and tables, cannot be broken across pages. They must be typeset outside of the normal flow of text, for instance floating to the top of a later page.

LATEX can have a number of different classes of floating material. The default is the two classes, figure (see Section 8.10 [figure], page 44) and table (see Section 8.22 [table], page 61), but you can create a new class with the package float.

Within any one float class LATEX always respects the order, so that the first figure in a document source must be typeset before the second figure. However, LATEX may mix the classes, so it can happen that while the first table appears in the source before the first figure, it appears in the output after it.

The placement of floats is subject to parameters, given below, that limit the number of floats that can appear at the top of a page, and the bottom, etc. If so many floats are

queued that the limits prevent them all from fitting on a page then LaTeX places what it can and defers the rest to the next page. In this way, floats may end up being typeset far from their place in the source. In particular, a float that is big may migrate to the end of the document. In which event, because all floats in a class must appear in sequential order, every following float in that class also appears at the end.

In addition to changing the parameters, for each float you can tweak where the float placement algorithm tries to place it by using its *placement* argument. The possible values are a sequence of the letters below. The default for both `figure` and `table`, in both `article` and `book` classes, is `tbp`.

`t` (Top)—at the top of a text page.

`b` (Bottom)—at the bottom of a text page. (However, `b` is not allowed for full-width floats (`figure*`) with double-column output. To ameliorate this, use the `stfloats` or `dblfloatfix` package, but see the discussion at caveats in the FAQ: `http://www.tex.ac.uk/cgi-bin/texfaq2html?label=2colfloat`.

`h` (Here)—at the position in the text where the `figure` environment appears. However, `h` is not allowed by itself; `t` is automatically added.

To absolutely force a float to appear "here", you can `\usepackage{float}` and use the `H` specifier which it defines. For further discussion, see the FAQ entry at `http://www.tex.ac.uk/cgi-bin/texfaq2html?label=figurehere`.

`p` (Page of floats)—on a separate *float page*, which is a page containing no text, only floats.

`!` Used in addition to one of the above; for this float only, LaTeX ignores the restrictions on both the number of floats that can appear and the relative amounts of float and non-float text on the page. The `!` specifier does *not* mean "put the float here"; see above.

Note: the order in which letters appear in the *placement* argument does not change the order in which LaTeX tries to place the float; for instance, `btp` has the same effect as `tbp`. All that *placement* does is that if a letter is not present then the algorithm does not try that location. Thus, LaTeX's default of `tbp` is to try every location except placing the float where it occurs in the source.

To prevent LaTeX from moving floats to the end of the document or a chapter you can use a `\clearpage` command to start a new page and insert all pending floats. If a pagebreak is undesirable then you can use the `afterpage` package and issue `\afterpage{\clearpage}`. This will wait until the current page is finished and then flush all outstanding floats.

LaTeX can typeset a float before where it appears in the source (although on the same output page) if there is a `t` specifier in the *placement* parameter. If this is not desired, and deleting the `t` is not acceptable as it keeps the float from being placed at the top of the next page, then you can prevent it by either using the `flafter` package or using the command `\suppressfloats[t]`, which causes floats for the top position on this page to moved to the next page.

Parameters relating to fractions of pages occupied by float and non-float text (change them with `\renewcommand{`*`parameter`*`}{`*`decimal between 0 and 1`*`}`):

`\bottomfraction`
: The maximum fraction of the page allowed to be occupied by floats at the bottom; default '`.3`'.

`\floatpagefraction`
: The minimum fraction of a float page that must be occupied by floats; default '`.5`'.

`\textfraction`
: Minimum fraction of a page that must be text; if floats take up too much space to preserve this much text, floats will be moved to a different page. The default is '`.2`'.

`\topfraction`
: Maximum fraction at the top of a page that may be occupied before floats; default '`.7`'.

Parameters relating to vertical space around floats (change them with `\setlength{`*`parameter`*`}{`*`length expression`*`}`):

`\floatsep`
: Space between floats at the top or bottom of a page; default '`12pt plus2pt minus2pt`'.

`\intextsep`
: Space above and below a float in the middle of the main text; default '`12pt plus2pt minus2pt`' for 10 point and 11 point documents, and '`14pt plus4pt minus4pt`' for 12 point documents.

`\textfloatsep`
: Space between the last (first) float at the top (bottom) of a page; default '`20pt plus2pt minus4pt`'.

Counters relating to the number of floats on a page (change them with `\setcounter{`*`ctrname`*`}{`*`natural number`*`}`):

`bottomnumber`
: Maximum number of floats that can appear at the bottom of a text page; default 1.

`dbltopnumber`
: Maximum number of full-sized floats that can appear at the top of a two-column page; default 2.

`topnumber`
: Maximum number of floats that can appear at the top of a text page; default 2.

`totalnumber`
: Maximum number of floats that can appear on a text page; default 3.

The principal TEX FAQ entry relating to floats `http://www.tex.ac.uk/cgi-bin/texfaq2html?label=floats` contains suggestions for relaxing LATEX's default parameters to reduce the problem of floats being pushed to the end. A full explanation of the float

placement algorithm is in Frank Mittelbach's article "How to influence the position of float environments like figure and table in LaTeX?" (`http://latex-project.org/papers/tb111mitt-float.pdf`).

6 Sectioning

Sectioning commands provide the means to structure your text into units:

`\part`

`\chapter` (`report` and `book` class only)

`\section`

`\subsection`
`\subsubsection`
`\paragraph`
`\subparagraph`

All sectioning commands take the same general form, e.g.,

```
\chapter[toctitle]{title}
```

In addition to providing the heading *title* in the main text, the section title can appear in two other places:

1. The table of contents.
2. The running head at the top of the page.

You may not want the same text in these places as in the main text. To handle this, the sectioning commands have an optional argument *toctitle* that, when given, specifies the text for these other places.

Also, all sectioning commands have `*`-forms that print *title* as usual, but do not include a number and do not make an entry in the table of contents. For instance:

```
\section*{Preamble}
```

The `\appendix` command changes the way following sectional units are numbered. The `\appendix` command itself generates no text and does not affect the numbering of parts. The normal use of this command is something like

```
\chapter{A Chapter}
...
\appendix
\chapter{The First Appendix}
```

The `secnumdepth` counter controls printing of section numbers. The setting

```
\setcounter{secnumdepth}{level}
```

suppresses heading numbers at any depth > *level*, where `chapter` is level zero. The default `secnumdepth` is 3 in LaTeX's `article` class and 2 in the `book` and `report` classes. (See Section 13.4 [\setcounter], page 89.)

6.1 \@startsection

Synopsis:

```
\@startsection{name}{level}{indent}{beforeskip}{afterskip}{style}
```

Used to help redefine the behavior of commands that start sectioning divisions such as `\section` or `\subsection`.

Note that the `titlesec` package makes manipulation of sectioning easier. Further, while most requirements for sectioning commands can be satisfied with `\@startsection`, some cannot. For instance, in the standard LaTeX `book` and `report` classes the commands `\chapter` and `\report` are not constructed in this way. To make such a command you may want to use the `\secdef` command.

Technically, `\@startsection` has the form

```
\@startsection{name}{level}{indent}{beforeskip}{afterskip}{style}*[toctitle]{title}
```

(the star `*` is optional), so that issuing

```
\renewcommand{\section}{\@startsection{name}{level}{indent}{beforeskip}{afterskip}{sty
```

redefines `\section` to have the form `\section*[toctitle]{title}` (here too, the star `*` is optional). See Chapter 6 [Sectioning], page 31. This implies that when you write a command like `\renewcommand{section}{...}`, the `\@startsection{...}` must come last in the definition. See the examples below.

name Name of the counter used to number the sectioning header. This counter must be defined separately. Most commonly this is either `section`, `subsection`, or `paragraph`. Although in those three cases the counter name is the same as the sectioning command itself, using the same name is not required.

Then `\the`*name* displays the title number and `\`*name*`mark` is for the page headers. See the third example below.

level An integer giving the depth of the sectioning command: 0 for `chapter` (only applies to the standard `book` and `report` classes), 1 for `section`, 2 for `subsection`, 3 for `subsubsection`, 4 for `paragraph`, and 5 for `subparagraph`. In the `book` and `report` classes `part` has level -1, while in the `article` class `part` has level 0.

If *level* is less than or equal to the value of `secnumdepth` then the titles for this sectioning command will be numbered. For instance, in an `article`, if `secnumdepth` is 1 then a `\section{Introduction}` command will produce output like "1 Introduction" while `\subsection{Discussion}` will produce output like "Discussion", without the number prefix. See [Sectioning/secnumdepth], page 31.

If *level* is less than or equal to the value of *tocdepth* then the table of contents will have an entry for this sectioning unit. For instance, in an `article`, if *tocdepth* is 1 then the table of contents will list sections but not subsections.

indent A length giving the indentation of all of the title lines with respect to the left margin. To have the title flush with the margin use `0pt`. A negative indentation such as `-\parindent` will move the title into the left margin.

beforeskip The absolute value of this length is the amount of vertical space that is inserted before this sectioning unit's title. This space will be discarded if the sectioning unit happens to start at the top of a fresh page. If this number is negative then the first paragraph following the header is not indented, if it is non-negative then the first paragraph is indented. (Note that the negative of `1pt plus 2pt minus 3pt` is `-1pt plus -2pt minus -3pt`.)

For example, if *beforeskip* is `-3.5ex plus -1ex minus -0.2ex` then to start the new sectioning unit, LaTeX will add about 3.5 times the height of a letter x in

vertical space, and the first paragraph in the section will not be indented. Using a rubber length, with `plus` and `minus`, is good practice here since it gives LaTeX more flexibility in making up the page (see Chapter 14 [Lengths], page 90).

The full accounting of the vertical space between the baseline of the line prior to this sectioning unit's header and the baseline of the header is that it is the sum of the `\parskip` of the text font, the `\baselineskip` of the title font, and the absolute value of the *beforeskip*. This space is typically rubber so it may stretch or shrink. (If the sectioning unit starts on a fresh page so that the vertical space is discarded then the baseline of the header text will be where LaTeX would put the baseline of the first text line on that page.)

afterskip This is a length. If *afterskip* is non-negative then this is the vertical space inserted after the sectioning unit's title header. If it is negative then the title header becomes a run-in header, so that it becomes part of the next paragraph. In this case the absolute value of the length gives the horizontal space between the end of the title and the beginning of the following paragraph. (Note that the negative of `1pt plus 2pt minus 3pt` is `-1pt plus -2pt minus -3pt`.)

As with *beforeskip*, using a rubber length, with `plus` and `minus` components, is good practice here since it gives LaTeX more flexibility in putting together the page.

If `afterskip` is non-negative then the full accounting of the vertical space between the baseline of the sectioning unit's header and the baseline of the first line of the following paragraph is that it is the sum of the `\parskip` of the title font, the `\baselineskip` of the text font, and the value of *after*. That space is typically rubber so it may stretch or shrink. (Note that because the sign of `afterskip` changes the sectioning unit header's from standalone to run-in, you cannot use a negative `afterskip` to cancel part of the `\parskip`.)

style Controls the styling of the title. See the examples below. Typical commands to use here are `\centering`, `\raggedright`, `\normalfont`, `\hrule`, or `\newpage`. The last command in *style* may be one such as `\MakeUppercase` or `\fbox` that takes one argument. The section title will be supplied as the argument to this command. For instance, setting *style* to `\bfseries\MakeUppercase` would produce titles that are bold and upper case.

These are LaTeX's defaults for the first three sectioning units that are defined with `\@startsection`, for the `article`, `book`, and `report` classes.

	`section`	`subsection`	`subsubsection`
[*name*], page 32,	section	subsection	subsubsection
[*level*], page 32,	1	2	3
[*indent*], page 32,	`0pt`	`0pt`	`0pt`
[*beforeskip*], page 32,	`-3.5ex plus -1ex minus -0.2ex`	`-3.25ex plus -1ex minus -0.2ex`	`-3.25ex plus -1ex minus -0.2ex`

[*afterskip*], page 33, `2.3ex plus 0.2ex` `1.5ex plus 0.2ex` `1.5ex plus 0.2ex`

[*style*], page 33, `\normalfont\Large\bfseries` `\normalfont\large\bfseries` `\normalfont\normalsize\bfseries`

Here are examples. They go either in a package or class file or in the preamble of a LaTeX document. If you put them in the preamble they must go between a `\makeatletter` command and a `\makeatother`. (Probably the error message `You can't use ‘\spacefactor’ in vertical mode.` means that you forgot this.) See Section 2.4.3 [\makeatletter and \makeatother], page 6.

This will put section titles in large boldface type, centered. It says `\renewcommand` because LaTeX's standard classes have already defined a `\section`. For the same reason it does not define a `section` counter, or the commands `\thesection` and `\l@section`.

```
\renewcommand\section{%
  \@startsection{section}% [name], page 32
    {1}% [level], page 32
    {0pt}% [indent], page 32
    {-3.5ex plus -1ex minus -.2ex}% [beforeskip], page 32
    {2.3ex plus.2ex}% [afterskip], page 33
    {\centering\normalfont\Large\bfseries}% [style], page 33
  }
```

This will put `subsection` titles in small caps type, inline with the paragraph.

```
\renewcommand\subsection{%
  \@startsection{subsection}%  [name], page 32
    {2}% [level], page 32
    {0em}% [indent], page 32
    {-1ex plus 0.1ex minus -0.05ex}% [beforeskip], page 32
    {-1em plus 0.2em}% [afterskip], page 33
    {\scshape}% [style], page 33
  }
```

The prior examples redefined existing sectional unit title commands. This defines a new one, illustrating the needed counter and macros to display that counter.

```
\setcounter{secnumdepth}{6}% show counters this far down
\newcounter{subsubparagraph}[subparagraph]% counter for numbering
\renewcommand{\thesubsubparagraph}%               how to display
  {\thesubparagraph.\@arabic\c@subsubparagraph}%  numbering
\newcommand{\subsubparagraph}{\@startsection
                        {subsubparagraph}%
                        {6}%
                        {0em}%
                        {\baselineskip}%
                        {0.5\baselineskip}%
                        {\normalfont\normalsize}}
\newcommand*\l@subsubparagraph{\@dottedtocline{6}{10em}{5em}}% for toc
\newcommand{\subsubparagraphmark}[1]{}% for page headers
```

7 Cross references

One reason for numbering things such as figures and equations is to refer the reader to them, as in "See Figure~3 for more details."

Including the figure number in the source is poor practice since if that number changes as the document evolves then you must remember to update this reference by hand. Instead, LaTeX has you write a *label* like `\label{eq:GreensThm}` and refer to it with `See equation~\ref{eq:GreensThm}`.

LaTeX writes the information from the labels to a file with the same name as the file containing the `\label{...}` but with an `.aux` extension. (The information has the format `\newlabel{`*label*`}{{`*currentlabel*`}{`*pagenumber*`}}` where *currentlabel* is the current value of the macro `\@currentlabel` that is usually updated whenever you call `\refstepcounter{`*counter*`}`.)

The most common side effect of the prior paragraph happens when your document has a *forward reference*, a `\ref{`*key*`}` that appears earlier than the associated `\label{`*key*`}`; see the example in the `\pageref{...}` description. LaTeX gets the information for references from the `.aux` file. If this is the first time you are compiling the document then you will get a message `LaTeX Warning: Label(s) may have changed. Rerun to get cross references right.` and in the output the reference will appear as two question marks '`??`', in boldface. Or, if you change some things so the references change then you get the same warning and the output contains the old reference information. The solution in either case is just to compile the document a second time.

7.1 `\label`

Synopsis:

```
\label{key}
```

Assign a reference number to *key*. In ordinary text `\label{`*key*`}` assigns to *key* the number of the current sectional unit. Inside an environment with numbering, such as a `table` or `theorem` environment, `\label{`*key*`}` assigns to *key* the number of that environment. Retrieve the assigned number with the `\ref{`*key*`}` command (see Section 7.3 [\ref], page 36).

A key name can consist of any sequence of letters, digits, or common punctuation characters. Upper and lowercase letters are distinguished, as usual.

A common convention is to use labels consisting of a prefix and a suffix separated by a colon or period. This helps to avoid accidentally creating two labels with the same name, and makes your source more readable. Some commonly-used prefixes:

`ch`	for chapters
`sec`	for lower-level sectioning commands
`fig`	for figures
`tab`	for tables
`eq`	for equations

Thus, `\label{fig:Euler}` is a label for a figure with a portrait of the great man.

In this example below the key `sec:test` will get the number of the current section and the key `fig:test` will get the number of the figure. (Incidentally, put labels after captions in figures and tables.)

```
\section{section name}
\label{sec:test}
This is Section~\ref{sec:test}.
\begin{figure}
  ...
  \caption{caption text}
  \label{fig:test}
\end{figure}
See Figure~\ref{fig:test}.
```

7.2 \pageref{*key*}

Synopsis:

```
\pageref{key}
```

Produce the page number of the place in the text where the corresponding `\label{`*key*`}` command appears.

In this example the `\label{eq:main}` is used both for the formula number and for the page number. (Note that the two references are forward references, so this document would need to be compiled twice to resolve those.)

```
The main result is formula~\ref{eq:main} on page~\pageref{eq:main}.
  ...
\begin{equation} \label{eq:main}
   \mathbf{P}=\mathbf{NP}
\end{equation}
```

7.3 \ref{*key*}

Synopsis:

```
\ref{key}
```

Produces the number of the sectional unit, equation, footnote, figure, ..., of the corresponding `\label` command (see Section 7.1 [\label], page 35). It does not produce any text, such as the word 'Section' or 'Figure', just the bare number itself.

In this example, the `\ref{popular}` produces '2'. Note that it is a forward reference since it comes before `\label{popular}`.

```
The most widely-used format is item number~\ref{popular}.
\begin{enumerate}
\item Plain \TeX
\item \label{popular} \LaTeX
\item Con\TeX t
\end{enumerate}
```

8 Environments

LaTeX provides many environments for delimiting certain behavior. An environment begins with `\begin` and ends with `\end`, like this:

```
\begin{environment-name}
  ...
\end{environment-name}
```

The *environment-name* at the beginning must exactly match that at the end. For instance, the input `\begin{table*}...\end{table}` will cause an error like: '`! LaTeX Error: \begin{table*} on input line 5 ended by \end{table}.`'

Environments are executed within a group.

8.1 abstract

Synopsis:

```
\begin{abstract}
...
\end{abstract}
```

Produce an abstract, possibly of multiple paragraphs. This environment is only defined in the `article` and `report` document classes (see Chapter 3 [Document classes], page 8).

Using the example below in the `article` class produces a displayed paragraph. Document class option `titlepage` causes the abstract to be on a separate page (see Section 3.1 [Document class options], page 8); this is the default only in the `report` class.

```
\begin{abstract}
  We compare all known accounts of the proposal made by Porter Alexander
  to Robert E Lee at the Appomattox Court House that the army continue
  in a guerrilla war, which Lee refused.
\end{abstract}
```

The next example produces a one column abstract in a two column document (for a more flexible solution, use the package `abstract`).

```
\documentclass[twocolumn]{article}
  ...
\begin{document}
\title{Babe Ruth as Cultural Progenitor: a Atavistic Approach}
\author{Smith \\ Jones \\ Robinson\thanks{Railroad tracking grant.}}
\twocolumn[
  \begin{@twocolumnfalse}
     \maketitle
     \begin{abstract}
       Ruth was not just the Sultan of Swat, he was the entire swat
       team.
     \end{abstract}
   \end{@twocolumnfalse}
   ]
{   % by-hand insert a footnote at page bottom
```

```
 \renewcommand{\thefootnote}{\fnsymbol{footnote}}
 \footnotetext[1]{Thanks for all the fish.}
}
```

8.2 array

Synopsis:

```
\begin{array}{cols}
  column 1 entry &column 2 entry ... &column n entry \\
  ...
\end{array}
```

or

```
\begin{array}[pos]{cols}
  column 1 entry &column 2 entry ... &column n entry \\
  ...
\end{array}
```

Produce a mathematical array. This environment can only be used in math mode, and normally appears within a displayed mathematics environment such as `equation` (see Section 8.9 [equation], page 44). Column entries are separated by an ampersand (`&`). Rows are terminated with double-backslashes (see Section 9.1 [\\], page 71).

The required argument *cols* describes the number of columns, their alignment, and the formatting of the intercolumn regions. See Section 8.23 [tabular], page 62, for the complete description of *cols*, and of the other common features of the two environments, including the optional *pos* argument.

There are two ways that `array` diverges from `tabular`. The first is that `array` entries are typeset in math mode, in textstyle (except if the *cols* definition specifies the column with `p{...}`, which causes the entry to be typeset in text mode). The second is that, instead of `tabular`'s parameter `\tabcolsep`, LaTeX's intercolumn space in an `array` is governed by `\arraycolsep`, which gives half the width between columns. The default for this is '`5pt`'.

To obtain arrays with braces the standard is to use the `amsmath` package. It comes with environments `pmatrix` for an array surrounded by parentheses `(...)`, `bmatrix` for an array surrounded by square brackets `[...]`, `Bmatrix` for an array surrounded by curly braces `{...}`, `vmatrix` for an array surrounded by vertical bars `|...|`, and `Vmatrix` for an array surrounded by double vertical bars `||...||`, along with a number of other array constructs.

Here is an example of an array:

```
\begin{equation}
  \begin{array}{cr}
    \sqrt{y}  &12.3 \\
    x^2       &3.4
  \end{array}
\end{equation}
```

The next example works if `\usepackage{amsmath}` is in the preamble:

```
\begin{equation}
  \begin{vmatrix}{cc}
```

```
    a  &b \\
    c  &d
  \end{vmatrix}=ad-bc
\end{equation}
```

8.3 center

Synopsis:

```
\begin{center}
  ... text ...
\end{center}
```

Create a new paragraph consisting of a sequence of lines that are centered within the left and right margins on the current page. Use double-backslash to get a line break at a particular spot (see Section 9.1 [\\], page 71). If some text environment body is too long to fit on a line, LaTeX will insert line breaks that avoid hyphenation and avoid stretching or shrinking any interword space.

This environment inserts space above and below the text body. See Section 8.3.1 [\centering], page 39, to avoid such space, for example inside a `figure` environment.

This example produces three centered lines. There is extra vertical space between the last two lines.

```
\begin{center}
  A Thesis Submitted in Partial Fufillment \\
  of the Requirements of \\[0.5ex]
  the School of Environmental Engineering
\end{center}
```

In this example, depending on the page's line width, LaTeX may choose a line break for the part before the double backslash. If so, it will center each of the two lines and if not it will center the single line. Then LaTeX will break at the double backslash, and will center the ending.

```
\begin{center}
  My father considered that anyone who went to chapel and didn't drink
  alcohol was not to be tolerated.\\
  I grew up in that belief.  --Richard Burton
\end{center}
```

A double backslash after the final line is optional.

8.3.1 \centering

A declaration that causes material in its scope to be centered. It is most often used inside an environment such as `figure`, or in a `parbox`.

Unlike the `center` environment, the `\centering` command does not add vertical space above and below the text.

It also does not start a new paragraph; it simply changes how LaTeX formats paragraph units. If `ww {\centering xx \\ yy} zz` is surrounded by blank lines then LaTeX will create a paragraph whose first line '`ww xx`' is centered and whose second line, not centered, contains '`yy zz`'. Usually what is desired is for the scope of the declaration to contain a blank line

or the `\end` command of an environment such as `figure` or `table` that ends the paragraph unit. Thus, if `{\centering xx \\ yy\par} zz` is surrounded by blank lines then it makes a new paragraph with two centered lines '`xx`' and '`yy`', followed by a new paragraph with '`zz`' that is formatted as usual. See also the following example.

This example's `\centering` causes the graphic to be horizontally centered.

```
\begin{figure}
  \centering
  \includegraphics[width=0.6\textwidth]{ctan_lion.png}
  \caption{CTAN Lion}  \label{fig:CTANLion}
\end{figure}
```

The scope of the `\centering` ends with the `\end{figure}`.

8.4 description

Synopsis:

```
\begin{description}
\item[label of first item] text of first item
\item[label of second item] text of second item
  ...
\end{description}
```

Environment to make a labeled list of items. Each item's *label* is typeset in bold, and is flush left so that long labels continue into the first line of the item text. There must be at least one item; having none causes the LaTeX error '`Something's wrong--perhaps a missing \item`'.

This example shows the environment used for a sequence of definitions. The labels '`lama`' and '`llama`' come out in boldface with their left edges aligned on the left margin.

```
\begin{definition}
  \item[lama] A priest.
  \item[llama] A beast.
\end{definition}
```

Start list items with the `\item` command (see Section 8.16.1 [\item], page 53). Use the optional labels, as in `\item[Main point]`, because there is no sensible default. Following the `\item` is optional text, which may contain multiple paragraphs.

Since the labels are in bold style, if the label text calls for a font change given in argument style (see Section 4.1 [Font styles], page 17) then it will come out bold. For instance, if the label text calls for typewriter with `\item[\texttt{label text}]` then it will appear in bold typewriter, if that is available. The simplest way to get non-bold typewriter is to use declarative style: `\item[{\tt label text}]`. Similarly, get the standard roman font with `\item[{\rm label text}]`.

For other major LaTeX labelled list environments, see Section 8.14 [itemize], page 47, and Section 8.7 [enumerate], page 42. Unlike those environments, nesting `description` environments does not change the default label; it is boldface and flush left at all levels.

For information about list layout parameters, including the default values, and for information about customizing list layout, see Section 8.16 [list], page 48. The package `enumitem` is useful for customizing lists.

This example changes the description labels to small caps.

```
\renewcommand{\descriptionlabel}[1]{%
  {\hspace{\labelsep}\textsc{#1}}}
```

8.5 displaymath

Synopsis:

```
\begin{displaymath}
math text
\end{displaymath}
```

Environment to typeset the math text on its own line, in display style and centered. To make the text be flush-left use the global option `fleqn`; see Section 3.1 [Document class options], page 8.

In the `displaymath` environment no equation number is added to the math text. One way to get an equation number is to use the `equation` environment (see Section 8.9 [equation], page 44).

LaTeX will not break the *math text* across lines.

Note that the `amsmath` package has significantly more extensive displayed equation facilities. For example, there are a number of ways in that package for having math text broken across lines.

The construct `\[`*math text*`\]` is essentially a synonym for `\begin{displaymath}`*math text*`\end{displaymath}` but the latter is easier to work with in the source file; for instance, searching for a square bracket may get false positives but the word `displaymath` will likely be unique. (The construct `$$`*math text*`$$` from Plain TeX is sometimes mistakenly used as a synonym for `displaymath`. It is not a synonym, because the `displaymath` environment checks that it isn't started in math mode and that it ends in math mode begun by the matching environment start, because the `displaymath` environment has different vertical spacing, and because the `displaymath` environment honors the `fleqn` option.)

The output from this example is centered and alone on its line.

```
\begin{displaymath}
  \int_1^2 x^2\,dx=7/3
\end{displaymath}
```

Also, the integral sign is larger than the inline version `\( \int_1^2 x^2\,dx=7/3 \)` produces.

8.6 document

The `document` environment encloses the entire body of a document. It is required in every LaTeX document. See Section 2.1 [Starting and ending], page 3.

8.6.1 \AtBeginDocument

Synopsis:

```
\AtBeginDocument{code}
```

Save *code* and execute it when `\begin{document}` is executed, at the very end of the preamble. The code is executed after the font selection tables have been set up, so the

normal font for the document is the current font. However, the code is executed as part of the preamble so you cannot do any typesetting with it.

You can issue this command more than once; the successive code lines will be executed in the order that you gave them.

8.6.2 `\AtEndDocument`

Synopsis:

```
\AtEndDocument{code}
```

Save *code* and execute it near the end of the document. Specifically, it is executed when `\end{document}` is executed, before the final page is finished and before any leftover floating environments are processed. If you want some of the code to be executed after these two processes then include a `\clearpage` at the appropriate point in *code*.

You can issue this command more than once; the successive code lines will be executed in the order that you gave them.

8.7 `enumerate`

Synopsis:

```
\begin{enumerate}
\item[optional label of first item] text of first item
\item[optional label of second item] text of second item
...
\end{enumerate}
```

Environment to produce a numbered list of items. The format of the label numbering depends on the nesting level of this environment; see below. The default top-level numbering is '`1.`', '`2.`', etc. Each `enumerate` list environment must have at least one item; having none causes the LaTeX error '`Something's wrong--perhaps a missing \item`'.

This example gives the first two finishers in the 1908 Olympic marathon. As a top-level list the labels would come out as '`1.`' and '`2.`'.

```
\begin{enumerate}
 \item Johnny Hayes (USA)
 \item Charles Hefferon (RSA)
\end{enumerate}
```

Start list items with the `\item` command (see Section 8.16.1 [\item], page 53). If you give `\item` an optional argument by following it with square brackets, as in `\item[Interstitial label]`, then the next item will continue the interrupted sequence (see Section 8.16.1 [\item], page 53). That is, you will get labels like '`1.`', then '`Interstitial label`', then '`2.`'. Following the `\item` is optional text, which may contain multiple paragraphs.

Enumerations may be nested within other `enumerate` environments, or within any paragraph-making environment such as `itemize` (see Section 8.14 [itemize], page 47), up to four levels deep. This gives LaTeX's default for the format at each nesting level, where 1 is the top level, the outermost level.

1. arabic number followed by a period: '`1.`', '`2.`', ...
2. lower case letter inside parentheses: '`(a)`', '`(b)`' ...

3. lower case roman numeral followed by a period: ‘`i.`’, ‘`ii.`’, ...
4. upper case letter followed by a period: ‘`A.`’, ‘`B.`’, ...

The `enumerate` environment uses the counters `\enumi` through `\enumiv` (see Chapter 13 [Counters], page 87).

For other major LaTeX labeled list environments, see Section 8.4 [description], page 40, and Section 8.14 [itemize], page 47. For information about list layout parameters, including the default values, and for information about customizing list layout, see Section 8.16 [list], page 48. The package `enumitem` is useful for customizing lists.

To change the format of the label use `\renewcommand` (see Section 12.1 [\newcommand & \renewcommand], page 78) on the commands `\labelenumi` through `\labelenumiv`. For instance, this first level list will be labelled with uppercase letters, in boldface, and without a trailing period.

```
\renewcommand{\labelenumi}{\textbf{\Alph{enumi}}}
\begin{enumerate}
  \item Shows as boldface A
  \item Shows as boldface B
\end{enumerate}
```

For a list of counter-labeling commands see Section 13.1 [\alph \Alph \arabic \roman \Roman \fnsymbol], page 87.

8.8 eqnarray

First, a caveat: the `eqnarray` environment is depreciated. It has infelicities that cannot be overcome, including spacing that is inconsistent with other mathematics elements (see the article “Avoid eqnarray!” by Lars Madsen `http://tug.org/TUGboat/tb33-1/tb103madsen.pdf`). New documents should include the `amsmath` package and use the displayed mathematics environments provided there, such as the `align` environment.

Nevertheless, for completeness and for a reference when working with old documents, a synopsis:

```
\begin{eqnarray}
  first formula left  &first formula middle  &first formula right \\
  ...
\end{eqnarray}
```

or

```
\begin{eqnarray*}
  first formula left  &first formula middle  &first formula right \\
  ...
\end{eqnarray*}
```

Display a sequence of equations or inequalities. The left and right sides are typeset in display mode, while the middle is typeset in text mode.

It is similar to a three-column `array` environment, with items within a row separated by an ampersand (`&`), and with rows separated by double backslash `\\`). The starred form of line break (`\\*`) can also be used to separate equations, and will disallow a page break there (see Section 9.1 [\\], page 71).

The unstarred form `eqnarray` places an equation number on every line (using the `equation` counter), unless that line contains a `\nonumber` command. The starred form `eqnarray*` omits equation numbering, while otherwise being the same.

The command `\lefteqn` is used for splitting long formulas across lines. It typesets its argument in display style flush left in a box of zero width.

This example shows three lines. The first two lines make an inequality, while the third line has not entry on the left side.

```
\begin{eqnarray*}
  \lefteqn{x_1+x_2+\cdots+x_n}     \\
    &\leq &y_1+y_2+\cdots+y_n      \\
    &=    &z+y_3+\cdots+y_n
\end{eqnarray*}
```

8.9 `equation`

Synopsis:

```
\begin{equation}
  math text
\end{equation}
```

Make a `displaymath` environment (see Section 8.5 [displaymath], page 41) with an equation number in the right margin.

The equation number is generated using the `equation` counter.

You should have no blank lines between `\begin{equation}` and `\begin{equation}`, or LaTeX will tell you that there is a missing dollar sign, `$$`.

Note that the `amsmath` package has extensive displayed equation facilities. Those facilities are the best approach for such output in new documents.

8.10 `figure`

Synopsis:

```
\begin{figure}[placement]
  figure body
\caption[loftitle]{title}
\label{label}
\end{figure}
```

or

```
\begin{figure*}[placement]
  figure body
\caption[loftitle]{title}
\label{label}
\end{figure*}
```

A class of floats (see Section 5.6 [Floats], page 27). Because they cannot be split across pages, they are not typeset in sequence with the normal text but instead are “floated” to a convenient place, such as the top of a following page.

For the possible values of *placement* and their effect on the float placement algorithm, see Section 5.6 [Floats], page 27.

The starred form `figure*` is used when a document is in double-column mode (see Section 5.2 [\twocolumn], page 23). It produces a figure that spans both columns, at the top of the page. To add the possibility of placing at a page bottom see the discussion of *placement* `b` in Section 5.6 [Floats], page 27.

The figure body is typeset in a `parbox` of width `\textwidth` and so it can contain text, commands, etc.

The label is optional; it is used for cross references (see Chapter 7 [Cross references], page 35). The optional `\caption` command specifies caption text for the figure. By default it is numbered. If *loftitle* is present, it is used in the list of figures instead of *title* (see Section 25.1 [Tables of contents], page 146).

This example makes a figure out of a graphic. It requires one of the packages `graphics` or `graphicx`. The graphic, with its caption, will be placed at the top of a page or, if it is pushed to the end of the document, on a page of floats.

```
\begin{figure}[t]
  \centering
  \includegraphics[width=0.5\textwidth]{CTANlion.png}
  \caption{The CTAN lion, by Duane Bibby}
\end{figure}
```

8.11 `filecontents`: Write an external file

Synopsis:

```
\begin{filecontents}{filename}
  text
\end{filecontents}
```

or

```
\begin{filecontents*}{filename}
  text
\end{filecontents*}
```

Create a file named *filename* and fill it with *text*. The unstarred version of the environment `filecontents` prefixes the content of the created file with a header; see the example below. The starred version `filecontents*` does not include the header.

This environment can be used anywhere in the preamble, although it often appears before the `\documentclass` command. It is typically used when a source file requires a nonstandard style or class file. The environment will write that file to the directory containing the source and thus make the source file self-contained. Another use is to include `bib` references in the file, again to make it self-contained.

The environment checks whether a file of that name already exists and if so, does not do anything. There is a `filecontents` package that redefines the `filecontents` environment so that instead of doing nothing in that case, it will overwrite the existing file.

For example, this document

```
\documentclass{article}
```

```
\begin{filecontents}{JH.sty}
\newcommand{\myname}{Jim Hef{}feron}
\end{filecontents}
\usepackage{JH}
\begin{document}
Article by \myname.
\end{document}
```

produces this file `JH.sty`.

```
%% LaTeX2e file `JH.sty'
%% generated by the `filecontents' environment
%% from source `test' on 2015/10/12.
%%
\newcommand{\myname}{Jim Hef{}feron}
```

8.12 `flushleft`

```
\begin{flushleft}
line1 \\
line2 \\
...
\end{flushleft}
```

The `flushleft` environment allows you to create a paragraph consisting of lines that are flush to the left-hand margin and ragged right. Each line must be terminated with the string `\\`.

8.12.1 `\raggedright`

The `\raggedright` declaration corresponds to the `flushleft` environment. This declaration can be used inside an environment such as `quote` or in a `parbox`.

Unlike the `flushleft` environment, the `\raggedright` command does not start a new paragraph; it only changes how LaTeX formats paragraph units. To affect a paragraph unit's format, the scope of the declaration must contain the blank line or `\end` command that ends the paragraph unit.

8.13 `flushright`

```
\begin{flushright}
line1 \\
line2 \\
...
\end{flushright}
```

The `flushright` environment allows you to create a paragraph consisting of lines that are flush to the right-hand margin and ragged left. Each line must be terminated with the control sequence `\\`.

8.13.1 `\raggedleft`

The `\raggedleft` declaration corresponds to the `flushright` environment. This declaration can be used inside an environment such as `quote` or in a `parbox`.

Unlike the `flushright` environment, the `\raggedleft` command does not start a new paragraph; it only changes how LaTeX formats paragraph units. To affect a paragraph unit's format, the scope of the declaration must contain the blank line or `\end` command that ends the paragraph unit.

8.14 itemize

Synopsis:

```
\begin{itemize}
\item[optional label of first item] text of first item
\item[optional label of second item] text of second item
...
\end{itemize}
```

The `itemize` environment produces an "unordered", "bulleted" list. The format of the label numbering depends on the nesting level of this environment; see below. Each `itemize` list environment must have at least one item; having none causes the LaTeX error '`Something's wrong--perhaps a missing \item`'.

This example gives a two-item list. As a top-level list each label would come out as a bullet, •.

```
\begin{itemize}
 \item Pencil and watercolor sketch by Cassandra
 \item Rice portrait
\end{itemize}
```

Start list items with the `\item` command (see Section 8.16.1 [\item], page 53). If you give `\item` an optional argument by following it with square brackets, as in `\item[Optional label]`, then by default it will appear in bold and be flush right, so it could extend into the left margin. For labels that are flush left see the Section 8.4 [description], page 40, environment. Following the `\item` is optional text, which may contain multiple paragraphs.

Itemized lists can be nested within one another, up to four levels deep. They can also be nested within other paragraph-making environments, such as `enumerate` (see Section 8.7 [enumerate], page 42). The `itemize` environment uses the commands `\labelitemi` through `\labelitemiv` to produce the default label (this also uses the convention of lower case roman numerals at the end of the command names that signify the nesting level). These are the default marks at each level.

1. • (bullet, from `\textbullet`)
2. -- (bold en-dash, from `\normalfont\bfseries\textendash`)
3. * (asterisk, from `\textasteriskcentered`)
4. · (centered dot, from `\textperiodcentered`)

Change the labels with `\renewcommand`. For instance, this makes the first level use diamonds.

```
\renewcommand{\labelitemi}{$\diamond$}
```

The distance between the left margin of the enclosing environment and the left margin of the `itemize` list is determined by the parameters `\leftmargini` through `\leftmarginvi`. (Note the convention of using lower case roman numerals a the end of the command name

to denote the nesting level.) The defaults are: `2.5em` in level 1 (`2em` in two-column mode), `2.2em` in level 2, `1.87em` in level 3, and `1.7em` in level 4, with smaller values for more deeply nested levels.

For other major LaTeX labeled list environments, see Section 8.4 [description], page 40, and Section 8.7 [enumerate], page 42. For information about list layout parameters, including the default values, and for information about customizing list layout, see Section 8.16 [list], page 48. The package `enumitem` is useful for customizing lists.

This example greatly reduces the margin space for outermost itemized lists.

```
\setlength{\leftmargini}{1.25em} % default 2.5em
```

Especially for lists with short items, it may be desirable to elide space between items. Here is an example defining an `itemize*` environment with no extra spacing between items, or between paragraphs within a single item (`\parskip` is not list-specific, see Section 15.3 [\parskip], page 92):

```
\newenvironment{itemize*}%
  {\begin{itemize}%
    \setlength{\itemsep}{0pt}%
    \setlength{\parsep}{0pt}}%
    \setlength{\parskip}{0pt}}%
  {\end{itemize}}
```

8.15 `letter` environment: writing letters

This environment is used for creating letters. See Chapter 26 [Letters], page 148.

8.16 `list`

Synopsis:

```
\begin{list}{labeling}{spacing}
\item[optional label of first item] text of first item
\item[optional label of second item] text of second item
...
\end{list}
```

The `list` environment is a generic environment for constructing more specialized lists. It is most often used to create lists via the `description`, `enumerate`, and `itemize` environments (see Section 8.4 [description], page 40, Section 8.7 [enumerate], page 42, and Section 8.14 [itemize], page 47).

Also, many standard LaTeX environments that are not visually lists are constructed using `list`, including `quotation`, `quote`, `center`, `verbatim`, and plenty more (see Section 8.20 [quotation and quote], page 58, see Section 8.3 [center], page 39, see Section 8.13 [flushright], page 46).

The third-party package `enumitem` is useful for customizing lists. Here, we describe the `list` environment by defining a new custom environment.

```
\newcounter{namedlistcounter}  % number the items
\newenvironment{named}
  {\begin{list}
```

```
      {Item~\Roman{namedlistcounter}.} % labeling argument
      {\usecounter{namedlistcounter}   % spacing argument
       \setlength{\leftmargin}{3.5em}} % still spacing arg
  }
  {\end{list}}

\begin{named}
  \item Shows as ``Item~I.''
  \item[Special label.] Shows as ``Special label.''
  \item Shows as ``Item~II.''
\end{named}
```

The `list` environment's mandatory first argument, *labeling*, specifies the default labeling of list items. It can contain text and LaTeX commands, as above where it contains both '`Item`' and '`\Roman{...}`'. LaTeX forms the label by putting the *labeling* argument in a box of width `\labelwidth`. If the label is wider than that, the additional material extends to the right. When making an instance of a list you can override the default labeling by giving `\item` an optional argument by including square braces and the text, as in the above `\item[Special label.]`; see Section 8.16.1 [\item], page 53.

The label box is constructed by the command `\makelabel`. By default it positions the contents flush right. It takes one argument, the label. It typesets the contents in LR mode. An example of changing its definition is that to the above example before the definition of the `named` environment add `\newcommand{\namedmakelabel}[1]{\textsc{#1}}`, and between the `\setlength` command and the parenthesis that closes the *spacing* argument also add `\let\makelabel\namedmakelabel`. Then the items will be typeset in small caps. Similarly, changing the second code line to `\let\makelabel\fbox` puts the labels inside a framed box. Another example is at the bottom of this entry.

The mandatory second argument *spacing* can have a list of commands to redefine the spacing parameters for the list, such as `\setlength{\labelwidth}{2em}`. If this argument is empty, i.e., `{}`, then the list will have the default spacing given below. To number the items using a counter, put `\usecounter{`*`countername`*`}` in this argument (see Section 13.2 [\usecounter], page 88).

Below are the spacing parameters for list formatting. See also the figure below. Each is a length (see Chapter 14 [Lengths], page 90). The vertical spaces are normally rubber lengths, with `plus` and `minus` components, to give TeX flexibility in setting the page. Change each with a command such as `\setlength{itemsep}{2pt plus1pt minus1pt}`. For some effects these lengths should be zero or negative. Default values for derived environments such as `itemize` can be changed from the values shown here for the basic `list`.

`\itemindent`
: Extra horizontal space indentation, beyond `leftmargin`, of the first line each item. Its default value is `0pt`.

`\itemsep`
: Vertical space between items, beyond the `\parsep`. The defaults for the first three levels in LaTeX's '`article`', '`book`', and '`report`' classes at 10 point size are: `4pt plus2pt minus1pt`, `\parsep` (that is, `2pt plus1pt minus1pt`), and `\topsep` (that is, `2pt plus1pt minus1pt`). The defaults at 11 point are: `4.5pt plus2pt minus1pt`, `\parsep` (that is, `2pt plus1pt minus1pt`),

and `topsep` (that is, `2pt plus1pt minus1pt`). The defaults at 12 point are: `5pt plus2.5pt minus1pt`, `\parsep` (that is, `2.5pt plus1pt minus1pt`), and `\topsep` (that is, `2.5pt plus1pt minus1pt`).

`\labelsep`

Horizontal space between the label and text of an item. The default for LaTeX's '`article`', '`book`', and '`report`' classes is `0.5em`.

`\labelwidth`

Horizontal width. The box containing the label is nominally this wide. If `\makelabel` returns text that is wider than this then the first line of the item will be indented to make room for this extra material. If `\makelabel` returns text of width less than or equal to `\labelwidth` then LaTeX's default is that the label is typeset flush right in a box of this width.

The left edge of the label box is `\leftmargin+\itemindent-\labelsep-\labelwidth` from the left margin of the enclosing environment.

The default for LaTeX's '`article`', '`book`', and '`report`' classes at the top level is `\leftmargini-\labelsep`, (which is `2em` in one column mode and `1.5em` in two column mode). At the second level it is `\leftmarginii-\labelsep`, and at the third level it is `\leftmarginiii-\labelsep`. These definitions make the label's left edge coincide with the left margin of the enclosing environment.

`\leftmargin`

Horizontal space between the left margin of the enclosing environment (or the left margin of the page if this is a top-level list), and the left margin of this list. It must be non-negative.

In the standard LaTeX document classes, a top-level list has this set to the value of `\leftmargini`, while a list that is nested inside a top-level list has this margin set to `\leftmarginii`. More deeply nested lists get the values of `\leftmarginiii` through `\leftmarginvi`. (Nesting greater than level five generates the error message '`Too deeply nested`'.)

The defaults for the first three levels in LaTeX's '`article`', '`book`', and '`report`' classes are: `\leftmargini` is `2.5em` (in two column mode, `2em`), `\leftmarginii` is `2.2em`, and `\leftmarginiii` is `1.87em`.

`\listparindent`

Horizontal space of additional line indentation, beyond `\leftmargin`, for second and subsequent paragraphs within a list item. A negative value makes this an "outdent". Its default value is `0pt`.

`\parsep` Vertical space between paragraphs within an item. In the '`book`' and '`article`' classes The defaults for the first three levels in LaTeX's '`article`', '`book`', and '`report`' classes at 10 point size are: `4pt plus2pt minus1pt`, `2pt plus1pt minus1pt`, and `0pt`. The defaults at 11 point size are: `4.5pt plus2pt minus1pt`, `2pt plus1pt minus1pt`, and `0pt`. The defaults at 12 point size are: `5pt plus2.5pt minus1pt`, `2.5pt plus1pt minus1pt`, and `0pt`.

`\partopsep`

Vertical space added, beyond `\topsep+\parskip`, to the top and bottom of the entire environment if the list instance is preceded by a blank line. (A blank line

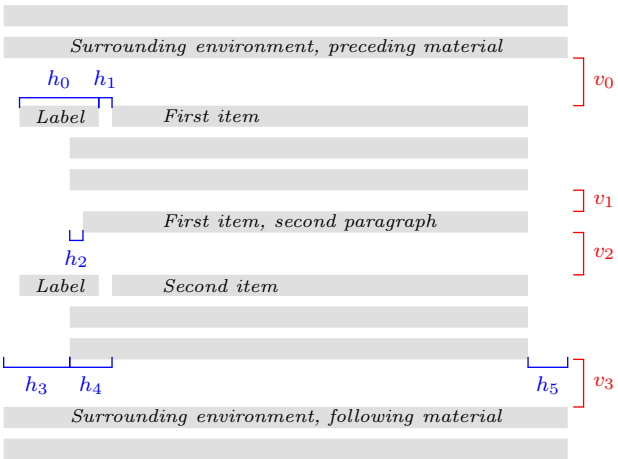
in the LaTeX source before the list changes spacing at both the top and bottom of the list; whether the line following the list is blank does not matter.)

The defaults for the first three levels in LaTeX's '`article`', '`book`', and '`report`' classes at 10 point size are: `2pt plus1 minus1pt`, `2pt plus1pt minus1pt`, and `1pt plus0pt minus1pt`. The defaults at 11 point are: `3pt plus1pt minus1pt`, `3pt plus1pt minus1pt`, and `1pt plus0pt minus1pt`). The defaults at 12 point are: `3pt plus2pt minus3pt`, `3pt plus2pt minus2pt`, and `1pt plus0pt minus1pt`.

`\rightmargin`
: Horizontal space between the right margin of the list and the right margin of the enclosing environment. Its default value is `0pt`. It must be non-negative.

`\topsep`
: Vertical space added to both the top and bottom of the list, in addition to `\parskip` (see Section 15.3 [\parskip], page 92). The defaults for the first three levels in LaTeX's '`article`', '`book`', and '`report`' classes at 10 point size are: `8pt plus2pt minus4pt`, `4pt plus2pt minus1pt`, and `2pt plus1pt minus1pt`. The defaults at 11 point are: `9pt plus3pt minus5pt`, `4.5pt plus2pt minus1pt`, and `2pt plus1pt minus1pt`. The defaults at 12 point are: `10pt plus4pt minus6pt`, `5pt plus2.5pt minus1pt`, and `2.5pt plus1pt minus1pt`.

This shows the horizontal and vertical distances.



The lengths shown are listed below. The key relationship is that the right edge of the bracket for *h1* equals the right edge of the bracket for *h4*, so that the left edge of the label box is at *h3*+*h4*-(*h0*+*h1*).

v0
: `\topsep` + `\parskip` if the list environment does not start a new paragraph, and `\topsep+\parskip+\partopsep` if it does

v1
: `\parsep`

v2
: `\itemsep+\parsep`

v3 Same as *v0*. (This space is affected by whether a blank line appears in the source above the environment; whether a blank line appears in the source below the environment does not matter.)

h1 `\labelsep`

h2 `\listparindent`

h3 `\leftmargin`

h4 `\itemindent`

h5 `\rightmargin`

The list's left and right margins, shown above as *h3* and *h5*, are with respect to the ones provided by the surrounding environment, or with respect to the page margins for a top-level list. The line width used for typesetting the list items is `\linewidth` (see Section 5.5 [Page layout parameters], page 25). For instance, set the list's left margin to be one quarter of the distance between the left and right margins of the enclosing environment with `\setlength{\leftmargin}{0.25\linewidth}`.

Page breaking in a list structure is controlled by the three parameters below. For each, the LaTeX default is `-\@lowpenalty`, that is, `-51`. Because it is negative, it somewhat encourages a page break at each spot. Change it with, e.g., `\@beginparpenalty=9999`; a value of 10000 prohibits a page break.

`\@beginparpenalty`
: The page breaking penalty for breaking before the list (default `-51`).

`\@itempenalty`
: The page breaking penalty for breaking before a list item (default `-51`).

`\@endparpenalty`
: The page breaking penalty for breaking after a list (default `-51`).

This example has the labels in red. They are numbered, and the left edge of the label lines up with the left edge of the item text.

```
\usepackage{color}
\newcounter{cnt}
\newcommand{\makeredlabel}[1]{\textcolor{red}{#1.}}
\newenvironment{redlabel}
  {\begin{list}
    {\arabic{cnt}}
    {\usecounter{cnt}
     \setlength{\labelwidth}{0em}
     \setlength{\labelsep}{0.5em}
     \setlength{\leftmargin}{1.5em}
     \setlength{\itemindent}{0.5em} % equals \labelwidth+\labelsep
     \let\makelabel=\makeredlabel
    }
  }
{\end{list} }
```

8.16.1 `\item`: An entry in a list

Synopsis:

```
\item text of item
```

or

```
\item[optional-label] text of item
```

An entry in a list. The entries are prefixed by a label, whose default depends on the list type.

Because the optional label is surrounded by square brackets '`[...]`', if you have an item whose text starts with '`[`', you have to to hide the bracket inside curly braces, as in: `\item {[} is an open square bracket`; otherwise, LaTeX will think it marks the start of an optional label.

Similarly, if the item does have the optional label and you need a close square bracket inside that label, you must hide it in the same way: `\item[Close square bracket, {]}]`. See Section 2.4 [LaTeX command syntax], page 5.

In this example the enumerate list has two items that use the default label and one that uses the optional label.

```
\begin{enumerate}
  \item Moe
  \item[sometimes] Shemp
  \item Larry
\end{enumerate}
```

The first item is labelled '`1.`', the second item is labelled '`sometimes`', and the third item is labelled '`2.`'. Because of the optional label in the second item, the third item is not labelled '`3.`'.

8.16.2 `trivlist`: A restricted form of `list`

Synopsis:

```
\begin{trivlist}
  ...
\end{trivlist}
```

A restricted version of the list environment, in which margins are not indented and an `\item` without an optional argument produces no text. It is most often used in macros, to define an environment where the `\item` command as part of the environment's definition. For instance, the `center` environment is defined essentially like this:

```
\newenvironment{center}
  {\begin{trivlist}\centering\item\relax}
  {\end{trivlist}}
```

Using `trivlist` in this way allows the macro to inherit some common code: combining vertical space of two adjacent environments; detecting whether the text following the environment should be considered a new paragraph or a continuation of the previous one; adjusting the left and right margins for possible nested list environments.

Specifically, `trivlist` uses the current values of the list parameters (see Section 8.16 [list], page 48), except that `\parsep` is set to the value of `\parskip`, and `\leftmargin`, `\labelwidth`, and `\itemindent` are set to zero.

This example outputs the items as two paragraphs, except that (by default) they have no paragraph indent and are vertically separated.

```
\begin{trivlist}
\item The \textit{Surprise} is not old; no one would call her old.
\item She has a bluff bow, lovely lines.
\end{trivlist}
```

8.17 math

Synopsis:

```
\begin{math}
math
\end{math}
```

The `math` environment inserts given *math* material within the running text. `\(...\)` and `$...$` are synonyms. See Chapter 16 [Math formulas], page 94.

8.18 minipage

```
\begin{minipage}[position][height][inner-pos]{width}
text
\end{minipage}
```

The `minipage` environment typesets its body *text* in a block that will not be broken across pages. This is similar to the `\parbox` command (see Section 20.5 [\parbox], page 120), but unlike `\parbox`, other paragraph-making environments can be used inside a minipage.

The arguments are the same as for `\parbox` (see Section 20.5 [\parbox], page 120).

By default, paragraphs are not indented in the `minipage` environment. You can restore indentation with a command such as `\setlength{\parindent}{1pc}` command.

Footnotes in a `minipage` environment are handled in a way that is particularly useful for putting footnotes in figures or tables. A `\footnote` or `\footnotetext` command puts the footnote at the bottom of the minipage instead of at the bottom of the page, and it uses the `\mpfootnote` counter instead of the ordinary `footnote` counter (see Chapter 13 [Counters], page 87).

However, don't put one minipage inside another if you are using footnotes; they may wind up at the bottom of the wrong minipage.

8.19 picture

```
\begin{picture}(width,height)(xoffset,yoffset)
... picture commands ...
\end{picture}
```

The `picture` environment allows you to create just about any kind of picture you want containing text, lines, arrows and circles. You tell LaTeX where to put things in the picture by specifying their coordinates. A coordinate is a number that may have a decimal point and a minus sign—a number like `5`, `0.3` or `-3.1416`. A coordinate specifies a length in multiples of the unit length `\unitlength`, so if `\unitlength` has been set to `1cm`, then the coordinate 2.54 specifies a length of 2.54 centimeters.

You should only change the value of `\unitlength`, using the `\setlength` command, outside of a `picture` environment. The default value is `1pt`.

The `picture` package redefine the `picture` environment so that everywhere a number is used in a *picture commands* to specify a coordinate, one can use alternatively a length. Be aware however that this will prevent scaling those lengths by changing `\unitlength`.

A *position* is a pair of coordinates, such as `(2.4,-5)`, specifying the point with x-coordinate `2.4` and y-coordinate `-5`. Coordinates are specified in the usual way with respect to an origin, which is normally at the lower-left corner of the picture. Note that when a position appears as an argument, it is not enclosed in braces; the parentheses serve to delimit the argument.

The `picture` environment has one mandatory argument which is a position (*width*,*height*), which specifies the size of the picture. The environment produces a rectangular box with these *width* and *height*.

The `picture` environment also has an optional position argument (*xoffset*,*yoffset*), following the size argument, that can change the origin. (Unlike ordinary optional arguments, this argument is not contained in square brackets.) The optional argument gives the coordinates of the point at the lower-left corner of the picture (thereby determining the origin). For example, if `\unitlength` has been set to `1mm`, the command

```
\begin{picture}(100,200)(10,20)
```

produces a picture of width 100 millimeters and height 200 millimeters, whose lower-left corner is the point (10,20) and whose upper-right corner is therefore the point (110,220). When you first draw a picture, you typically omit the optional argument, leaving the origin at the lower-left corner. If you then want to modify your picture by shifting everything, you can just add the appropriate optional argument.

The environment's mandatory argument determines the nominal size of the picture. This need bear no relation to how large the picture really is; LaTeX will happily allow you to put things outside the picture, or even off the page. The picture's nominal size is used by LaTeX in determining how much room to leave for it.

Everything that appears in a picture is drawn by the `\put` command. The command

```
\put (11.3,-.3){...}
```

puts the object specified by `...` in the picture, with its reference point at coordinates $(11.3, -.3)$. The reference points for various objects will be described below.

The `\put` command creates an *LR box*. You can put anything that can go in an `\mbox` (see Section 20.1 [\mbox], page 119) in the text argument of the `\put` command. When you do this, the reference point will be the lower left corner of the box.

The `picture` commands are described in the following sections.

8.19.1 `\circle`

Synopsis:

```
\circle[*]{diameter}
```

The `\circle` command produces a circle with a diameter as close to the specified one as possible. The `*`-form of the command draws a solid circle.

Circles up to 40 pt can be drawn.

8.19.2 \makebox

Synopsis:

```
\makebox(width,height)[position]{text}
```

The \makebox command for the picture environment is similar to the normal \makebox command except that you must specify a *width* and *height* in multiples of \unitlength.

The optional argument, [*position*], specifies the quadrant that your *text* appears in. You may select up to two of the following:

`t` Moves the item to the top of the rectangle.

`b` Moves the item to the bottom.

`l` Moves the item to the left.

`r` Moves the item to the right.

See Section 20.4 [\makebox], page 119.

8.19.3 \framebox

Synopsis:

```
\framebox(width,height)[pos]{...}
```

The \framebox command is like \makebox (see previous section), except that it puts a frame around the outside of the box that it creates.

The \framebox command produces a rule of thickness \fboxrule, and leaves a space \fboxsep between the rule and the contents of the box.

8.19.4 \dashbox

Draws a box with a dashed line. Synopsis:

```
\dashbox{dlen}(rwidth,rheight)[pos]{text}
```

\dashbox creates a dashed rectangle around *text* in a `picture` environment. Dashes are *dlen* units long, and the rectangle has overall width *rwidth* and height *rheight*. The *text* is positioned at optional *pos*.

A dashed box looks best when the *rwidth* and *rheight* are multiples of the *dlen*.

8.19.5 \frame

Synopsis:

```
\frame{text}
```

The \frame command puts a rectangular frame around *text*. The reference point is the bottom left corner of the frame. No extra space is put between the frame and the object.

8.19.6 \line

Synopsis:

```
\line(xslope,yslope){length}
```

The \line command draws a line with the given *length* and slope *xslope*/*yslope*.

Standard LaTeX can only draw lines with *slope* $= x/y$, where x and y have integer values from -6 through 6. For lines of any slope, and plenty of other shapes, see `pict2e` and many other packages on CTAN.

8.19.7 \linethickness

The `\linethickness{dim}` command declares the thickness of horizontal and vertical lines in a picture environment to be *dim*, which must be a positive length.

`\linethickness` does not affect the thickness of slanted lines, circles, or the quarter circles drawn by `\oval`.

8.19.8 \thicklines

The `\thicklines` command is an alternate line thickness for horizontal and vertical lines in a picture environment; cf. Section 8.19.7 [\linethickness], page 57, and Section 8.19.9 [\thinlines], page 57.

8.19.9 \thinlines

The `\thinlines` command is the default line thickness for horizontal and vertical lines in a picture environment; cf. Section 8.19.7 [\linethickness], page 57, and Section 8.19.8 [\thicklines], page 57.

8.19.10 \multiput

Synopsis:

```
\multiput(x,y)(delta_x,delta_y){n}{obj}
```

The `\multiput` command copies the object *obj* in a regular pattern across a picture. *obj* is first placed at position (x, y), then at $(x + \delta x, y + \delta y)$, and so on, n times.

8.19.11 \oval

Synopsis:

```
\oval(width,height)[portion]
```

The `\oval` command produces a rectangle with rounded corners. The optional argument *portion* allows you to produce only half of the oval via the following:

`t` selects the top half;

`b` selects the bottom half;

`r` selects the right half;

`l` selects the left half.

It is also possible to produce only one quarter of the oval by setting *portion* to `tr`, `br`, `bl`, or `tl`.

The "corners" of the oval are made with quarter circles with a maximum radius of 20 pt, so large "ovals" will look more like boxes with rounded corners.

8.19.12 \put

Synopsis:

```
\put(xcoord,ycoord){ ... }
```

The `\put` command places the material specified by the (mandatory) argument in braces at the given coordinate, (*xcoord*,*ycoord*).

8.19.13 \shortstack

Synopsis:

```
\shortstack[position]{...\\...\\...}
```

The `\shortstack` command produces a stack of objects. The valid positions are:

`r` Move the objects to the right of the stack.

`l` Move the objects to the left of the stack

`c` Move the objects to the centre of the stack (default)

Objects are separated with `\\`.

8.19.14 \vector

Synopsis:

```
\vector(xslope,yslope){length}
```

The `\vector` command draws a line with an arrow of the specified length and slope. The *xslope* and *yslope* values must lie between −4 and +4, inclusive.

8.20 quotation and quote

Synopsis:

```
\begin{quotation}
text
\end{quotation}
```

or

```
\begin{quote}
text
\end{quote}
```

Include a quotation.

In both environments, margins are indented on both sides by `\leftmargin` and the text is justified at both. As with the main text, leaving a blank line produces a new paragraph.

To compare the two: in the `quotation` environment, paragraphs are indented by 1.5 em and the space between paragraphs is small, `0pt plus 1pt`. In the `quote` environment, paragraphs are not indented and there is vertical space between paragraphs (it is the rubber length `\parsep`). Thus, the `quotation` environment may be more suitable for documents where new paragraphs are marked by an indent rather than by a vertical separation. In addition, `quote` may be more suitable for a short quotation or a sequence of short quotations.

```
\begin{quotation}
\it Four score and seven years ago
  ... shall not perish from the earth.
\hspace{1em plus 1fill}---Abraham Lincoln
\end{quotation}
```

8.21 tabbing

Synopsis:

```
\begin{tabbing}
row1col1 \= row1col2 ...  \\
row2col1 \> row2col2 ...  \\
...
\end{tabbing}
```

The `tabbing` environment aligns text in columns. It works by setting tab stops and tabbing to them much as was done on a typewriter. It is best suited for cases where the width of each column is constant and known in advance.

This example has a first line where the tab stops are set to explicit widths, ended by a `\kill` command (which is described below):

```
\begin{tabbing}
\hspace{0.75in}     \= \hspace{0.40in}  \= \hspace{0.40in}    \kill
Ship                \> Guns             \> Year    \\
\textit{Sophie}     \> 14               \> 1800    \\
\textit{Polychrest} \> 24               \> 1803    \\
\textit{Lively}     \> 38               \> 1804    \\
\textit{Surprise}   \> 28               \> 1805    \\
\end{tabbing}
```

Both the `tabbing` environment and the more widely-used `tabular` environment put text in columns. The most important distinction is that in `tabular` the width of columns is determined automatically by LaTeX, while in `tabbing` the user sets the tab stops. Another distinction is that `tabular` generates a box, but `tabbing` can be broken across pages. Finally, while `tabular` can be used in any mode, `tabbing` can be used only in paragraph mode and it starts a new paragraph.

A `tabbing` environment always starts a new paragraph, without indentation. Moreover, as shown in the example above, there is no need to use the starred form of the `\hspace` command at the beginning of a tabbed row. The right margin of the `tabbing` environment is the end of line, so that the width of the environment is `\linewidth`.

The `tabbing` environment contains a sequence of *tabbed rows*. The first tabbed row begins immediately after `\begin{tabbing}` and each row ends with `\\` or `\kill`. The last row may omit the `\\` and end with just `\end{tabbing}`.

At any point the `tabbing` environment has a current tab stop pattern, a sequence of $n > 0$ tab stops, numbered 0, 1, etc. These create n corresponding columns. Tab stop 0 is always the left margin, defined by the enclosing environment. Tab stop number i is set if it is assigned a horizontal position on the page. Tab stop number i can only be set if all the stops $0, \ldots, i-1$ have already been set; normally later stops are to the right of earlier ones.

By default any text typeset in a `tabbing` environment is typeset ragged right and left-aligned on the current tab stop. Typesetting is done in LR mode (see Chapter 17 [Modes], page 110).

The following commands can be used inside a `tabbing` environment. They are all fragile (see Section 12.9 [\protect], page 84).

`\\` (tabbing)
: End a tabbed line and typeset it.

`\=` (tabbing)
: Sets a tab stop at the current position.

`\>` (tabbing)
: Advances to the next tab stop.

`\<`
: Put following text to the left of the local margin (without changing the margin). Can only be used at the start of the line.

`\+`
: Moves the left margin of the next and all the following commands one tab stop to the right, beginning tabbed line if necessary.

`\-`
: Moves the left margin of the next and all the following commands one tab stop to the left, beginning tabbed line if necessary.

`\'` (tabbing)
: Moves everything that you have typed so far in the current column, i.e., everything from the most recent `\>`, `\<`, `\'`, `\\`, or `\kill` command, to the previous column and aligned to the right, flush against the current column's tab stop.

`\`` (tabbing)
: Allows you to put text flush right against any tab stop, including tab stop 0. However, it can't move text to the right of the last column because there's no tab stop there. The `\`` command moves all the text that follows it, up to the `\\` or `\end{tabbing}` command that ends the line, to the right margin of the `tabbing` environment. There must be no `\>` or `\'` command between the `\`` and the `\\` or `\end{tabbing}` command that ends the line.

`\a` (tabbing)
: In a `tabbing` environment, the commands `\=`, `\'` and `\`` do not produce accents as usual (see Section 23.5 [Accents], page 141). Instead, use the commands `\a=`, `\a'` and `\a`.

`\kill`
: Sets tab stops without producing text. Works just like `\\` except that it throws away the current line instead of producing output for it. Any `\=`, `\+` or `\-` commands in that line remain in effect.

`\poptabs`
: Restores the tab stop positions saved by the last `\pushtabs`.

`\pushtabs`
: Saves all current tab stop positions. Useful for temporarily changing tab stop positions in the middle of a `tabbing` environment.

`\tabbingsep`
: Distance of the text moved by `\'` to left of current tab stop.

This example typesets a Pascal function:

```
\begin{tabbing}
function \= fact(n : integer) : integer;\\
         \> begin \= \+ \\
                \> if \= n > 1 then \+ \\
```

```
                        fact := n * fact(n-1) \- \\
                    else \+ \\
                        fact := 1; \-\- \\
                end;\\
\end{tabbing}
```

The output looks like this:

```
function fact(n : integer) : integer;
         begin
               if n > 1 then
                  fact := n * fact(n-1);
               else
                  fact := 1;
         end;
```

(The above example is just for illustration of the environment. To actually typeset computer code in typewriter like this, a verbatim environment (see Section 8.27 [verbatim], page 69) would normally suffice. For pretty-printed code, there are quite a few packages, including `algorithm2e`, `fancyvrb`, `listings`, and `minted`.)

8.22 `table`

Synopsis:

```
\begin{table}[placement]
  table body
\caption[loftitle]{title}
\label{label}
\end{table}
```

A class of floats (see Section 5.6 [Floats], page 27). Because they cannot be split across pages, they are not typeset in sequence with the normal text but instead are "floated" to a convenient place, such as the top of a following page.

For the possible values of *placement* and their effect on the float placement algorithm, see Section 5.6 [Floats], page 27.

The table body is typeset in a `parbox` of width `\textwidth` and so it can contain text, commands, etc.

The label is optional; it is used for cross references (see Chapter 7 [Cross references], page 35). The optional `\caption` command specifies caption text for the table. By default it is numbered. If *lottitle* is present, it is used in the list of tables instead of *title* (see Section 25.1 [Tables of contents], page 146).

In this example the table and caption will float to the bottom of a page, unless it is pushed to a float page at the end.

```
\begin{table}[b]
  \centering
  \begin{tabular}{r|p{2in}} \hline
    One &The loneliest number \\
    Two &Can be as sad as one.
         It's the loneliest number since the number one.
```

```
  \end{tabular}
  \caption{Cardinal virtues}
  \label{tab:CardinalVirtues}
\end{table}
```

8.23 tabular

Synopsis:

```
\begin{tabular}[pos]{cols}
column 1 entry &column 2 entry ... &column n entry \\
  ...
\end{tabular}
```

or

```
\begin{tabular*}{width}[pos]{cols}
column 1 entry &column 2 entry ... &column n entry \\
  ...
\end{tabular*}
```

These environments produce a table, a box consisting of a sequence of horizontal rows. Each row consists of items that are aligned vertically in columns. This illustrates many of the features.

```
\begin{tabular}{l|l}
  \textit{Player name}  &\textit{Career home runs}  \\
  \hline
  Hank Aaron  &755 \\
  Babe Ruth   &714
\end{tabular}
```

The vertical format of two left-aligned columns, with a vertical bar between them, is specified in `tabular`'s argument `{l|l}`. Columns are separated with an ampersand `&`. A horizontal rule between two rows is created with `\hline`. The end of each row is marked with a double backslash `\\`. This `\\` is optional after the last row unless an `\hline` command follows, to put a rule below the table.

The required and optional arguments to `tabular` consist of:

width Required for `tabular*`, not allowed for `tabular`. Specifies the width of the `tabular*` environment. The space between columns should be rubber, as with `@{\extracolsep{\fill}}`, to allow the table to stretch or shrink to make the specified width, or else you are likely to get the `Underfull \hbox (badness 10000) in alignment ...` warning.

pos Optional. Specifies the table's vertical position. The default is to align the table so its vertical center matches the baseline of the surrounding text. There are two other possible alignments: `t` aligns the table so its top row matches the baseline of the surrounding text, and `b` aligns on the bottom row.

This only has an effect if there is other text. In the common case of a `tabular` alone in a `center` environment this option makes no difference.

cols Required. Specifies the formatting of columns. It consists of a sequence of the following specifiers, corresponding to the types of column and intercolumn material.

`l` A column of left-aligned items.

`r` A column of right-aligned items.

`c` A column of centered items.

`|` A vertical line the full height and depth of the environment.

`@{`*text or space*`}`

This inserts *text or space* at this location in every row. The *text or space* material is typeset in LR mode. This text is fragile (see Section 12.9 [\protect], page 84).

This specifier is optional: with no @-expression, LaTeX's `book`, `article`, and `report` classes will put on either side of each column a space of length `\tabcolsep`, which by default is '`6pt`'. That is, by default adjacent columns are separated by 12pt (so `\tabcolsep` is misleadingly-named since it is not the separation between tabular columns). By implication, a space of 6pt also comes before the first column and after the final column, unless you put a `@{...}` or `|` there.

If you override the default and use an @-expression then you must insert any desired space yourself, as in `@{\hspace{1em}}`.

An empty expression `@{}` will eliminate the space, including the space at the start or end, as in the example below where the tabular lines need to lie on the left margin.

```
\begin{flushleft}
  \begin{tabular}{@{}l}
    ...
  \end{tabular}
\end{flushleft}
```

This example shows text, a decimal point, between the columns, arranged so the numbers in the table are aligned on that decimal point.

```
\begin{tabular}{r@{$.$}l}
  $3$ &$14$  \\
  $9$ &$80665$
\end{tabular}
```

An `\extracolsep{`*wd*`}` command in an @-expression causes an extra space of width *wd* to appear to the left of all subsequent columns, until countermanded by another `\extracolsep` command. Unlike ordinary intercolumn space, this extra space is not suppressed by an @-expression. An `\extracolsep` command can be used only in an @-expression in the `cols` argument. Below,

LaTeX inserts the right amount of intercolumn space to make the entire table 4 inches wide.

```
\begin{center}
  \begin{tabular*}{4in}{l@{\ \ldots\extracolsep{\fill}}l}
    Seven times down, eight times up
    &such is life!
  \end{tabular*}
\end{center}
```

To insert commands that are automatically executed before a given column, load the `array` package and use the `>{...}` specifier.

`p{`*wd*`}` Each item in the column is typeset in a parbox of width *wd*.

Note that a line break double backslash `\\` may not appear in the item, except inside an environment like `minipage`, `array`, or `tabular`, or inside an explicit `\parbox`, or in the scope of a `\centering`, `\raggedright`, or `\raggedleft` declaration (when used in a `p`-column element these declarations must appear inside braces, as with `{\centering .. \\ ..}`). Otherwise LaTeX will misinterpret the double backslash as ending the row.

`*{`*num*`}{`*cols*`}`

Equivalent to *num* copies of *cols*, where *num* is a positive integer and *cols* is a list of specifiers. Thus `\begin{tabular}{|*{3}{l|r}|}` is equivalent to `\begin{tabular}{|l|rl|rl|r|}`. Note that *cols* may contain another `*`-expression.

Parameters that control formatting:

`\arrayrulewidth`

A length that is the thickness of the rule created by `|`, `\hline`, and `\vline` in the `tabular` and `array` environments. The default is '`.4pt`'. Change it as in `\setlength{\arrayrulewidth}{0.8pt}`.

`\arraystretch`

A factor by which the spacing between rows in the `tabular` and `array` environments is multiplied. The default is '`1`', for no scaling. Change it as `\renewcommand{\arraystretch}{1.2}`.

`\doublerulesep`

A length that is the distance between the vertical rules produced by the `||` specifier. The default is '`2pt`'.

`\tabcolsep`

A length that is half of the space between columns. The default is '`6pt`'. Change it with `\setlength`.

The following commands can be used inside the body of a `tabular` environment, the first two inside an entry and the second two between lines:

8.23.1 \multicolumn

Synopsis:

```
\multicolumn{numcols}{cols}{text}
```

Make an `array` or `tabular` entry that spans several columns. The first argument *numcols* gives the number of columns to span. The second argument *cols* specifies the formatting of the entry, with `c` for centered, `l` for flush left, or `r` for flush right. The third argument *text* gives the contents of that entry.

In this example, in the first row, the second and third columns are spanned by the single heading 'Name'.

```
\begin{tabular}{lccl}
  \textit{ID}       &\multicolumn{2}{c}{\textit{Name}} &\textit{Age} \\ \hline % row o
  978-0-393-03701-2 &O'Brian &Patrick                  &55           \\        % row t
    ...
\end{tabular}
```

What counts as a column is: the column format specifier for the `array` or `tabular` environment is broken into parts, where each part (except the first) begins with `l`, `c`, `r`, or `p`. So from `\begin{tabular}{|r|ccp{1.5in}|}` the parts are `|r|`, `c`, `c`, and `p{1.5in}|`.

The *cols* argument overrides the `array` or `tabular` environment's intercolumn area default adjoining this multicolumn entry. To affect that area, this argument can contain vertical bars `|` indicating the placement of vertical rules, and `@{...}` expressions. Thus if *cols* is '`|c|`' then this multicolumn entry will be centered and a vertical rule will come in the intercolumn area before it and after it. This table details the exact behavior.

```
\begin{tabular}{|cc|c|c|}
  \multicolumn{1}{r}{w}       % entry one
    &\multicolumn{1}{|r|}{x}  % entry two
    &\multicolumn{1}{|r}{y}   % entry three
    &z                        % entry four
\end{tabular}
```

Before the first entry the output will not have a vertical rule because the `\multicolumn` has the *cols* specifier '`r`' with no initial vertical bar. Between entry one and entry two there will be a vertical rule; although the first *cols* does not have an ending vertical bar, the second *cols* does have a starting one. Between entry two and entry three there is a single vertical rule; despite that the *cols* in both of the surrounding `multicolumn`'s call for a vertical rule, you only get one rule. Between entry three and entry four there is no vertical rule; the default calls for one but the *cols* in the entry three `\multicolumn` leaves it out, and that takes precedence. Finally, following entry four there is a vertical rule because of the default.

The number of spanned columns *numcols* can be 1. Besides giving the ability to change the horizontal alignment, this also is useful to override for one row the `tabular` definition's default intercolumn area specification, including the placement of vertical rules.

In the example below, in the `tabular` definition the first column is specified to default to left justified but in the first row the entry is centered with `\multicolumn{1}{c}{\textsc{Period}}`. Also in the first row, the second and third

columns are spanned by a single entry with `\multicolumn{2}{c}{\textsc{Span}}`, overriding the specification to center those two columns on the page range en-dash.

```
\begin{tabular}{l|r@{--}l}
  \multicolumn{1}{c}{\textsc{Period}}
    &multicolumn{2}{c}{\textsc{Span}} \\ \hline
  Baroque          &1600           &1760         \\
  Classical        &1730           &1820         \\
  Romantic         &1780           &1910         \\
  Impressionistic  &1875           &1925
\end{tabular}
```

Note that although the `tabular` specification by default puts a vertical rule between the first and second columns, because there is no vertical bar in the *cols* of either of the first row's `\multicolumn` commands, no rule appears in the first row.

8.23.2 \vline

Draw a vertical line in a `tabular` or `array` environment extending the full height and depth of an entry's row. Can also be used in an @-expression, although its synonym vertical bar `|` is more common. This command is rarely used in the body of a table; typically a table's vertical lines are specified in `tabular`'s *cols* argument and overridden as needed with `\multicolumn`.

This example illustrates some pitfalls. In the first line's second entry the `\hfill` moves the `\vline` to the left edge of the cell. But that is different than putting it halfway between the two columns, so in that row between the first and second columns there are two vertical rules, with the one from the `{c|cc}` specifier coming before the one produced by the `\vline\hfill`. In contrast, the first line's third entry shows the usual way to put a vertical bar between two columns. In the second line, the `ghi` is the widest entry in its column so in the `\vline\hfill` the `\hfill` has no effect and the vertical line in that entry appears immediately next to the `g`, with no whitespace.

```
\begin{tabular}{c|cc}
  x   &\vline\hfill y   &\multicolumn{1}{|r}{z} \\
  abc &def &\vline\hfill ghi
\end{tabular}
```

8.23.3 \cline

Synopsis:

```
\cline{i-j}
```

Draw a horizontal rule in an `array` or `tabular` environment beginning in column *i* and ending in column *j*. The dash `-` must appear in the mandatory argument. To span a single column use the number twice.

This example puts two horizontal lines between the first and second rows, one line in the first column only, and the other spanning the third and fourth columns. The two lines are side-by-side, at the same height.

```
\begin{tabular}{llrr}
  a &b &c &d \\ \cline{1-1} \cline{3-4}
  e &f &g &h
```

```
\end{tabular}
```

8.23.4 \hline

Draws a horizontal line the width of the enclosing `tabular` or `array` environment. It's most commonly used to draw a line at the top, bottom, and between the rows of a table.

In this example the top of the table has two horizontal rules, one above the other, that span both columns. The bottom of the table has a single rule spanning both columns. Because of the `\hline`, the `tabular` second row's line ending double backslash `\\` is required.

```
\begin{tabular}{ll} \hline\hline
  Baseball   &Red Sox  \\
  Basketball &Celtics  \\ \hline
\end{tabular}
```

8.24 thebibliography

Synopsis:

```
\begin{thebibliography}{widest-label}
\bibitem[label]{cite_key}
...
\end{thebibliography}
```

The `thebibliography` environment produces a bibliography or reference list.

In the `article` class, this reference list is labelled '`References`' and the label is stored in macro `\refname`; in the `report` class, it is labelled '`Bibliography`' and the label is stored in macro `\bibname`.

You can change the label by redefining the command `\refname` or `\bibname`, whichever is applicable depending on the class:

- For standard classes whose top level sectioning is `\chapter` (such as `book` and `report`), the label is in the macro `\bibname`;
- For standard classes whose the top level sectioning is `\section` (such as `article`), the label is in macro `\refname`.

Typically it is neither necessary nor desirable to directly redefine `\refname` or `\bibname`; language support packages like `babel` do this.

The mandatory *widest-label* argument is text that, when typeset, is as wide as the widest item label produced by the `\bibitem` commands. It is typically given as `9` for bibliographies with less than 10 references, `99` for ones with less than 100, etc.

8.24.1 \bibitem

Synopsis:

```
\bibitem[label]{cite_key}
```

The `\bibitem` command generates an entry labelled by *label*. If the *label* argument is missing, a number is automatically generated using the `enumi` counter. The *cite_key* is a *citation key* consisting in any sequence of letters, numbers, and punctuation symbols not containing a comma.

This command writes an entry to the `.aux` file containing the item's *cite_key* and *label*. When the `.aux` file is read by the `\begin{document}` command, the item's *label* is associated with `cite_key`, causing references to *cite_key* with a `\cite` command (see Section 8.24.2 [\cite], page 68) to produce the associated *label*.

8.24.2 `\cite`

Synopsis:

```
\cite[subcite]{keys}
```

The *keys* argument is a list of one or more citation keys (see Section 8.24.1 [\bibitem], page 67), separated by commas. This command generates an in-text citation to the references associated with *keys* by entries in the `.aux` file.

The text of the optional *subcite* argument appears after the citation. For example, `\cite[p.~314]{knuth}` might produce '`[Knuth, p. 314]`'.

8.24.3 `\nocite`

Synopsis:

```
\nocite{keys}
```

The `\nocite` command produces no text, but writes *keys*, which is a list of one or more citation keys, to the `.aux` file.

8.24.4 Using BibTEX

If you use the BibTEX program by Oren Patashnik (highly recommended if you need a bibliography of more than a couple of titles) to maintain your bibliography, you don't use the `thebibliography` environment (see Section 8.24 [thebibliography], page 67). Instead, you include the lines

```
\bibliographystyle{bibstyle}
\bibliography{bibfile1,bibfile2}
```

The `\bibliographystyle` command does not produce any output of its own. Rather, it defines the style in which the bibliography will be produced: *bibstyle* refers to a file *bibstyle*`.bst`, which defines how your citations will look. The standard *bibstyle* names distributed with BibTEX are:

`alpha` Sorted alphabetically. Labels are formed from name of author and year of publication.

`plain` Sorted alphabetically. Labels are numeric.

`unsrt` Like `plain`, but entries are in order of citation.

`abbrv` Like `plain`, but more compact labels.

In addition, numerous other BibTEX style files exist tailored to the demands of various publications. See `http://mirror.ctan.org/biblio/bibtex/contrib`.

The `\bibliography` command is what actually produces the bibliography. The argument to `\bibliography` refers to files named *bibfile1*`.bib`, *bibfile2*`.bib`, ..., which should contain your database in BibTEX format. Only the entries referred to via `\cite` and `\nocite` will be listed in the bibliography.

8.25 theorem

Synopsis:

```
\begin{theorem}
theorem-text
\end{theorem}
```

The `theorem` environment produces "Theorem *n*" in boldface followed by *theorem-text*, where the numbering possibilities for *n* are described under `\newtheorem` (see Section 12.7 [\newtheorem], page 82).

8.26 titlepage

Synopsis:

```
\begin{titlepage}
  ... text and spacing ...
\end{titlepage}
```

Create a title page, a page with no printed page number or heading. The following page will be numbered page one.

To instead produce a standard title page without a `titlepage` environment you can use `\maketitle` (see Section 18.1 [\maketitle], page 112).

Notice in this example that all formatting, including vertical spacing, is left to the author.

```
\begin{titlepage}
\vspace*{\stretch{1}}
\begin{center}
  {\huge\bfseries Thesis \\[1ex]
                  title}                  \\[6.5ex]
  {\large\bfseries Author name}           \\
  \vspace{4ex}
  Thesis  submitted to                    \\[5pt]
  \textit{University name}                \\[2cm]
  in partial fulfilment for the award of the degree of \\[2cm]
  \textsc{\Large Doctor of Philosophy}    \\[2ex]
  \textsc{\large Mathematics}             \\[12ex]
  \vfill
  Department of Mathematics               \\
  Address                                 \\
  \vfill
  \today
\end{center}
\vspace{\stretch{2}}
\end{titlepage}
```

8.27 verbatim

Synopsis:

```
\begin{verbatim}
```

```
literal-text
\end{verbatim}
```

The `verbatim` environment is a paragraph-making environment in which LaTeX produces exactly what you type in; for instance the `\` character produces a printed '\'. It turns LaTeX into a typewriter with carriage returns and blanks having the same effect that they would on a typewriter.

The `verbatim` environment uses a monospaced typewriter-like font (`\tt`).

8.27.1 `\verb`

Synopsis:

```
\verbcharliteral-textchar
\verb*charliteral-textchar
```

The `\verb` command typesets *literal-text* as it is input, including special characters and spaces, using the typewriter (`\tt`) font. No spaces are allowed between `\verb` or `\verb*` and the delimiter *char*, which begins and ends the verbatim text. The delimiter must not appear in *literal-text*.

The `*`-form differs only in that spaces are printed with a "visible space" character. (Namely, ␣.)

8.28 `verse`

Synopsis:

```
\begin{verse}
line1 \\
line2 \\
...
\end{verse}
```

The `verse` environment is designed for poetry, though you may find other uses for it.

The margins are indented on the left and the right, paragraphs are not indented, and the text is not justified. Separate the lines of each stanza with `\\`, and use one or more blank lines to separate the stanzas.

9 Line breaking

The first thing LaTeX does when processing ordinary text is to translate your input file into a sequence of glyphs and spaces. To produce a printed document, this sequence must be broken into lines (and these lines must be broken into pages).

LaTeX usually does the line (and page) breaking in the text body for you but in some environments you manually force line breaks.

9.1 \\

Synopsis:

```
\\[morespace]
```

or

```
\\*[morespace]
```

Start a new line. The optional argument *morespace* specifies extra vertical space to be insert before the next line. This can be a negative length. The text before the break is set at its normal length, that is, it is not stretched to fill out the line width.

Explicit line breaks in the text body are unusual in LaTeX. In particular, to start a new paragraph instead leave a blank line. This command is mostly used outside of the main flow of text such as in a `tabular` or `array` environment.

Under ordinary circumstances (e.g., outside of a `p{...}` column in a `tabular` environment) the `\newline` command is a synonym for `\\` (see Section 9.3 [\newline], page 71).

In addition to starting a new line, the starred form `\\*` tells LaTeX not to start a new page between the two lines, by issuing a `\nobreak`.

```
\title{My story: \\[0.25in]
       a tale of woe}
```

9.2 \obeycr & \restorecr

The `\obeycr` command makes a return in the input file (‘`^^M`’, internally) the same as `\\` (followed by `\relax`). So each new line in the input will also be a new line in the output.

`\restorecr` restores normal line-breaking behavior.

9.3 \newline

In ordinary text this is equivalent to double-backslash (see Section 9.1 [\\], page 71); it breaks a line, with no stretching of the text before it.

Inside a `tabular` or `array` environment, in a column with a specifier producing a paragraph box, like typically `p{...}`, `\newline` will insert a line break inside of the column, that is, it does not break the entire row. To break the entire row use `\\` or its equivalent `\tabularnewline`.

This will print ‘`Name:`’ and ‘`Address:`’ as two lines in a single cell of the table.

```
\begin{tabular}{p{1in}{\hspace{2in}}p{1in}}
  Name: \newline Address: &Date: \\ \hline
\end{tabular}
```

The ‘`Date:`’ will be baseline-aligned with ‘`Name:`’.

9.4 \- (discretionary hyphen)

The `\-` command tells LaTeX that it may hyphenate the word at that point. LaTeX is pretty good at hyphenating, and usually finds most of the correct hyphenation points, while almost never using an incorrect one. The `\-` command is used for the exceptional cases.

When you insert `\-` commands in a word, the word will only be hyphenated at those points and not at any of the hyphenation points that LaTeX might otherwise have chosen.

9.5 \discretionary (generalized hyphenation point)

Synopsis:

```
\discretionary{pre-break-text}{post-break-text}{no-break-text}
```

9.6 \fussy

The declaration `\fussy` (which is the default) makes TeX picky about line breaking. This usually avoids too much space between words, at the cost of an occasional overfull box.

This command cancels the effect of a previous `\sloppy` command (see Section 9.7 [\sloppy], page 72).

9.7 \sloppy

The declaration `\sloppy` makes TeX less fussy about line breaking. This will avoid overfull boxes, at the cost of loose interword spacing.

Lasts until a `\fussy` command is issued (see Section 9.6 [\fussy], page 72).

9.8 \hyphenation

Synopsis:

```
\hyphenation{word-one word-two}
```

The `\hyphenation` command declares allowed hyphenation points with a `-` character in the given words. The words are separated by spaces. TeX will only hyphenate if the word matches exactly, no inflections are tried. Multiple `\hyphenation` commands accumulate. Some examples (the default TeX hyphenation patterns misses the hyphenations in these words):

```
\hyphenation{ap-pen-dix col-umns data-base data-bases}
```

9.9 \linebreak & \nolinebreak

Synopses:

```
\linebreak[priority]
\nolinebreak[priority]
```

By default, the `\linebreak` (`\nolinebreak`) command forces (prevents) a line break at the current position. For `\linebreak`, the spaces in the line are stretched out so that it extends to the right margin as usual.

With the optional argument *priority*, you can convert the command from a demand to a request. The *priority* must be a number from 0 to 4. The higher the number, the more insistent the request.

10 Page breaking

LaTeX starts new pages asynchronously, when enough material has accumulated to fill up a page. Usually this happens automatically, but sometimes you may want to influence the breaks.

10.1 \cleardoublepage

The `\cleardoublepage` command ends the current page and causes all the pending floating figures and tables that have so far appeared in the input to be printed. In a two-sided printing style, it also makes the next page a right-hand (odd-numbered) page, producing a blank page if necessary.

10.2 \clearpage

The `\clearpage` command ends the current page and causes all the pending floating figures and tables that have so far appeared in the input to be printed.

10.3 \newpage

The `\newpage` command ends the current page, but does not clear floats (see Section 10.2 [\clearpage], page 73).

10.4 \enlargethispage

`\enlargethispage{size}`

`\enlargethispage*{size}`

Enlarge the `\textheight` for the current page by the specified amount; e.g., `\enlargethispage{\baselineskip}` will allow one additional line.

The starred form tries to squeeze the material together on the page as much as possible. This is normally used together with an explicit `\pagebreak`.

10.5 \pagebreak & \nopagebreak

Synopses:

```
\pagebreak[priority]
\nopagebreak[priority]
```

By default, the `\pagebreak` (`\nopagebreak`) command forces (prevents) a page break at the current position. With `\pagebreak`, the vertical space on the page is stretched out where possible so that it extends to the normal bottom margin.

With the optional argument *priority*, you can convert the `\pagebreak` command from a demand to a request. The number must be a number from 0 to 4. The higher the number, the more insistent the request is.

11 Footnotes

Place a numbered footnote at the bottom of the current page, as here.

```
Noël Coward quipped that having to read a footnote is like having
to go downstairs to answer the door, while in the midst of making
love.\footnote{I wouldn't know, I don't read footnotes.}
```

You can place multiple footnotes on a page. If the text becomes too long it will flow to the next page.

You can also produce footnotes by combining the `\footnotemark` and the `\footnotetext` commands, which is useful in special circumstances.

To make bibliographic references come out as footnotes you need to include a bibliographic style with that behavior.

11.1 `\footnote`

Synopsis:

```
\footnote[number]{text}
```

Place a numbered footnote *text* at the bottom of the current page.

```
There are over a thousand footnotes in Gibbon's
\textit{Decline and Fall of the Roman Empire}.\footnote{After
reading an early version with endnotes David Hume complained,
``One is also plagued with his Notes, according to the present Method
of printing the Book'' and suggested that they ``only to be printed
at the Margin or the Bottom of the Page.''}
```

The optional argument *number* allows you to specify the footnote number. If you use this option then the footnote number counter is not incremented, and if you do not use it then the counter is incremented.

Change how LaTeX shows the footnote counter with something like `\renewcommand{\thefootnote}{\fnsymbol{footnote}}`, which uses a sequence of symbols (see Section 13.1 [\alph \Alph \arabic \roman \Roman \fnsymbol], page 87). To make this change global put that in the preamble. If you make the change local then you may want to reset the counter with `\setcounter{footnote}{0}`. By default LaTeX uses arabic numbers.

LaTeX's default puts many restrictions on where you can use a `\footnote`; for instance, you cannot use it in an argument to a sectioning command such as `\chapter` (it can only be used in outer paragraph mode). There are some workarounds; see following sections.

In a `minipage` environment the `\footnote` command uses the `mpfootnote` counter instead of the `footnote` counter, so they are numbered independently. They are shown at the bottom of the environment, not at the bottom of the page. And by default they are shown alphabetically. See Section 8.18 [minipage], page 54.

11.2 \footnotemark

Synopsis, one of:

```
\footnotemark
\footnotemark[number]
```

Put the current footnote number in the text. (See Section 11.3 [\footnotetext], page 75, for giving the text of the footnote separately.) The version with the optional argument *number* uses that number to determine the mark printed. This command can be used in inner paragraph mode.

This example gives the same institutional affiliation to both the first and third authors (`\thanks` is a version of `footnote`).

```
\title{A Treatise on the Binomial Theorem}
\author{J Moriarty\thanks{University of Leeds}
  \and A C Doyle\thanks{Durham University}
  \and S Holmes\footnotemark[1]}
\begin{document}
\maketitle
```

If you use `\footnotemark` without the optional argument then it increments the footnote counter but if you use the optional *number* then it does not. This produces several consecutive footnote markers referring to the same footnote.

```
The first theorem\footnote{Due to Gauss.}
and the second theorem\footnotemark[\value{footnote}]
and the third theorem.\footnotemark[\value{footnote}]
```

11.3 \footnotetext

Synopsis, one of:

```
\footnotetext{text}
\footnotetext[number]{text}
```

Place *text* at the bottom of the page as a footnote. This command can come anywhere after the `\footnotemark` command. The optional argument *number* changes the displayed footnote number. The `\footnotetext` command must appear in outer paragraph mode.

11.4 Footnotes in a table

Inside a `table` environment the `\footnote` command does not work. For instance, if the code below appears on its own then the footnote simply disappears; there is a footnote mark in the table cell but nothing is set at the bottom of the page.

```
\begin{center}
     \begin{tabular}{l|l}
     \textsc{Ship}  &\textsc{Book} \\ \hline
     \textit{HMS Sophie}     &Master and Commander  \\
     \textit{HMS Polychrest} &Post Captain  \\
     \textit{HMS Lively}     &Post Captain \\
     \textit{HMS Surprise}   &A number of books\footnote{Starting with
                                HMS Surprise.}
```

```
    \end{tabular}
\end{center}
```

The solution is to surround the `tabular` environment with a `minipage` environment, as here (see Section 8.18 [minipage], page 54).

```
\begin{center}
  \begin{minipage}{.5\textwidth}
    ... tabular material ...
  \end{minipage}
\end{center}
```

The same technique will work inside a floating `table` environment (see Section 8.22 [table], page 61). To get the footnote at the bottom of the page use the `tablefootnote` package, as illustrated in this example. If you put `\usepackage{tablefootnote}` in the preamble and use the code shown then the footnote appears at the bottom and is numbered in sequence with other footnotes.

```
\begin{table}
  \centering
    \begin{tabular}{l|l}
    \textsc{Date}  &\textsc{Campaign} \\ \hline
    1862           &Fort Donelson \\
    1863           &Vicksburg     \\
    1865           &Army of Northern Virginia\footnote{Ending the war.}
    \end{tabular}
   \caption{Forces captured by US Grant}
\end{table}
```

11.5 Footnotes in section headings

Putting a footnote in a section heading, as in:

```
\section{Full sets\protect\footnote{This material due to ...}}
```

causes the footnote to appear at the bottom of the page where the section starts, as usual, but also at the bottom of the table of contents, where it is not likely to be desired. To have it not appear on the table of contents use the package `footmisc` with the `stable` option.

```
\usepackage[stable]{footmisc}
...
\begin{document}
...
\section{Full sets\footnote{This material due to ...}}
```

Note that the `\protect` is gone; including it would cause the footnote to reappear on the table of contents.

11.6 Footnotes of footnotes

Particularly in the humanities, authors can have multiple classes of footnotes, including having footnotes of footnotes. The package `bigfoot` extends LaTeX's default footnote mechanism in many ways, including allow these two, as in this example.

```
\usepackage{bigfoot}
```

```
\DeclareNewFootnote{Default}
\DeclareNewFootnote{from}[alph]   % create class \footnotefrom{}
 ...
\begin{document}
...
The third theorem is a partial converse of the
second.\footnotefrom{First noted in Wilson.\footnote{Second edition only.}}
...
```

11.7 Multiple references to footnotes

You can refer to a single footnote more than once. This example uses the package `cleverref`.

```
\usepackage{cleveref}[2012/02/15]   % this version of package or later
\crefformat{footnote}{#2\footnotemark[#1]#3}
...
\begin{document}
...
The theorem is from Evers.\footnote{\label{fn:TE}Tinker and Evers, 1994.}
The corollary is from Chance.\footnote{Evers and Chance, 1990.}
But the key lemma is from Tinker.\cref{fn:TE}
...
```

This solution will work with the package `hyperref`. See Section 11.2 [\footnotemark], page 75, for a simpler solution in the common case of multiple authors with the same affiliation.

11.8 Footnote parameters

`\footnoterule`

Produces the rule separating the main text on a page from the page's footnotes. Default dimensions: `0.4pt` thick (or wide), and `0.4\columnwidth` long in the standard document classes (except `slides`, where it does not appear).

`\footnotesep`

The height of the strut placed at the beginning of the footnote. By default, this is set to the normal strut for `\footnotesize` fonts (see Section 4.2 [Font sizes], page 19), therefore there is no extra space between footnotes. This is '`6.65pt`' for '`10pt`', '`7.7pt`' for '`11pt`', and '`8.4pt`' for '`12pt`'.

12 Definitions

LaTeX has support for making new commands of many different kinds.

12.1 \newcommand & \renewcommand

`\newcommand` and `\renewcommand` define and redefine a command, respectively. Synopses:

```
\newcommand{\cmd}[nargs][optargdefault]{defn}
\newcommand*{\cmd}[nargs][optargdefault]{defn}
\renewcommand{\cmd}[nargs][optargdefault]{defn}
\renewcommand*{\cmd}[nargs][optargdefault]{defn}
```

The starred form of these two commands requires that the arguments not contain multiple paragraphs of text (not `\long`, in plain TeX terms).

cmd Required; `\cmd` is the command name. For `\newcommand`, it must not be already defined and must not begin with `\end`. For `\renewcommand`, it must already be defined.

nargs Optional; an integer from 0 to 9, specifying the number of arguments that the command can take, including any optional argument. If this argument is not present, the default is for the command to have no arguments. When redefining a command, the new version can have a different number of arguments than the old version.

optargdefault
Optional; if this argument is present then the first argument of defined command `\cmd` is optional, with default value *optargdefault* (which may be the empty string). If this argument is not present then `\cmd` does not take an optional argument.

That is, if `\cmd` is used with square brackets following, as in `\cmd[myval]`, then within *defn* the first *positional parameter* `#1` expands *myval*. On the other hand, if `\cmd` is called without square brackets following, then within *defn* the positional parameter `#1` expands to the default *optargdefault*. In either case, any required arguments will be referred to starting with `#2`.

Omitting `[myval]` in a call is different from having the square brackets with no contents, as in `[]`. The former results in `#1` expanding to *optargdefault*; the latter results in `#1` expanding to the empty string.

defn The text to be substituted for every occurrence of `\cmd`; the positional parameter `#n` in *defn* is replaced by the text of the *n*th argument.

TeX ignores spaces in the source following an alphabetic control sequence, as in '`\cmd `'. If you actually want a space there, one solution is to type `{}` after the command ('`\cmd{} `'; another solution is to use an explicit control space ('`\cmd\ `').

A simple example of defining a new command: `\newcommand{\RS}{Robin Smith}` results in `\RS` being replaced by the longer text.

Redefining an existing command is similar: `\renewcommand{\qedsymbol}{{\small QED}}`.

Here's a command definition with one required argument:

```
\newcommand{\defref}[1]{Definition~\ref{#1}}
```

Then, `\defref{def:basis}` expands to `Definition~\ref{def:basis}`, which will ultimately expand to something like '`Definition~3.14`'.

An example with two required arguments: `\newcommand{\nbym}[2]{$#1 \times #2$}` is invoked as `\nbym{2}{k}`.

An example with an optional argument:

```
\newcommand{\salutation}[1][Sir or Madam]{Dear #1:}
```

Then, `\salutation` gives '`Dear Sir or Madam:`' while `\salutation[John]` gives '`Dear John:`'. And `\salutation[]` gives '`Dear :`'.

The braces around *defn* do not define a group, that is, they do not delimit the scope of the result of expanding *defn*. So `\newcommand{\shipname}[1]{\it #1}` is problematic; in this sentence,

```
The \shipname{Monitor} met the \shipname{Merrimac}.
```

the words '`met the`' would incorrectly be in italics. Another pair of braces in the definition is needed, like this: `\newcommand{\shipname}[1]{{\it #1}}`. Those braces are part of the definition and thus do define a group.

12.2 \providecommand

Defines a command, as long as no command of this name already exists. Synopses:

```
\providecommand{cmd}[nargs][optargdefault]{defn}
\providecommand*{cmd}[nargs][optargdefault]{defn}
```

If no command of this name already exists then this has the same effect as `\newcommand` (see Section 12.1 [\newcommand & \renewcommand], page 78). If a command of this name already exists then this definition does nothing. This is particularly useful in a style file, or other file that may be loaded more than once.

12.3 \newcounter: Allocating a counter

Synopsis, one of:

```
\newcounter{countername}
\newcounter{countername}[supercounter]
```

Globally defines a new counter named *countername* and initialize the new counter to zero.

The name *countername* must consists of letters only, and does not begin with a backslash. This name must not already be in use by another counter.

When you use the optional argument `[supercounter]` then *countername* will be numbered within, or subsidiary to, the existing counter *supercounter*. For example, ordinarily `subsection` is numbered within `section` so that any time *supercounter* is incremented with `\stepcounter` (see Section 13.7 [\stepcounter], page 89) or `\refstepcounter` (see Section 13.6 [\refstepcounter], page 89) then *countername* is reset to zero.

See Chapter 13 [Counters], page 87, for more information about counters.

12.4 \newlength: Allocating a length

Allocate a new *length* register. Synopsis:

```
\newlength{\arg}
```

This command takes one required argument, which must begin with a backslash ('\'). It creates a new length register named `\arg`, which is a place to hold (rubber) lengths such as `1in plus.2in minus.1in` (what plain TEX calls a `skip` register). The register gets an initial value of zero. The control sequence `\arg` must not already be defined.

See Chapter 14 [Lengths], page 90, for more about lengths.

12.5 \newsavebox: Allocating a box

Allocate a "bin" for holding a box. Synopsis:

```
\newsavebox{\cmd}
```

Defines `\cmd` to refer to a new bin for storing boxes. Such a box is for holding typeset material, to use multiple times (see Chapter 20 [Boxes], page 119) or to measure or manipulate. The name `\cmd` must start with a backslash ('\'), and must not be already defined.

The allocation of a box is global. This command is fragile (see Section 12.9 [\protect], page 84).

12.6 \newenvironment & \renewenvironment

These commands define or redefine an environment *env*, that is, `\begin{env}` *body* `\end{env}`. Synopses:

```
\newenvironment{env}[nargs][optargdefault]{begdefn}{enddefn}
\newenvironment*{env}[nargs][optargdefault]{begdefn}{enddefn}
\renewenvironment{env}[nargs][optargdefault]{begdefn}{enddefn}
\renewenvironment*{env}[nargs][optargdefault]{begdefn}{enddefn}
```

The starred form of these commands requires that the arguments not contain multiple paragraphs of text. The body of these environments can still contain multiple paragraphs.

env Required; the environment name. It consists only of letters or the `*` character, and thus does not begin with backslash (\). It must not begin with the string `end`. For `\newenvironment`, the name *env* must not be the name of an already existing environment, and also the command `\env` must be undefined. For `\renewenvironment`, *env* must be the name of an existing environment.

nargs Optional; an integer from 0 to 9 denoting the number of arguments of that the environment will take. When the environment is used these arguments appear after the `\begin`, as in `\begin{env}{arg1}...{argn}`. If this argument is not present then the default is for the environment to have no arguments. When redefining an environment, the new version can have a different number of arguments than the old version.

optargdefault
Optional; if this argument is present then the first argument of the defined environment is optional, with default value *optargdefault* (which may be the

empty string). If this argument is not present then the environment does not take an optional argument.

That is, when `[optargdefault]` is present in the environment definition, if `\begin{env}` is used with square brackets following, as in `\begin{env}[myval]`, then, within *begdefn*, the positional parameter `#1` expands to *myval*. If `\begin{env}` is called without square brackets following, then, within within *begdefn*, the positional parameter `#1` expands to the default *optargdefault*. In either case, any required arguments will be referred to starting with `#2`.

Omitting `[myval]` in the call is different from having the square brackets with no contents, as in `[]`. The former results in `#1` expanding to *optargdefault*; the latter results in `#1` expanding to the empty string.

begdefn Required; the text expanded at every occurrence of `\begin{env}`. Within *begdef*, the *n*th positional parameter (i.e., `#n`) is replaced by the text of the *n*th argument.

enddefn Required; the text expanded at every occurrence of `\end{env}`. This may not contain any positional parameters, so `#n` cannot be used here (but see the final example below).

All environments, that is to say the *begdefn* code, the environment body and the *enddefn* code, are processed within a group. Thus, in the first example below, the effect of the `\small` is limited to the quote and does not extend to material following the environment.

This example gives an environment like LaTeX's `quotation` except that it will be set in smaller type:

```
\newenvironment{smallquote}{%
  \small\begin{quotation}
}{%
  \end{quotation}
}
```

This one shows the use of arguments; it gives a quotation environment that cites the author:

```
\newenvironment{citequote}[1][Shakespeare]{%
  \begin{quotation}
  \noindent\textit{#1}:
}{%
  \end{quotation}
}
```

The author's name is optional, and defaults to '`Shakespeare`'. In the document, use the environment like this:

```
\begin{citequote}[Lincoln]
  ...
\end{citequote}
```

The final example shows how to save the value of an argument to use in *enddefn*, in this case in a box (see Section 20.8 [\sbox], page 121):

```
\newsavebox{\quoteauthor}
```

```
\newenvironment{citequote}[1][Shakespeare]{%
  \sbox\quoteauthor{#1}%
  \begin{quotation}
}{%
  \hspace{1em plus 1fill}---\usebox{\quoteauthor}
  \end{quotation}
}
```

12.7 \newtheorem

Define a new theorem-like environment. Synopses:

```
\newtheorem{name}{title}
\newtheorem{name}{title}[numbered_within]
\newtheorem{name}[numbered_like]{title}
```

Using the first form, `\newtheorem{name}{title}` creates an environment that will be labelled with *title*. See the first example below.

The second form `\newtheorem{name}{title}[numbered_within]` creates an environment whose counter is subordinate to the existing counter *numbered_within* (its counter will be reset when *numbered_within* is reset).

The third form `\newtheorem{name}[numbered_like]{title}`, with optional argument between the two required arguments, will create an environment whose counter will share the previously defined counter *numbered_like*.

You can specify one of *numbered_within* and *numbered_like*, or neither, but not both.

This command creates a counter named *name*. In addition, unless the optional argument *numbered_like* is used, inside of the theorem-like environment the current `\ref` value will be that of `\thenumbered_within` (see Section 7.3 [\ref], page 36).

This declaration is global. It is fragile (see Section 12.9 [\protect], page 84).

Arguments:

name The name of the environment. It must not begin with a backslash ('\'). It must not be the name of an existing environment; indeed, the command name `\name` must not already be defined as anything.

title The text printed at the beginning of the environment, before the number. For example, '`Theorem`'.

numbered_within
Optional; the name of an already defined counter, usually a sectional unit such as `chapter` or `section`. When the *numbered_within* counter is reset then the *name* environment's counter will also be reset.

If this optional argument is not used then the command `\thename` is set to `\arabic{name}`.

numbered_like
Optional; the name of an already defined theorem-like environment. The new environment will be numbered in sequence with *numbered_like*.

Without any optional arguments the environments are numbered sequentially. The example below has a declaration in the preamble that results in '`Definition 1`' and '`Definition 2`' in the output.

```
\newtheorem{defn}{Definition}
\begin{document}
\section{...}
\begin{defn}
  First def
\end{defn}

\section{...}
\begin{defn}
  Second def
\end{defn}
```

Because the next example specifies the optional argument *numbered_within* to `\newtheorem` as `section`, the example, with the same document body, gives '`Definition 1.1`' and '`Definition 2.1`'.

```
\newtheorem{defn}{Definition}[section]
\begin{document}
\section{...}
\begin{defn}
  First def
\end{defn}

\section{...}
\begin{defn}
  Second def
\end{defn}
```

In the next example there are two declarations in the preamble, the second of which calls for the new `thm` environment to use the same counter as `defn`. It gives '`Definition 1.1`', followed by '`Theorem 2.1`' and '`Definition 2.2`'.

```
\newtheorem{defn}{Definition}[section]
\newtheorem{thm}[defn]{Theorem}
\begin{document}
\section{...}
\begin{defn}
  First def
\end{defn}

\section{...}
\begin{thm}
  First thm
\end{thm}

\begin{defn}
  Second def
```

```
\end{defn}
```

12.8 \newfont: Define a new font (obsolete)

`\newfont`, now obsolete, defines a command that will switch fonts. Synopsis:

```
\newfont{\cmd}{font description}
```

This defines a control sequence *`\cmd`* that will change the current font. LaTeX will look on your system for a file named *`fontname`*`.tfm`. The control sequence must must not already be defined. It must begin with a backslash ('\').

This command is obsolete. It is a low-level command for setting up an individual font. Today fonts are almost always defined in families (which allows you to, for example, associate a boldface with a roman) through the so-called "New Font Selection Scheme", either by using `.fd` files or through the use of an engine that can access system fonts such as XeLaTeX (see Section 2.3 [TeX engines], page 4).

But since it is part of LaTeX, here is an explanation: the *font description* consists of a *fontname* and an optional *at clause*; this can have the form either `at` *`dimen`* or `scaled` *`factor`*, where a *factor* of '`1000`' means no scaling. For LaTeX's purposes, all this does is scale all the character and other font dimensions relative to the font's design size, which is a value defined in the `.tfm` file.

This example defines two equivalent fonts and typesets a few characters in each:

```
\newfont{\testfontat}{cmb10 at 11pt}
\newfont{\testfontscaled}{cmb10 scaled 1100}
\testfontat abc
\testfontscaled abc
```

12.9 \protect

All LaTeX commands are either *fragile* or *robust*. A fragile command can break when it is used in the argument to certain other commands. Commands that contain data that LaTeX writes to an auxiliary file and re-reads later are fragile. This includes material that goes into a table of contents, list of figures, list of tables, etc. Fragile commands also include line breaks, any command that has an optional argument, and many more. To prevent such commands from breaking, one solution is to preceded them with the command `\protect`.

For example, when LaTeX runs the `\section{`*`section name`*`}` command it writes the *section name* text to the `.aux` auxiliary file, moving it there for use elsewhere in the document such as in the table of contents. Any argument that is internally expanded by LaTeX without typesetting it directly is referred to as a *moving argument*. A command is fragile if it can expand during this process into invalid TeX code. Some examples of moving arguments are those that appear in the `\caption{...}` command (see Section 8.10 [figure], page 44), in the `\thanks{...}` command (see Section 18.1 [\maketitle], page 112), and in @-expressions in the `tabular` and `array` environments (see Section 8.23 [tabular], page 62).

If you get strange errors from commands used in moving arguments, try preceding it with `\protect`. Every fragile commands must be protected with their own `\protect`.

Although usually a `\protect` command doesn't hurt, length commands are robust and should not be preceded by a `\protect` command. Nor can a `\protect` command be used in the argument to `\addtocounter` or `\setcounter` command.

In this example the `\caption` command gives a mysterious error about an extra curly brace. Fix the problem by preceding each `\raisebox` command with `\protect`.

```
\begin{figure}
  ...
  \caption{Company headquarters of A\raisebox{1pt}{B}\raisebox{-1pt}{C}}
\end{figure}
```

In the next example the `\tableofcontents` command gives an error because the `\(..\)` in the section title expands to illegal TEX in the `.toc` file. You can solve this by changing `\(..\)` to `\protect\(..\protect\)`.

```
\begin{document}
\tableofcontents
...
\section{Einstein's \( e=mc^2 \)}
...
```

12.10 \ignorespaces & \ignorespacesafterend

Synopsis:

```
\ignorespaces
```

or

```
\ignorespacesafterend
```

Both commands cause LATEX to ignore spaces after the end of the command up until the first non-space character. The first is a command from Plain TEX, and the second is LATEX-specific.

The `ignorespaces` is often used when defining commands via `\newcommand`, or `\newenvironment`, or `\def`. The example below illustrates. It allows a user to show the points values for quiz questions in the margin but it is inconvenient because, as shown in the `enumerate` list, users must not put any space between the command and the question text.

```
\newcommand{\points}[1]{\makebox[0pt]{\makebox[10em][l]{#1~pts}}
\begin{enumerate}
  \item\points{10}no extra space output here
  \item\points{15} extra space output between the number and the word 'extra'
\end{enumerate}
```

The solution is to change to `\newcommand{\points}[1]{\makebox[0pt]{\makebox[10em][l]{#1~pts}}\i`

A second example shows spaces being removed from the front of text. The commands below allow a user to uniformly attach a title to names. But, as given, if a title accidentally starts with a space then `\fullname` will reproduce that.

```
\makeatletter
\newcommand{\honorific}[1]{\def\@honorific{#1}} % remember title
\newcommand{\fullname}[1]{\@honorific~#1}       % recall title; put before name
\makeatother
\begin{tabular}{|l|}
\honorific{Mr/Ms}  \fullname{Jones} \\  % no extra space here
```

```
\honorific{ Mr/Ms} \fullname{Jones}     % extra space before title
\end{tabular}
```

To fix this, change to `\newcommand{\fullname}[1]{\ignorespaces\@honorific~#1}`.

The `\ignorespaces` is also often used in a `\newenvironment` at the end of the *begin* clause, that is, as part of the second argument, as in `\begin{newenvironment}{`*env name*`}{... \ignorespaces}{...}`.

To strip spaces off the end of an environment use `\ignorespacesafterend`. An example is that this will show a much larger vertical space between the first and second environments than between the second and third.

```
\newenvironment{eq}{\begin{equation}}{\end{equation}}
\begin{eq}
e=mc^2
\end{eq}
\begin{equation}
F=ma
\end{equation}
\begin{equation}
E=IR
\end{equation}
```

Putting a comment character `%` immediately after the `\end{eq}` will make the vertical space disappear, but that is inconvenient. The solution is to change to `\newenvironment{eq}{\begin{equation}}{\end{equation}\ignorespacesafterend}`.

13 Counters

Everything LaTeX numbers for you has a counter associated with it. The name of the counter is often the same as the name of the environment or command associated with the number, except that the counter's name has no backslash \. Thus, associated with the `\chapter` command is the `chapter` counter that keeps track of the chapter number.

Below is a list of the counters used in LaTeX's standard document classes to control numbering.

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

The `mpfootnote` counter is used by the `\footnote` command inside of a minipage (see Section 8.18 [minipage], page 54). The counters `enumi` through `enumiv` are used in the `enumerate` environment, for up to four levels of nesting (see Section 8.7 [enumerate], page 42).

New counters are created with `\newcounter`. See Section 12.3 [\newcounter], page 79.

13.1 \alph \Alph \arabic \roman \Roman \fnsymbol: Printing counters

Print the value of a counter, in a specified style. For instance, if the counter *counter* has the value 1 then a `\alph{`*`counter`*`}` in your source will result in a lower case letter a appearing in the output.

All of these commands take a single counter as an argument, for instance, `\alph{enumi}`. Note that the counter name does not start with a backslash.

`\alph{`*`counter`*`}`
: Print the value of *counter* in lowercase letters: 'a', 'b', ...

`\Alph{`*`counter`*`}`
: Print in uppercase letters: 'A', 'B', ...

`\arabic{`*`counter`*`}`
: Print in Arabic numbers: '1', '2', ...

`\roman{`*`counter`*`}`
: Print in lowercase roman numerals: 'i', 'ii', ...

`\Roman{`*`counter`*`}`
: Print in uppercase roman numerals: 'I', 'II', ...

`\fnsymbol{`*`counter`*`}`
: Prints the value of *counter* in a specific sequence of nine symbols (conventionally used for labeling footnotes). The value of *counter* must be between 1 and 9, inclusive.

  Here are the symbols:

Name	Command	Symbol
asterisk	`\ast`	*
dagger	`\dagger`	†
ddagger	`\ddagger`	‡
section-sign	`\S`	§
paragraph-sign	`\P`	¶
double-vert	`\parallel`	‖
double-asterisk	`\ast\ast`	**
double-dagger	`\dagger\dagger`	††
double-ddagger	`\ddagger\ddagger`	‡‡

13.2 `\usecounter{`*counter*`}`

Synopsis:

```
\usecounter{counter}
```

In the `list` environment, when used in the second argument, this command sets up *counter* to number the list items. It initializes *counter* to zero, and arranges that when `\item` is called without its optional argument then *counter* is incremented by `\refstepcounter`, making its value be the current `ref` value. This command is fragile (see Section 12.9 [\protect], page 84).

Put in the preamble, this makes a new list environment enumerated with *testcounter*:

```
\newcounter{testcounter}
\newenvironment{test}{%
  \begin{list}{}{%
    \usecounter{testcounter}
  }
}{%
  \end{list}
}
```

13.3 `\value{`*counter*`}`

Synopsis:

```
\value{counter}
```

This command expands to the value of *counter*. It is often used in `\setcounter` or `\addtocounter`, but `\value` can be used anywhere that LaTeX expects a number. It must not be preceded by `\protect` (see Section 12.9 [\protect], page 84).

The `\value` command is not used for typesetting the value of the counter. See Section 13.1 [\alph \Alph \arabic \roman \Roman \fnsymbol], page 87.

This example outputs '`Test counter is 6. Other counter is 5.`'.

```
\newcounter{test} \setcounter{test}{5}
\newcounter{other} \setcounter{other}{\value{test}}
\addtocounter{test}{1}

Test counter is \arabic{test}.
Other counter is \arabic{other}.
```

This example inserts `\hspace{4\parindent}`.

```
\setcounter{myctr}{3} \addtocounter{myctr}{1}
\hspace{\value{myctr}\parindent}
```

13.4 \setcounter{*counter*}{*value*}

Synopsis:

```
\setcounter{counter}{value}
```

The `\setcounter` command globally sets the value of *counter* to the *value* argument. Note that the counter name does not start with a backslash.

In this example the section value appears as '`V`'.

```
\setcounter{section}{5}
Here it is in Roman: \Roman{section}.
```

13.5 \addtocounter{*counter*}{*value*}

The `\addtocounter` command globally increments *counter* by the amount specified by the *value* argument, which may be negative.

In this example the section value appears as '`VII`'.

```
\setcounter{section}{5}
\addtocounter{section}{2}
Here it is in Roman: \Roman{section}.
```

13.6 \refstepcounter{*counter*}

The `\refstepcounter` command works in the same way as `\stepcounter` (see Section 13.7 [\stepcounter], page 89): it globally increments the value of *counter* by one and resets the value of any counter numbered within it. (For the definition of "counters numbered within", see Section 12.3 [\newcounter], page 79.)

In addition, this command also defines the current `\ref` value to be the result of `\thecounter`.

While the counter value is set globally, the `\ref` value is set locally, i.e., inside the current group.

13.7 \stepcounter{*counter*}

The `\stepcounter` command globally adds one to *counter* and resets all counters numbered within it. (For the definition of "counters numbered within", see Section 12.3 [\newcounter], page 79.)

13.8 \day \month \year: Predefined counters

LaTeX defines counters for the day of the month (`\day`, 1–31), month of the year (`\month`, 1–12), and year (`\year`, Common Era). When TeX starts up, they are set to the current values on the system where TeX is running. They are not updated as the job progresses.

The related command `\today` produces a string representing the current day (see Section 23.8 [\today], page 143).

14 Lengths

A *length* is a measure of distance. Many LaTeX commands take a length as an argument.

Lengths come in two types. A *rigid length* (what Plain TeX calls a *dimen*) such as `10pt` cannot contain a `plus` or `minus` component. A *rubber length* (what Plain TeX calls a *skip*) can contain those, as with `1cm plus0.05cm minus0.01cm`. These give the ability to stretch or shrink; the length in the prior sentence could appear in the output as long as 1.05 cm or as short as 0.99 cm, depending on what TeX's typesetting algorithm finds optimum.

The `plus` or `minus` component of a rubber length can contain a *fill* component, as in `1in plus2fill`. This gives the length infinite stretchability or shrinkability, so that the length in the prior sentence can be set by TeX to any distance greater than or equal to 1 inch. TeX actually provides three infinite glue components `fil`, `fill`, and `filll`, such that the later ones overcome the earlier ones, but only the middle value is ordinarily used. See Section 19.2 [\hfill], page 114, See Section 19.10 [\vfill], page 117.

Multiplying an entire rubber length by a number turns it into a rigid length, so that after `\setlength{\ylength}{1in plus 0.2in}` and `\setlength{\zlength}{3\ylength}` then the value of `\zlength` is `3in`.

14.1 Units of length

TeX and LaTeX know about these units both inside and outside of math mode.

`pt` Point 1/72.27 inch. The conversion to metric units, to two decimal places, is 1 point = 2.85 mm = 28.45 cm.

`pc` Pica, 12 pt

`in` Inch, 72.27 pt

`bp` Big point, 1/72 inch. This length is the definition of a point in PostScript and many desktop publishing systems.

`cm` Centimeter

`mm` Millimeter

`dd` Didot point, 1.07 pt

`cc` Cicero, 12 dd

`sp` Scaled point, 1/65536 pt

Two other lengths that are often used are values set by the designer of the font. The x-height of the current font *ex*, traditionally the height of the lower case letter x, is often used for vertical lengths. Similarly *em*, traditionally the width of the capital letter M, is often used for horizontal lengths (there is also `\enspace`, which is `0.5em`). Use of these can help make a definition work better across font changes. For example, a definition of the vertical space between list items given as `\setlength{\itemsep}{1ex plus 0.05ex minus 0.01ex}` is more likely to still be reasonable if the font is changed than a definition given in points.

In math mode, many definitions are expressed in terms of the math unit *mu* given by 1 em = 18 mu, where the em is taken from the current math symbols family. See Section 16.5 [Spacing in math mode], page 107.

14.2 \setlength

Synopsis:

```
\setlength{\len}{amount}
```

The `\setlength` sets the value of *length command* `\len` to the *value* argument which can be expressed in any units that LaTeX understands, i.e., inches (`in`), millimeters (`mm`), points (`pt`), big points (`bp`), etc.

14.3 \addtolength

Synopsis:

```
\addtolength{\len}{amount}
```

The `\addtolength` command increments a length command `\len` by the amount specified in the *amount* argument, which may be negative.

14.4 \settodepth

Synopsis:

```
\settodepth{\len}{text}
```

The `\settodepth` command sets the value of a length command `\len` equal to the depth of the *text* argument.

14.5 \settoheight

Synopsis:

```
\settoheight{\len}{text}
```

The `\settoheight` command sets the value of a length command `\len` equal to the height of the `text` argument.

14.6 \settowidth{\len}{text}

Synopsis:

```
\settowidth{\len}{text}
```

The `\settowidth` command sets the value of the command *len* to the width of the *text* argument.

14.7 Predefined lengths

`\width`

`\height`

`\depth`

`\totalheight`

These length parameters can be used in the arguments of the box-making commands (see Chapter 20 [Boxes], page 119). They specify the natural width, etc., of the text in the box. `\totalheight` equals `\height` + `\depth`. To make a box with the text stretched to double the natural size, e.g., say

```
\makebox[2\width]{Get a stretcher}
```

15 Making paragraphs

A paragraph is ended by one or more completely blank lines—lines not containing even a `%`. A blank line should not appear where a new paragraph cannot be started, such as in math mode or in the argument of a sectioning command.

15.1 \indent

`\indent` produces a horizontal space whose width equals to the `\parindent` length, the normal paragraph indentation. It is used to add paragraph indentation where it would otherwise be suppressed.

The default value for `\parindent` is `1em` in two-column mode, otherwise `15pt` for `10pt` documents, `17pt` for `11pt`, and `1.5em` for `12pt`.

15.2 \noindent

When used at the beginning of the paragraph, this command suppresses any paragraph indentation, as in this example.

```
... end of the prior paragraph.

\noindent This paragraph is not indented.
```

It has no effect when used in the middle of a paragraph.

To eliminate paragraph indentation in an entire document, put `\setlength{\parindent}{0pt}` in the preamble.

15.3 \parskip

`\parskip` is a rubber length defining extra vertical space added before each paragraph. The default is `0pt plus1pt`.

15.4 Marginal notes

Synopsis:

```
\marginpar[left]{right}
```

The `\marginpar` command creates a note in the margin. The first line of the note will have the same baseline as the line in the text where the `\marginpar` occurs.

When you only specify the mandatory argument *right*, the text will be placed

- in the right margin for one-sided layout (option `oneside`, see Section 3.1 [Document class options], page 8);
- in the outside margin for two-sided layout (option `twoside`, see Section 3.1 [Document class options], page 8);
- in the nearest margin for two-column layout (option `twocolumn`, see Section 3.1 [Document class options], page 8).

The command `\reversemarginpar` places subsequent marginal notes in the opposite (inside) margin. `\normalmarginpar` places them in the default position.

When you specify both arguments, *left* is used for the left margin, and *right* is used for the right margin.

The first word will normally not be hyphenated; you can enable hyphenation there by beginning the node with `\hspace{0pt}`.

These parameters affect the formatting of the note:

`\marginparpush`
: Minimum vertical space between notes; default '`7pt`' for '`12pt`' documents, '`5pt`' else.

`\marginparsep`
: Horizontal space between the main text and the note; default '`11pt`' for '`10pt`' documents, '`10pt`' else.

`\marginparwidth`
: Width of the note itself; default for a one-sided '`10pt`' document is '`90pt`', '`83pt`' for '`11pt`', and '`68pt`' for '`12pt`'; '`17pt`' more in each case for a two-sided document. In two column mode, the default is '`48pt`'.

The standard LaTeX routine for marginal notes does not prevent notes from falling off the bottom of the page.

16 Math formulas

There are three environments that put LaTeX in math mode:

`math` For formulas that appear right in the text.

`displaymath`
For formulas that appear on their own line.

`equation` The same as the displaymath environment except that it adds an equation number in the right margin.

The `math` environment can be used in both paragraph and LR mode, but the `displaymath` and `equation` environments can be used only in paragraph mode. The `math` and `displaymath` environments are used so often that they have the following short forms:

`\(...\)`	instead of	`\begin{math}...\end{math}`
`\[...\]`	instead of	`\begin{displaymath}...\end{displaymath}`

In fact, the `math` environment is so common that it has an even shorter form:

`$ ... $`	instead of	`\(...\)`

The `\boldmath` command changes math letters and symbols to be in a bold font. It is used *outside* of math mode. Conversely, the `\unboldmath` command changes math glyphs to be in a normal font; it too is used *outside* of math mode.

The `\displaystyle` declaration forces the size and style of the formula to be that of `displaymath`, e.g., with limits above and below summations. For example:

```
$\displaystyle \sum_{n=0}^\infty x_n $
```

16.1 Subscripts & superscripts

In math mode, use the caret character `^` to make the *exp* appear as a superscript: `^{exp}`. Similarly, in math mode, underscore `_{exp}` makes a subscript out of *exp*.

In this example the `0` and `1` appear as subscripts while the `2` is a superscript.

```
\( (x_0+x_1)^2 \)
```

To have more than one character in *exp* use curly braces as in `e^{-2x}`.

LaTeX handles superscripts on superscripts, and all of that stuff, in the natural way, so expressions such as `e^{x^2}` and `x_{a_0}` will look right. It also does the right thing when something has both a subscript and a superscript. In this example the `0` appears at the bottom of the integral sign while the `10` appears at the top.

```
\int_0^{10} x^2 \,dx
```

You can put a superscript or subscript before a symbol with a construct such as `{}_t K^2` in math mode (the initial `{}` prevents the prefixed subscript from being attached to any prior symbols in the expression).

Outside of math mode, a construct like `A test$_\textnormal{subscript}$` will produce a subscript typeset in text mode, not math mode. Note that there are packages specialized for writing Chemical formulas such as `mhchem`.

16.2 Math symbols

LaTeX provides almost any mathematical symbol you're likely to need. For example, if you include `$\pi$` in your source, you will get the pi symbol π.

Below is a list of commonly-available symbols. It is by no means an exhaustive list. Each symbol here is described with a short phrase, and its symbol class (which determines the spacing around it) is given in parenthesis. Unless said otherwise, the commands for these symbols can be used only in math mode.

To redefine a command so that it can be used whatever the current mode, see Section 17.1 [\ensuremath], page 110.

`\|` $\|$ Parallel (relation). Synonym: `\parallel`.

`\aleph` $\aleph$ Aleph, transfinite cardinal (ordinary).

`\alpha` α Lower case Greek letter alpha (ordinary).

`\amalg` $\amalg$ Disjoint union (binary)

`\angle` $\angle$ Geometric angle (ordinary). Similar: less-than sign `<` and angle bracket `\langle`.

`\approx` $\approx$ Almost equal to (relation).

`\ast` $*$ Asterisk operator, convolution, six-pointed (binary). Synonym: `*`, which is often a superscript or subscript, as in the Kleene star. Similar: `\star`, which is five-pointed, and is sometimes used as a general binary operation, and sometimes reserved for cross-correlation.

`\asymp` $\asymp$ Asymptotically equivalent (relation).

`\backslash`
: $\backslash$ Backslash (ordinary). Similar: set minus `\setminus`, and `\textbackslash` for backslash outside of math mode.

`\beta` β Lower case Greek letter beta (ordinary).

`\bigcap` $\bigcap$ Variable-sized, or n-ary, intersection (operator). Similar: binary intersection `\cap`.

`\bigcirc` $\bigcirc$ Circle, larger (binary). Similar: function composition `\circ`.

`\bigcup` $\bigcup$ Variable-sized, or n-ary, union (operator). Similar: binary union `\cup`.

`\bigodot` $\bigodot$ Variable-sized, or n-ary, circled dot operator (operator).

`\bigoplus`
: $\bigoplus$ Variable-sized, or n-ary, circled plus operator (operator).

`\bigotimes`
: $\bigotimes$ Variable-sized, or n-ary, circled times operator (operator).

`\bigtriangledown`
: $\bigtriangledown$ Variable-sized, or n-ary, open triangle pointing down (operator).

`\bigtriangleup`
: $\bigtriangleup$ Variable-sized, or n-ary, open triangle pointing up (operator).

`\bigsqcup`
: ⨆ Variable-sized, or n-ary, square union (operator).

`\biguplus`
: ⨄ Variable-sized, or n-ary, union operator with a plus (operator). (Note that the name has only one p.)

`\bigvee` ⋁ Variable-sized, or n-ary, logical-and (operator).

`\bigwedge`
: ⋀ Variable-sized, or n-ary, logical-or (operator).

`\bot` `\bot` Up tack, bottom, least element of a partially ordered set, or a contradiction (ordinary). See also `\top`.

`\bowtie` ⋈ Natural join of two relations (relation).

`\Box` `\Box` Modal operator for necessity; square open box (ordinary). Not available in plain TEX. In LATEX you need to load the `amssymb` package.

`\bullet` • Bullet (binary). Similar: multiplication dot `\cdot`.

`\cap` ∩ Intersection of two sets (binary). Similar: variable-sized operator `\bigcap`.

`\cdot` · Multiplication (binary). Similar: Bullet dot `\bullet`.

`\chi` χ Lower case Greek chi (ordinary).

`\circ` ∘ Function composition, ring operator (binary). Similar: variable-sized operator `\bigcirc`.

`\clubsuit`
: ♣ Club card suit (ordinary).

`\complement`
: `\complement` Set complement, used as a superscript as in `$S^\complement$` (ordinary). Not available in plain TEX. In LATEX you need to load the `amssymb` package. Also used: `$S^{\mathsf{c}}$` or `$\bar{S}$`.

`\cong` ≅ Congruent (relation).

`\coprod` ∐ Coproduct (operator).

`\cup` ∪ Union of two sets (binary). Similar: variable-sized operator `\bigcup`.

`\dagger` † Dagger relation (binary).

`\dashv` ⊣ Dash with vertical, reversed turnstile (relation). Similar: turnstile `\vdash`.

`\ddagger` ‡ Double dagger relation (binary).

`\Delta` Δ Greek upper case delta, used for increment (ordinary).

`\delta` δ Greek lower case delta (ordinary).

`\Diamond` `\Diamond` Large diamond operator (ordinary). Not available in plain TEX. In LATEX you need to load the `amssymb` package.

`\diamond` ⋄ Diamond operator, or diamond bullet (binary). Similar: large diamond `\Diamond`, circle bullet `\bullet`.

`\diamondsuit`
: $\diamondsuit$ Diamond card suit (ordinary).

`\div` $\div$ Division sign (binary).

`\doteq` $\doteq$ Approaches the limit (relation). Similar: geometrically equal to `\Doteq`.

`\downarrow`
: $\downarrow$ Down arrow, converges (relation). Similar: double line down arrow `\Downarrow`.

`\Downarrow`
: $\Downarrow$ Double line down arrow (relation). Similar: single line down arrow `\downarrow`.

`\ell` ℓ Lowercase cursive letter l (ordinary).

`\emptyset`
: $\emptyset$ Empty set symbol (ordinary). The variant form is `\varnothing`.

`\epsilon` ϵ Lower case lunate epsilon (ordinary). Similar to Greek text letter. More widely used in mathematics is the script small letter epsilon `\varepsilon` ε. Related: the set membership relation `\in` $\in$.

`\equiv` $\equiv$ Equivalence (relation).

`\eta` η Lower case Greek letter (ordinary).

`\exists` $\exists$ Existential quantifier (ordinary).

`\flat` $\flat$ Musical flat (ordinary).

`\forall` $\forall$ Universal quantifier (ordinary).

`\frown` $\frown$ Downward curving arc (ordinary).

`\Gamma` Γ Upper case Greek letter (ordinary).

`\gamma` γ Lower case Greek letter (ordinary).

`\ge` $\geq$ Greater than or equal to (relation). This is a synonym for `\geq`.

`\geq` $\geq$ Greater than or equal to (relation). This is a synonym for `\ge`.

`\gets` $\leftarrow$ Is assigned the value (relation). Synonym: `\leftarrow`.

`\gg` $\gg$ Much greater than (relation). Similar: much less than `\ll`.

`\hbar` $\hbar$ Planck constant over two pi (ordinary).

`\heartsuit`
: $\heartsuit$ Heart card suit (ordinary).

`\hookleftarrow`
: $\hookleftarrow$ Hooked left arrow (relation).

`\hookrightarrow`
: $\hookrightarrow$ Hooked right arrow (relation).

`\iff` $\iff$ If and only if (relation). It is `\Longleftrightarrow` with a `\thickmuskip` on either side.

`\Im` $\Im$ Imaginary part (ordinary). See: real part `\Re`.

`\in` $\in$ Set element (relation). See also: lower case lunate epsilon `\epsilon`ϵ and small letter script epsilon `\varepsilon`.

`\infty` ∞ Infinity (ordinary).

`\int` $\int$ Integral (operator).

`\iota` ι Lower case Greek letter (ordinary).

`\Join` `\Join` Condensed bowtie symbol (relation). Not available in Plain TEX.

`\kappa` κ Lower case Greek letter (ordinary).

`\Lambda` Λ Upper case Greek letter (ordinary).

`\lambda` λ Lower case Greek letter (ordinary).

`\land` $\land$ Logical and (binary). This is a synonym for `\wedge`. See also logical or `\lor`.

`\langle` $\langle$ Left angle, or sequence, bracket (opening). Similar: less-than `<`. Matches `\rangle`.

`\lbrace` { Left curly brace (opening). Synonym: `\{`. Matches `\rbrace`.

`\lbrack` [Left square bracket (opening). Synonym: `[`. Matches `\rbrack`.

`\lceil` $\lceil$ Left ceiling bracket, like a square bracket but with the bottom shaved off (opening). Matches `\rceil`.

`\le` $\leq$ Less than or equal to (relation). This is a synonym for `\leq`.

`\leadsto` `\leadsto` Squiggly right arrow (relation). Not available in plain TEX. In LATEX you need to load the `amssymb` package. To get this symbol outside of math mode you can put `\newcommand*{\Leadsto}{\ensuremath{\leadsto}}` in the preamble and then use `\Leadsto` instead.

`\Leftarrow` $\Leftarrow$ Is implied by, double-line left arrow (relation). Similar: single-line left arrow `\leftarrow`.

`\leftarrow` $\leftarrow$ Single-line left arrow (relation). Synonym: `\gets`. Similar: double-line left arrow `\Leftarrow`.

`\leftharpoondown` $\leftharpoondown$ Single-line left harpoon, barb under bar (relation).

`\leftharpoonup` $\leftharpoonup$ Single-line left harpoon, barb over bar (relation).

`\Leftrightarrow` $\Leftrightarrow$ Bi-implication; double-line double-headed arrow (relation). Similar: single-line double headed arrow `\leftrightarrow`.

`\leftrightarrow` $\leftrightarrow$ Single-line double-headed arrow (relation). Similar: double-line double headed arrow `\Leftrightarrow`.

`\leq` $\leq$ Less than or equal to (relation). This is a synonym for `\le`.

`\lfloor` $\lfloor$ Left floor bracket (opening). Matches: `\floor`.

`\lhd` `\lhd` Arrowhead, that is, triangle, pointing left (binary). Not available in plain TEX. In LATEX you need to load the `amssymb` package. For the normal subgroup symbol you should load `amssymb` and use `\vartriangleleft` (which is a relation and so gives better spacing).

`\ll` $\ll$ Much less than (relation). Similar: much greater than `\gg`.

`\lnot` $\neg$ Logical negation (ordinary). Synonym: `\neg`.

`\longleftarrow`
$\longleftarrow$ Long single-line left arrow (relation). Similar: long double-line left arrow `\Longleftarrow`.

`\longleftrightarrow`
$\longleftrightarrow$ Long single-line double-headed arrow (relation). Similar: long double-line double-headed arrow `\Longleftrightarrow`.

`\longmapsto`
$\longmapsto$ Long single-line left arrow starting with vertical bar (relation). Similar: shorter version `\mapsto`.

`\longrightarrow`
$\longrightarrow$ Long single-line right arrow (relation). Similar: long double-line right arrow `\Longrightarrow`.

`\lor` $\vee$ Logical or (binary). Synonym: wedge `\wedge`.

`\mapsto` $\mapsto$ Single-line left arrow starting with vertical bar (relation). Similar: longer version `\longmapsto`.

`\mho` `\mho` Conductance, half-circle rotated capital omega (ordinary). Not available in plain TEX. In LATEX you need to load the `amssymb` package.

`\mid` | Single-line vertical bar (relation). A typical use of `\mid` is for a set `\{\, x \mid x\geq 5 \,\}`.

Similar: `\vert` and `|` produce the same single-line vertical bar symbol but without any spacing (they fall in class ordinary) and you should not use them as relations but instead only as ordinals, i.e., footnote symbols. For absolute value, see the entry for `\vert` and for norm see the entry for `\Vert`.

`\models` $\models$ Entails, or satisfies; double turnstile, short double dash (relation). Similar: long double dash `\vDash`.

`\mp` $\mp$ Minus or plus (relation).

`\mu` μ Lower case Greek letter (ordinary).

`\nabla` ∇ Hamilton's del, or differential, operator (ordinary).

`\natural` $\natural$ Musical natural notation (ordinary).

`\ne` $\neq$ Not equal (relation). Synonym: `\neq`.

`\nearrow` $\nearrow$ North-east arrow (relation).

`\neg` $\neg$ Logical negation (ordinary). Synonym: `\lnot`. Sometimes instead used for negation: `\sim`.

`\neq` $\neq$ Not equal (relation). Synonym: `\ne`.

`\ni` $\ni$ Reflected membership epsilon; has the member (relation). Synonym: `\owns`. Similar: is a member of `\in`.

`\not` $\not$ Long solidus, or slash, used to overstrike a following operator (relation).

Many negated operators that don't require `\not` are available, particularly with the `amssymb` package. For example, `\notin` is probably typographically preferable to `\not\in`.

`\notin` $\notin$ Not an element of (relation). Similar: not subset of `\nsubseteq`.

`\nu` ν Lower case Greek letter (ordinary).

`\nwarrow` $\nwarrow$ North-west arrow (relation).

`\odot` $\odot$ Dot inside a circle (binary). Similar: variable-sized operator `\bigodot`.

`\oint` $\oint$ Contour integral, integral with circle in the middle (operator).

`\Omega` Ω Upper case Greek letter (ordinary).

`\omega` ω Lower case Greek letter (ordinary).

`\ominus` $\ominus$ Minus sign, or dash, inside a circle (binary).

`\oplus` $\oplus$ Plus sign inside a circle (binary). Similar: variable-sized operator `\bigoplus`.

`\oslash` $\oslash$ Solidus, or slash, inside a circle (binary).

`\otimes` $\otimes$ Times sign, or cross, inside a circle (binary). Similar: variable-sized operator `\bigotimes`.

`\owns` $\ni$ Reflected membership epsilon; has the member (relation). Synonym: `\ni`. Similar: is a member of `\in`.

`\parallel`

$\parallel$ Parallel (relation). Synonym: `\|`.

`\partial` ∂ Partial differential (ordinary).

`\perp` $\perp$ Perpendicular (relation). Similar: `\bot` uses the same glyph but the spacing is different because it is in the class ordinary.

`\phi` ϕ Lower case Greek letter (ordinary). The variant form is `\varphi` φ.

`\Pi` Π Upper case Greek letter (ordinary).

`\pi` π Lower case Greek letter (ordinary). The variant form is `\varpi` ϖ.

`\pm` $\pm$ Plus or minus (binary).

`\prec` $\prec$ Precedes (relation). Similar: less than `<`.

`\preceq` $\preceq$ Precedes or equals (relation). Similar: less than or equals `\leq`.

`\prime` ′ Prime, or minute in a time expression (ordinary). Typically used as a superscript: `$f^\prime$`; `$f^\prime$` and `$f'$` produce the same result. An advantage of the second is that `$f'''$` produces the desired symbol, that is, the same result as `$f^{\prime\prime\prime}$`, but uses rather less typing. You can only use `\prime` in math mode. Using the right single quote `'` in text mode produces a different character (apostrophe).

`\prod` ∏ Product (operator).

`\propto` ∝ Is proportional to (relation)

`\Psi` Ψ Upper case Greek letter (ordinary).

`\psi` ψ Lower case Greek letter (ordinary).

`\rangle` ⟩ Right angle, or sequence, bracket (closing). Similar: greater than `>`. Matches:`\langle`.

`\rbrace` } Right curly brace (closing). Synonym: `\}`. Matches `\lbrace`.

`\rbrack`] Right square bracket (closing). Synonym: `]`. Matches `\lbrack`.

`\rceil` ⌉ Right ceiling bracket (closing). Matches `\lceil`.

`\Re` ℜ Real part, real numbers, cursive capital R (ordinary). Related: double-line, or blackboard bold, R `\mathbb{R}`; to access this, load the `amsfonts` package.

`\restriction`
`\restriction` Restriction of a function (relation). Synonym: `\upharpoonright`. Not available in plain TEX. In LATEX you need to load the `amssymb` package.

`\revemptyset`
`\revemptyset` Reversed empty set symbol (ordinary). Related: `\varnothing`. Not available in plain TEX. In LATEX you need to load the `stix` package.

`\rfloor` ⌋ Right floor bracket, a right square bracket with the top cut off (closing). Matches `\lfloor`.

`\rhd` `\rhd` Arrowhead, that is, triangle, pointing right (binary). Not available in plain TEX. In LATEX you need to load the `amssymb` package. For the normal subgroup symbol you should instead load `amssymb` and use `\vartriangleright` (which is a relation and so gives better spacing).

`\rho` ρ Lower case Greek letter (ordinary). The variant form is `\varrho` ϱ.

`\Rightarrow`
⇒ Implies, right-pointing double line arrow (relation). Similar: right single-line arrow `\rightarrow`.

`\rightarrow`
→ Right-pointing single line arrow (relation). Synonym: `\to`. Similar: right double line arrow `\Rightarrow`.

`\rightharpoondown`
⇁ Right-pointing harpoon with barb below the line (relation).

`\rightharpoonup`
⇀ Right-pointing harpoon with barb above the line (relation).

`\rightleftharpoons`
: $\rightleftharpoons$ Right harpoon up above left harpoon down (relation).

`\searrow` $\searrow$ Arrow pointing southeast (relation).

`\setminus`
: $\setminus$ Set difference, reverse solidus or slash, like `\` (binary). Similar: backslash `\backslash` and also `\textbackslash` outside of math mode.

`\sharp` $\sharp$ Musical sharp (ordinary).

`\Sigma` Σ Upper case Greek letter (ordinary).

`\sigma` σ Lower case Greek letter (ordinary). The variant form is `\varsigma` ς.

`\sim` $\sim$ Similar, in a relation (relation).

`\simeq` $\simeq$ Similar or equal to, in a relation (relation).

`\smallint`
: $\smallint$ Integral sign that does not change to a larger size in a display (operator).

`\smile` $\smile$ Upward curving arc, smile (ordinary).

`\spadesuit`
: $\spadesuit$ Spade card suit (ordinary).

`\sqcap` $\sqcap$ Square intersection symbol (binary). Similar: intersection `cap`.

`\sqcup` $\sqcup$ Square union symbol (binary). Similar: union `cup`. Related: variable-sized operator `\bigsqcup`.

`\sqsubset`
: `\sqsubset` Square subset symbol (relation). Similar: subset `\subset`. Not available in plain TEX. In LATEX you need to load the `amssymb` package.

`\sqsubseteq`
: $\sqsubseteq$ Square subset or equal symbol (binary). Similar: subset or equal to `\subseteq`.

`\sqsupset`
: `\sqsupset` Square superset symbol (relation). Similar: superset `\supset`. Not available in plain TEX. In LATEX you need to load the `amssymb` package.

`\sqsupseteq`
: $\sqsupseteq$ Square superset or equal symbol (binary). Similar: superset or equal `\supseteq`.

`\star` $\star$ Five-pointed star, sometimes used as a general binary operation but sometimes reserved for cross-correlation (binary). Similar: the synonyms asterisk `*` and `\ast`, which are six-pointed, and more often appear as a superscript or subscript, as with the Kleene star.

`\subset` $\subset$ Subset (occasionally, is implied by) (relation).

`\subseteq`
: $\subseteq$ Subset or equal to (relation).

`\succ` $\succ$ Comes after, succeeds (relation). Similar: is less than `>`.

`\succeq` $\succeq$ Succeeds or is equal to (relation). Similar: less than or equal to `\leq`.

`\sum` $\sum$ Summation (operator). Similar: Greek capital sigma `\Sigma`.

`\supset` $\supset$ Superset (relation).

`\supseteq`
$\supseteq$ Superset or equal to (relation).

`\surd` $\surd$ Radical symbol (ordinary). The LaTeX command `\sqrt{...}` typesets the square root of the argument, with a bar that extends to cover the argument.

`\swarrow` $\swarrow$ Southwest-pointing arrow (relation).

`\tau` τ Lower case Greek letter (ordinary).

`\theta` θ Lower case Greek letter (ordinary). The variant form is `\vartheta` ϑ.

`\times` $\times$ Primary school multiplication sign (binary). See also `\cdot`.

`\to` $\to$ Right-pointing single line arrow (relation). Synonym: `\rightarrow`.

`\top` `\top` Top, greatest element of a partially ordered set (ordinary). See also `\bot`.

`\triangle`
$\triangle$ Triangle (ordinary).

`\triangleleft`
$\triangleleft$ Not-filled triangle pointing left (binary). Similar: `\lhd`. For the normal subgroup symbol you should load `amssymb` and use `\vartriangleleft` (which is a relation and so gives better spacing).

`\triangleright`
$\triangleright$ Not-filled triangle pointing right (binary). For the normal subgroup symbol you should instead load `amssymb` and use `\vartriangleright` (which is a relation and so gives better spacing).

`\unlhd` `\unlhd` Left-pointing not-filled underlined arrowhead, that is, triangle, with a line under (binary). Not available in plain TeX. In LaTeX you need to load the `amssymb` package. For the normal subgroup symbol load `amssymb` and use `\vartrianglelefteq` (which is a relation and so gives better spacing).

`\unrhd` `\unrhd` Right-pointing not-filled underlined arrowhead, that is, triangle, with a line under (binary). Not available in plain TeX. In LaTeX you need to load the `amssymb` package. For the normal subgroup symbol load `amssymb` and use `\vartrianglerighteq` (which is a relation and so gives better spacing).

`\Uparrow` $\Uparrow$ Double-line upward-pointing arrow (relation). Similar: single-line up-pointing arrow `\uparrow`.

`\uparrow` $\uparrow$ Single-line upward-pointing arrow, diverges (relation). Similar: double-line up-pointing arrow `\Uparrow`.

`\Updownarrow`
$\Updownarrow$ Double-line upward-and-downward-pointing arrow (relation). Similar: single-line upward-and-downward-pointing arrow `\updownarrow`.

`\updownarrow`
: $\updownarrow$ Single-line upward-and-downward-pointing arrow (relation). Similar: double-line upward-and-downward-pointing arrow `\Updownarrow`.

`\upharpoonright`
: `\upharpoonright` Up harpoon, with barb on right side (relation). Synonym: `\restriction`. Not available in plain TEX. In LATEX you need to load the `amssymb` package.

`\uplus` $\uplus$ Multiset union, a union symbol with a plus symbol in the middle (binary). Similar: union `\cup`. Related: variable-sized operator `\biguplus`.

`\Upsilon` Υ Upper case Greek letter (ordinary).

`\upsilon` υ Lower case Greek letter (ordinary).

`\varepsilon`
: ε Small letter script epsilon (ordinary). This is more widely used in mathematics than the non-variant lunate epsilon form `\epsilon` ϵ. Related: set membership `\in`.

`\vanothing`
: `\varnothing` Empty set symbol. Similar: `\emptyset`. Related: `\revemptyset`. Not available in plain TEX. In LATEX you need to load the `amssymb` package.

`\varphi` φ Variant on the lower case Greek letter (ordinary). The non-variant form is `\phi` ϕ.

`\varpi` ϖ Variant on the lower case Greek letter (ordinary). The non-variant form is `\pi` π.

`\varrho` ϱ Variant on the lower case Greek letter (ordinary). The non-variant form is `\rho` ρ.

`\varsigma`
: ς Variant on the lower case Greek letter (ordinary). The non-variant form is `\sigma` σ.

`\vartheta`
: ϑ Variant on the lower case Greek letter (ordinary). The non-variant form is `\theta` θ.

`\vdash` $\vdash$ Provable; turnstile, vertical and a dash (relation). Similar: turnstile rotated a half-circle `\dashv`.

`\vee` $\vee$ Logical or; a downwards v shape (binary). Related: logical and `\wedge`. Similar: variable-sized operator `\bigvee`.

`\Vert` $\Vert$ Vertical double bar (ordinary). Similar: vertical single bar `\vert`.

For a norm symbol, you can use the `mathtools` package and add `\DeclarePairedDelimiter\norm{\lVert}{\rVert}` to your preamble. This gives you three command variants for double-line vertical bars that are correctly horizontally spaced: if in the document body you write the starred version `$\norm*{M^\perp}$` then the height of the vertical bars will match

the height of the argument, whereas with `\norm{M^\perp}` the bars do not grow with the height of the argument but instead are the default height, and `\norm[`*size command*`]{M^\perp}` also gives bars that do not grow but are set to the size given in the *size command*, e.g., `\Bigg`.

`\vert` | Single line vertical bar (ordinary). Similar: double-line vertical bar `\Vert`. For such that, as in the definition of a set, use `\mid` because it is a relation.

For absolute value you can use the `mathtools` package and add `\DeclarePairedDelimiter\abs{\lvert}{\rvert}` to your preamble. This gives you three command variants for single-line vertical bars that are correctly horizontally spaced: if in the document body you write the starred version `$\abs*{\frac{22}{7}}$` then the height of the vertical bars will match the height of the argument, whereas with `\abs{\frac{22}{7}}` the bars do not grow with the height of the argument but instead are the default height, and `\abs[`*size command*`]{\frac{22}{7}}` also gives bars that do not grow but are set to the size given in the *size command*, e.g., `\Bigg`.

`\wedge` $\wedge$ Logical and (binary). Synonym: `\land`. See also logical or `\vee`. Similar: variable-sized operator `\bigwedge`.

`\wp` $\wp$ Weierstrass p (ordinary).

`\wr` $\wr$ Wreath product (binary).

`\Xi` Ξ Upper case Greek letter (ordinary).

`\xi` ξ Lower case Greek letter (ordinary).

`\zeta` ζ Lower case Greek letter (ordinary).

16.3 Math functions

These commands produce roman function names in math mode with proper spacing.

`\arccos` arccos

`\arcsin` arcsin

`\arctan` arctan

`\arg` arg

`\bmod` Binary modulo operator ($x \bmod y$)

`\cos` cos

`\cosh` cosh

`\cot` cot

`\coth` coth

`\csc` csc

`\deg` deg

`\det` det

`\dim` dim

`\exp` exp

`\gcd` gcd

`\hom` hom

`\inf` inf

`\ker` ker

`\lg` lg

`\lim` lim

`\liminf` lim inf

`\limsup` lim sup

`\ln` ln

`\log` log

`\max` max

`\min` min

`\pmod` parenthesized modulus, as in ($\pmod 2^n - 1$)

`\Pr` Pr

`\sec` sec

`\sin` sin

`\sinh` sinh

`\sup` sup

`\tan` tan

`\tanh` tanh

16.4 Math accents

LaTeX provides a variety of commands for producing accented letters in math. These are different from accents in normal text (see Section 23.5 [Accents], page 141).

`\acute` Math acute accent: $\acute{x}$.

`\bar` Math bar-over accent: $\bar{x}$.

`\breve` Math breve accent: $\breve{x}$.

`\check` Math háček (check) accent: $\check{x}$.

`\ddot` Math dieresis accent: $\ddot{x}$.

`\dot` Math dot accent: $\dot{x}$.

`\grave` Math grave accent: $\grave{x}$.

`\hat` Math hat (circumflex) accent: $\hat{x}$.

`\imath` Math dotless i.

`\jmath` Math dotless j.

`\mathring`
Math ring accent: $\mathring{x}$.

`\tilde` Math tilde accent: $\tilde{x}$.

`\vec` Math vector symbol: $\vec{x}$.

`\widehat` Math wide hat accent: $\widehat{x+y}$.

`\widetilde`
Math wide tilde accent: $\widetilde{x+y}$.

16.5 Spacing in math mode

In a `math` environment, LaTeX ignores the spaces that you use in the source, and instead puts in the spacing according to the normal rules for mathematics texts.

Many math mode spacing definitions are expressed in terms of the math unit *mu* given by 1 em = 18 mu, where the em is taken from the current math symbols family (see Section 14.1 [Units of length], page 90). LaTeX provides the following commands for use in math mode:

`\;` Normally `5.0mu plus 5.0mu`. The longer name is `\thickspace`. Math mode only.

`\:`
`\>` Normally `4.0mu plus 2.0mu minus 4.0mu`. The longer name is `\medspace`. Math mode only.

`\,` Normally `3mu`. The longer name is `\thinspace`. This can be used in both math mode and text mode. See Section 19.5 [\thinspace], page 116.

`\!` A negative thin space. Normally `-3mu`. Math mode only.

`\quad` This is 18 mu, that is, 1 em. This is often used for space surrounding equations or expressions, for instance for the space between two equations inside a `displaymath` environment. It is available in both text and math mode.

`\qquad` A length of 2 quads, that is, 36 mu = 2 em. It is available in both text and math mode.

In this example a thinspace separates the function from the infinitesimal.

```
\int_0^1 f(x)\,dx
```

16.6 Math miscellany

`\*` A *discretionary* multiplication symbol, at which a line break is allowed. Without the break multiplication is implicitly indicated by a space, while in the case of a break a × symbol is printed immediately before the break. So

```
\documentclass{article}
```

```
\begin{document}
Now \(A_3 = 0\), hence the product of all terms \(A_1\)
through \(A_4\), that is \(A_1\* A_2\* A_3 \* A_4\), is
equal to zero.
\end{document}
```

will make that sort of output:

> Now $A_3 = 0$, hence the product of all terms A_1 through A_4, that is $A_1 A_2 \times A_3 A_4$, is equal to zero.

`\cdots` A horizontal ellipsis with the dots raised to the center of the line. As in: '$\cdots$'.

`\ddots` A diagonal ellipsis: $\ddots$.

`\frac{`*num*`}{`*den*`}`
: Produces the fraction *num* divided by *den*.

  eg. $\frac{1}{4}$

`\left` *delim1* `...` `\right` *delim2*
: The two delimiters need not match; '`.`' acts as a *null delimiter*, producing no output. The delimiters are sized according to the math in between. Example: `\left( \sum_{i=1}^{10} a_i \right]`.

`\mathdollar`
: Dollar sign in math mode: $.

`\mathellipsis`
: Ellipsis (spaced for text) in math mode: ....

`\mathparagraph`
: Paragraph sign (pilcrow) in math mode: ¶.

`\mathsection`
: Section sign in math mode.

`\mathsterling`
: Sterling sign in math mode: £.

`\mathunderscore`
: Underscore in math mode: _.

`\overbrace{`*math*`}`
: Generates a brace over *math*. For example, `\overbrace{x+\cdots+x}^{k \;\textrm{times}}`. The result looks like: $\overbrace{x+\cdots+x}^{k \text{ times}}$

`\overline{`*text*`}`
: Generates a horizontal line over *tex*. For example, `\overline{x+y}`. The result looks like: $\overline{x+y}$.

`\sqrt[`*root*`]{`*arg*`}`
: Produces the representation of the square root of *arg*. The optional argument *root* determines what root to produce. For example, the cube root of `x+y` would be typed as `\sqrt[3]{x+y}`. The result looks like this: $\sqrt[3]{x+y}$.

`\stackrel{`*text*`}{`*relation*`}`
: Puts *text* above *relation*. For example, `\stackrel{f}{\longrightarrow}`. The result looks like this: $\stackrel{f}{\longrightarrow}$.

`\underbrace{`*math*`}`
: Generates *math* with a brace underneath. For example, `\underbrace{x+y+z}_{>\,0}` The result looks like this: $\underbrace{x+y+z}_{>0}$.

`\underline{`*text*`}`
: Causes *text*, which may be either math mode or not, to be underlined. The line is always below the text, taking account of descenders. The result looks like this: $\underline{xyz}$

`\vdots` Produces a vertical ellipsis. The result looks like this: $\vdots$.

17 Modes

When LaTeX is processing your input text, it is always in one of three modes:

- Paragraph mode
- Math mode
- Left-to-right mode, called LR mode for short

Mode changes occur only when entering or leaving an environment, or when LaTeX is processing the argument of certain text-producing commands.

Paragraph mode is the most common; it's the one LaTeX is in when processing ordinary text. In this mode, LaTeX breaks the input text into lines and breaks the lines into pages.

LaTeX is in *math mode* when it's generating a mathematical formula, either displayed math or within a line.

In *LR mode*, as in paragraph mode, LaTeX considers the output that it produces to be a string of words with spaces between them. However, unlike paragraph mode, LaTeX keeps going from left to right; it never starts a new line in LR mode. Even if you put a hundred words into an `\mbox`, LaTeX would keep typesetting them from left to right inside a single box (and then most likely complain because the resulting box was too wide to fit on the line). LaTeX is in LR mode when it starts making a box with an `\mbox` command. You can get it to enter a different mode inside the box—for example, you can make it enter math mode to put a formula in the box.

There are also several text-producing commands and environments for making a box that put LaTeX into paragraph mode. The box made by one of these commands or environments will be called a `parbox`. When LaTeX is in paragraph mode while making a box, it is said to be in "inner paragraph mode" (no page breaks). Its normal paragraph mode, which it starts out in, is called "outer paragraph mode".

17.1 `\ensuremath`

Synopsis:

```
\ensuremath{formula}
```

The `\ensuremath` command ensures that *formula* is typeset in math mode whatever the current mode in which the command is used.

For instance:

```
\documentclass{report}
\newcommand{\ab}{\ensuremath{(\delta, \varepsilon)}}
\begin{document}
Now, the \ab\ pair is equal to \(\ab = (\frac{1}{\pi}, 0)\), ...
\end{document}
```

One can redefine commands that can be used only in math mode so that they ca be used in any mode like in the following example given for `\leadsto`:

```
\documentclass{report}
\usepackage{amssymb}
\newcommand{\originalMeaningOfLeadsTo}{}
\let\originalMeaningOfLeadsTo\leadsto
```

```
\renewcommand\leadsto{\ensuremath{\originalMeaningOfLeadsTo}}
\begin{document}
All roads \leadsto\ Rome.
\end{document}
```

18 Page styles

The `\documentclass` command determines the size and position of the page's head and foot. The page style determines what goes in them.

18.1 \maketitle

The `\maketitle` command generates a title on a separate title page—except in the `article` class, where the title is placed at the top of the first page. Information used to produce the title is obtained from the following declarations:

`\author{`*name* `\and` *name2*`}`
: The `\author` command declares the document author(s), where the argument is a list of authors separated by `\and` commands. Use `\\` to separate lines within a single author's entry—for example, to give the author's institution or address.

`\date{`*text*`}`
: The `\date` command declares *text* to be the document's date. With no `\date` command, the current date (see Section 23.8 [\today], page 143) is used.

`\thanks{`*text*`}`
: The `\thanks` command produces a `\footnote` to the title, usually used for credit acknowledgements.

`\title{`*text*`}`
: The `\title` command declares *text* to be the title of the document. Use `\\` to force a line break, as usual.

18.2 \pagenumbering

Synopsis:

```
\pagenumbering{style}
```

Specifies the style of page numbers, according to *style*; also resets the page number to 1. The *style* argument is one of the following:

`arabic` arabic numerals

`roman` lowercase Roman numerals

`Roman` uppercase Roman numerals

`alph` lowercase letters

`Alph` uppercase letters

18.3 \pagestyle

Synopsis:

```
\pagestyle{style}
```

The `\pagestyle` command specifies how the headers and footers are typeset from the current page onwards. Values for *style*:

`plain` Just a plain page number.

`empty` Empty headers and footers, e.g., no page numbers.

`headings` Put running headers on each page. The document style specifies what goes in the headers.

`myheadings`
Custom headers, specified via the `\markboth` or the `\markright` commands.

Here are the descriptions of `\markboth` and `\markright`:

`\markboth{`*`left`*`}{`*`right`*`}`
Sets both the left and the right heading. A "left-hand heading" (*left*) is generated by the last `\markboth` command before the end of the page, while a "right-hand heading" (*right*) is generated by the first `\markboth` or `\markright` that comes on the page if there is one, otherwise by the last one before the page.

`\markright{`*`right`*`}`
Sets the right heading, leaving the left heading unchanged.

18.4 `\thispagestyle{`*`style`*`}`

The `\thispagestyle` command works in the same manner as the `\pagestyle` command (see previous section) except that it changes to *style* for the current page only.

19 Spaces

LaTeX has many ways to produce white (or filled) space.

19.1 \hspace

Synopsis:

```
\hspace{length}
\hspace*{length}
```

Add the horizontal space given by *length*. The *length* is a rubber length, that is, it may contain a `plus` or `minus` component, in any unit that LaTeX understands (see Chapter 14 [Lengths], page 90).

This command can add both positive and negative space; adding negative space is like backspacing.

Normally when TeX breaks a paragraph into lines it discards white space (glues and kerns) that would come at the start of a line, so you get an inter-word space or a line break between words but not both. This command's starred version `\hspace*{...}` puts a non-discardable invisible item in front of the space, so the space appears in the output.

This example make a one-line paragraph that puts '`Name:`' an inch from the right margin.

```
\noindent\makebox[\linewidth]{\hspace{\fill}Name:\hspace{1in}}
```

19.2 \hfill

Produce a rubber length which has no natural space but can stretch horizontally as far as needed (see Chapter 14 [Lengths], page 90).

The command `\hfill` is equivalent to `\hspace{\fill}`. For space that does not disappear at line breaks use `\hspace*{\fill}` instead (see Section 19.1 [\hspace], page 114).

19.3 \spacefactor

Synopsis:

```
\spacefactor=integer
```

While LaTeX is making the page, to give the lines the best appearance it may stretch or shrink the gaps between words. The `\spacefactor` command (from Plain TeX) allows you to change the LaTeX's default behavior.

After LaTeX places each character, or rule or other box, it sets a parameter called the *space factor*. If the next thing in the input is a space then this parameter affects how much of a horizontal gap LaTeX will have it span. (This gap is not a character; it is called *interword glue*.) A larger space factor means that the glue gap can stretch more and shrink less.

Normally, the space factor is 1000; this value is in effect following most characters, and any non-character box or math formula. But it is 3000 after a period, exclamation mark, or question mark, it is 2000 after a colon, 1500 after a semicolon, 1250 after a comma, and 0 after a right parenthesis or bracket, or closing double quote or single quote. Finally, it is 999 after a capital letter.

If the space factor f is 1000 then the glue gap will be the font's normal space value (for Computer Modern Roman 10 point this is 3.3333 points). Otherwise, if the space factor f is greater than 2000 then TEX adds the font's extra space value (for Computer Modern Roman 10 point this is 1.11111 points), and then the font's normal stretch value is multiplied by $f/1000$ and the normal shrink value is multiplied by $1000/f$ (for Computer Modern Roman 10 point these are 1.66666 and 1.11111 points). In short, compared to a normal space, such as the space following a word ending in a lowercase letter, inter-sentence spacing has a fixed extra space added and then the space can stretch 3 times as much and shrink 1/3 as much.

The rules for how TEX uses space factors are even more complex because they play two more roles. In practice, there are two consequences. First, if a period or other punctuation is followed by a close parenthesis or close double quote then its effect is still in place, that is, the following glue will have increased stretch and shrink. Second, conversely, if punctuation comes after a capital letter then its effect is not in place so you get an ordinary space. For how to adjust to this second case, for instance if an abbreviation does not end in a capital letter, see Section 19.3.1 [\(SPACE) and \@], page 115.

19.3.1 `\(SPACE)` and `\@`

Here, `\(SPACE)` means a backslash followed by a space. These commands mark a punctuation character, typically a period, as either ending a sentence or as ending an abbreviation.

By default, in justifying a line LATEX adjusts the space after a sentence-ending period (or a question mark, exclamation point, comma, or colon) more than the space between words. See Section 19.3 [\spacefactor], page 114. As described there, LATEX assumes that the period ends a sentence unless it is preceded by a capital letter, in which case it takes that period for part of an abbreviation. Note that if a sentence-ending period is immediately followed by a right parenthesis or bracket, or right single or double quote, then the space effect of that period follows through that parenthesis or quote.

So: if you have a period ending an abbreviation whose last letter is not a capital letter, and that abbreviation is not the last word in the sentence, then follow that period with a backslash-space (`\ `) or a tie (`~`) or a `\@`. Examples are `Nat.\ Acad.\ Science`, and `Mr.~Bean`, and `(manure, etc.\@) for sale` (note that in the last the `\@` comes before the closing parenthesis).

In the opposite situation, if you have a capital letter followed by a period that does end the sentence, then put `\@` before the period. For example, `book by the MAA\@.` will have correct inter-sentence spacing after the period.

For another use of `\(SPACE)`, see Section 19.4 [\(SPACE) after control sequence], page 116.

19.3.2 `\frenchspacing`

This declaration (from Plain TEX) causes LATEX to treat inter-sentence spacing in the same way as interword spacing.

In justifying the text in a line, some typographic traditions, including English, prefer to adjust the space between sentences (or after other punctuation marks) more than the space between words. Following this declaration, all spaces are instead treated equally.

Revert to the default behavior by declaring `\nonfrenchspacing`.

19.3.3 \normalsfcodes

Reset the LaTeX space factor values to the default.

19.4 \ after control sequence

The \ command is often used after control sequences to keep them from gobbling the space that follows, as in '\TeX\ is nice'. And, under normal circumstances, \tab and \newline are equivalent to \ . For another use of \ , see also Section 19.3.1 [\(SPACE) and \@], page 115.

Some people prefer to use {} for the same purpose, as in \TeX{} is nice. This has the advantage that you can always write it the same way, namely \TeX{}, whether it is followed by a space or by a punctuation mark. Compare:

```
\TeX\ is a nice system. \TeX, a nice system.

\TeX{} is a nice system. \TeX{}, a nice system.
```

Some individual commands, notably those defined with the xspace, package do not follow the standard behavior.

19.5 \thinspace: Insert 1/6 em

\thinspace produces an unbreakable and unstretchable space that is 1/6 of an em. This is the proper space to use between nested quotes, as in '".

19.6 \/: Insert italic correction

The \/ command produces an *italic correction*. This is a small space defined by the font designer for a given character, to avoid the character colliding with whatever follows. The italic *f* character typically has a large italic correction value.

If the following character is a period or comma, it's not necessary to insert an italic correction, since those punctuation symbols have a very small height. However, with semicolons or colons, as well as normal letters, it can help. Compare *f: f;* with *f: f;*.

When changing fonts with commands such as \textit{italic text} or {\itshape italic text}, LaTeX will automatically insert an italic correction if appropriate (see Section 4.1 [Font styles], page 17).

Despite the name, roman characters can also have an italic correction. Compare pdfTeX with pdfTeX.

There is no concept of italic correction in math mode; spacing is done in a different way.

19.7 \hrulefill \dotfill

Produce an infinite rubber length (see Chapter 14 [Lengths], page 90) filled with a horizontal rule (that is, a line) or with dots, instead of just white space.

When placed between blank lines this example creates a paragraph that is left and right justified, where the space in the middle is filled with evenly spaced dots.

```
\noindent Jack Aubrey\dotfill Melbury Lodge
```

To make the rule or dots go to the line's end use \null at the start or end.

To change the rule's thickness, copy the definition and adjust it, as with `\renewcommand{\hrulefill}{\leavevmode\leaders\hrule height 1pt\hfill\kern\z@}`, which changes the default thickness of 0.4 pt to 1 pt. Similarly, adjust the dot spacing as with `\renewcommand{\dotfill}{\leavevmode\cleaders\hb@xt@ 1.00em{\hss .\hss }\hfill\kern\z@}`, which changes the default length of 0.33 em to 1.00 em.

19.8 \addvspace

`\addvspace{`*length*`}`

Add a vertical space of height *length*, which is a rubber length (see Chapter 14 [Lengths], page 90). However, if vertical space has already been added to the same point in the output by a previous `\addvspace` command then this command will not add more space than what is needed to make the natural length of the total vertical space equal to *length*.

Use this command to adjust the vertical space above or below an environment that starts a new paragraph. For instance, a Theorem environment is defined to begin and end with `\addvspace{...}` so that two consecutive Theorem's are separated by one vertical space, not two.

This command is fragile (see Section 12.9 [\protect], page 84).

The error '`Something's wrong--perhaps a missing \item`' means that you were not in vertical mode when you invoked this command; one way to change that is to precede this command with a `\par` command.

19.9 \bigskip \medskip \smallskip

These commands produce a given amount of space, specified by the document class.

`\bigskip` The same as `\vspace{\bigskipamount}`, ordinarily about one line space, with stretch and shrink (the default for the `book` and `article` classes is `12pt plus 4pt minus 4pt`).

`\medskip` The same as `\vspace{\medskipamount}`, ordinarily about half of a line space, with stretch and shrink (the default for the `book` and `article` classes is `6pt plus 2pt minus 2pt`).

`\smallskip`
The same as `\vspace{\smallskipamount}`, ordinarily about a quarter of a line space, with stretch and shrink (the default for the `book` and `article` classes is `3pt plus 1pt minus 1pt`).

19.10 \vfill

End the current paragraph and insert a vertical rubber length (see Chapter 14 [Lengths], page 90) that is infinite, so it can stretch or shrink as far as needed.

It is often used in the same way as `\vspace{\fill}`, except that `\vfill` ends the current paragraph, whereas `\vspace{\fill}` adds the infinite vertical space below its line irrespective of the paragraph structure. In both cases that space will disappear at a page boundary; to circumvent this see Section 19.11 [\vspace], page 118.

In this example the page is filled, so the top and bottom lines contain the text '`Lost Dog!`' and the third '`Lost Dog!`' is exactly halfway between them.

```
\begin{document}
Lost Dog!
\vfill
Lost Dog!
\vfill
Lost Dog!
\end{document}
```

19.11 \vspace{*length*}

Synopsis, one of these two:

```
\vspace{length}
\vspace*{length}
```

Add the vertical space *length*. This can be negative or positive, and is a rubber length (see Chapter 14 [Lengths], page 90).

LaTeX removes the vertical space from `\vspace` at a page break, that is, at the top or bottom of a page. The starred version `\vspace*{...}` causes the space to stay.

If `\vspace` is used in the middle of a paragraph (i.e., in horizontal mode), the space is inserted *after* the line with the `\vspace` command. A new paragraph is not started.

In this example the two questions will be evenly spaced vertically on the page, with at least one inch of space below each.

```
\begin{document}
1) Who put the bomp in the bomp bah bomp bah bomp?
\vspace{1in plus 1fill}

2) Who put the ram in the rama lama ding dong?
\vspace{1in plus 1fill}
\end{document}
```

20 Boxes

All the predefined length parameters (see Section 14.7 [Predefined lengths], page 91) can be used in the arguments of the box-making commands.

20.1 \mbox{*text*}

The `\mbox` command creates a box just wide enough to hold the text created by its argument. The *text* is not broken into lines, so it can be used to prevent hyphenation.

20.2 \fbox and \framebox

Synopses:

```
\fbox{text}
\framebox[width][position]{text}
```

The `\fbox` and `\framebox` commands are like `\mbox`, except that they put a frame around the outside of the box being created.

In addition, the `\framebox` command allows for explicit specification of the box width with the optional *width* argument (a dimension), and positioning with the optional *position* argument.

Both commands produce a rule of thickness `\fboxrule` (default `0.4pt`), and leave a space of `\fboxsep` (default `3pt`) between the rule and the contents of the box.

See Section 8.19.3 [\framebox (picture)], page 56, for the `\framebox` command in the `picture` environment.

20.3 lrbox

Synopsis:

```
\begin{lrbox}{\cmd}
  text
\end{lrbox}
```

This is the environment form of Section 20.8 [\sbox], page 121.

The *text* inside the environment is saved in the box `\cmd`, which must have been declared with `\newsavebox`.

20.4 \makebox

Synopsis:

```
\makebox[width][position]{text}
```

The `\makebox` command creates a box just wide enough to contain the *text* specified. The width of the box can be overridden by the optional *width* argument. The position of the text within the box is determined by the optional *position* argument, which may take the following values:

`c` Centered (default).

`l` Flush left.

`r` Flush right.

`s` Stretch (justify) across entire *width*; *text* must contain stretchable space for this to work.

`\makebox` is also used within the `picture` environment see Section 8.19.2 [\makebox (picture)], page 56.

20.5 \parbox

Synopsis:

```
\parbox[position][height][inner-pos]{width}{text}
```

The `\parbox` command produces a box whose contents are created in *paragraph mode*. It should be used to make a box small pieces of text, with nothing fancy inside. In particular, you shouldn't use any paragraph-making environments inside a `\parbox` argument. For larger pieces of text, including ones containing a paragraph-making environment, you should use a `minipage` environment (see Section 8.18 [minipage], page 54).

`\parbox` has two mandatory arguments:

width the width of the parbox;

text the text that goes inside the parbox.

By default LaTeX will position vertically a parbox so its center lines up with the center of the surrounding text line. When the optional *position* argument is present and equal either to '`t`' or '`b`', this allows you respectively to align either the top or bottom line in the parbox with the baseline of the surrounding text. You may also specify '`m`' for *position* to get the default behaviour.

The optional *height* argument overrides the natural height of the box.

The *inner-pos* argument controls the placement of the text inside the box, as follows; if it is not specified, *position* is used.

`t` text is placed at the top of the box.

`c` text is centered in the box.

`b` text is placed at the bottom of the box.

`s` stretch vertically; the text must contain vertically stretchable space for this to work.

20.6 \raisebox

Synopsis:

```
\raisebox{distance}[height][depth]{text}
```

The `\raisebox` command raises or lowers *text*. The first mandatory argument specifies how high *text* is to be raised (or lowered if it is a negative amount). *text* itself is processed in LR mode.

The optional arguments *height* and *depth* are dimensions. If they are specified, LaTeX treats *text* as extending a certain distance above the baseline (*height*) or below (*depth*), ignoring its natural height and depth.

20.7 \savebox

Synopsis:

```
\savebox{\boxcmd}[width][pos]{text}
```

This command typeset *text* in a box just as with `\makebox` (see Section 20.4 [\makebox], page 119), except that instead of printing the resulting box, it saves it in the box labeled *boxcmd*, which must have been declared with `\newsavebox` (see Section 12.5 [\newsavebox], page 80).

20.8 \sbox{\boxcmd}{text}

Synopsis:

```
\sbox{\boxcmd}{text}
```

`\sbox` types *text* in a box just as with `\mbox` (see Section 20.1 [\mbox], page 119) except that instead of the resulting box being included in the normal output, it is saved in the box labeled *boxcmd*. *boxcmd* must have been previously declared with `\newsavebox` (see Section 12.5 [\newsavebox], page 80).

20.9 \usebox{\boxcmd}

Synopsis:

```
\usebox{\boxcmd}
```

`\usebox` produces the box most recently saved in the bin *boxcmd* by a `\savebox` command (see Section 20.7 [\savebox], page 121).

21 Color

You can add color to text, rules, etc. You can also have color in a box or on an entire page and write text on top of it.

Color support comes as an additional package. So all the commands below will only work if your document preamble contains `\usepackage{color}`, that brings in the standard package.

Many other packages also supplement LaTeX's color abilities. Particularly worth mentioning is `xcolor`, which is widely used and significantly extends the capabilities described here, including adding '`HTML`' and '`Hsb`' color models.

21.1 Color package options

Synopsis (must be in the document preamble):

```
\usepackage[comma-separated option list]{color}
```

When you load the `color` package there are two kinds of available options.

The first specifies the *printer driver*. LaTeX doesn't contain information about different output systems but instead depends on information stored in a file. Normally you should not specify the driver option in the document, and instead rely on your system's default. One advantage of this is that it makes the document portable across systems. For completeness we include a list of the drivers. The currently relevant ones are: `dvipdfmx`, `dvips`, `dvisvgm`, `luatex`, `pdftex`, `xetex`. The two `xdvi` and `oztex` are essentially aliases for `dvips` (and `xdvi` is monochrome). Ones that should not be used for new systems are: `dvipdf`, `dvipdfm`, `dviwin`, `dvipsone`, `emtex`, `pctexps`, `pctexwin`, `pctexhp`, `pctex32`, `truetex`, `tcidvi`, `vtex` (and `dviwindo` is an alias for `dvipsone`).

The second kind of options, beyond the drivers, are below.

`monochrome`
: Disable the color commands, so that they do not generate errors but do not generate color either.

`dvipsnames`
: Make available a list of 68 color names that are often used, particularly in legacy documents. These color names were originally provided by the `dvips` driver, giving the option name.

`nodvipsnames`
: Do not load that list of color names, saving LaTeX a tiny amount of memory space.

21.2 Color models

A *color model* is a way of representing colors. LaTeX's capabilities depend on the printer driver. However, the `pdftex`, `xetex`, and `luatex` printer drivers are today by far the most commonly used. The models below work for those drivers. All but one of these is also supported by essentially all other printer drivers used today.

Note that color combination can be additive or subtractive. Additive mixes colors of light, so that for instance combining full intensities of red, green, and blue produces white.

Subtractive mixes pigments, such as with inks, so that combining full intensity of cyan, magenta, and yellow makes black.

`cmyk` A comma-separated list with four real numbers between 0 and 1, inclusive. The first number is the intensity of cyan, the second is magenta, and the others are yellow and black. A number value of 0 means minimal intensity, while a 1 is for full intensity. This model is often used in color printing. It is a subtractive model.

`gray` A single real number between 0 and 1, inclusive. The colors are shades of grey. The number 0 produces black while 1 gives white.

`rgb` A comma-separated list with three real numbers between 0 and 1, inclusive. The first number is the intensity of the red component, the second is green, and the third the blue. A number value of 0 means that none of that component is added in, while a 1 means full intensity. This is an additive model.

`RGB` (`pdftex`, `xetex`, `luatex` drivers) A comma-separated list with three integers between 0 and 255, inclusive. This model is a convenience for using `rgb` since outside of LaTeX colors are often described in a red-green-blue model using numbers in this range. The values entered here are converted to the `rgb` model by dividing by 255.

`named` Colors are accessed by name, such as '`PrussianBlue`'. The list of names depends on the driver, but all support the names '`black`', '`blue`', '`cyan`', '`green`', '`magenta`', '`red`', '`white`', and '`yellow`' (See the `dvipsnames` option in Section 21.1 [Color package options], page 122).

21.3 Commands for color

These are the commands available with the `color` package.

21.3.1 Define colors

Synopsis:

```
\definecolor{name}{model}{specification}
```

Give the name *name* to the color. For example, after `\definecolor{silver}{rgb}{0.75,0.75,0.74}` you can use that color name with `Hi ho, \textcolor{silver}{Silver}!`.

This example gives the color a more abstract name, so it could change and not be misleading.

```
\definecolor{logocolor}{RGB}{145,92,131}    % RGB needs pdflatex
\newcommand{\logo}{\textcolor{logocolor}{Bob's Big Bagels}}
```

Often a document's colors are defined in the preamble, or in the class or style, rather than in the document body.

21.3.2 Colored text

Synopses:

```
\textcolor{name}{...}
\textcolor[color model]{color specification}{...}
```

or

```
\color{name}
\color[color model]{specification}
```

The affected text gets the color. This line

```
\textcolor{magenta}{My name is Ozymandias, king of kings:} Look on my works, ye Mighty
```

causes the first half to be in magenta while the rest is in black. You can use a color declared with `\definecolor` in exactly the same way that we just used the builtin color 'magenta'.

```
\definecolor{MidlifeCrisisRed}{rgb}{1.0,0.11,0.0}
I'm thinking about getting a \textcolor{MidlifeCrisisRed}{sports car}.
```

The two `\textcolor` and `\color` differ in that the first is a command form, enclosing the text to be colored as an argument. Often this form is more convenient, or at least more explicit. The second form is a declaration, as in `The moon is made of {\color{green} green} cheese`, so it is in effect until the end of the current group or environment. This is sometimes useful when writing macros or as below where it colors everything inside the `center` environment, including the vertical and horizontal lines.

```
\begin{center} \color{blue}
  \begin{tabular}{l|r}
    UL &UR \\ \hline
    LL &LR
  \end{tabular}
\end{center}
```

You can use color in equations. A document might have `\definecolor{highlightcolor}{RGB}{225,15,0}` in the preamble, and then contain this equation.

```
\begin{equation}
  \int_a^b \textcolor{highlightcolor}{f'(x)}\,dx=f(b)-f(a)
\end{equation}
```

Typically the colors used in a document are declared in a class or style but sometimes you want a one-off. Those are the second forms in the synopses.

```
Colors of \textcolor[rgb]{0.33,0.14,0.47}{Purple} and {\color[rgb]{0.72,0.60,0.37} Gol
```

The format of *color specification* depends on the color model (see Section 21.2 [Color models], page 122). For instance, while `rgb` takes three numbers, `gray` takes only one.

```
The selection was \textcolor[gray]{0.5}{grayed out}.
```

Colors inside colors do not combine. Thus

```
\textcolor{green}{kind of \textcolor{blue}{blue}}
```

has a final word that is blue, not a combination of blue and green.

21.3.3 Colored boxes

Synopses:

```
\colorbox{name}{...}
\colorbox[model name]{box background color specification}{...}
```

or

```
\fcolorbox{frame color}{box background color}{...}
```

```
\fcolorbox[model name]{frame color specification}{box background color specification}{
```

Make a box with the stated background color. The `\fcolorbox` command puts a frame around the box. For instance this

```
Name:~\colorbox{cyan}{\makebox[5cm][l]{\strut}}
```

makes a cyan-colored box that is five centimeters long and gets its depth and height from the `\strut` (so the depth is `-.3\baselineskip` and the height is `\baselineskip`). This puts white text on a blue background.

```
\colorbox{blue}{\textcolor{white}{Welcome to the machine.}}
```

The `\fcolorbox` commands use the same parameters as `\fbox` (see Section 20.2 [\fbox and \framebox], page 119), `\fboxrule` and `\fboxsep`, to set the thickness of the rule and the boundary between the box interior and the surrounding rule. LaTeX's defaults are `0.4pt` and `3pt`, respectively.

This example changes the thickness of the border to 0.8 points. Note that it is surrounded by curly braces so that the change ends at the end of the second line.

```
{\setlength{\fboxrule}{0.8pt}
\fcolorbox{black}{red}{Under no circumstances turn this knob.}}
```

21.3.4 Colored pages

Synopses:

```
\pagecolor{name}
\pagecolor[color model]{color specification}
\nopagecolor
```

The first two set the background of the page, and all subsequent pages, to the color. For an explanation of the specification in the second form see Section 21.3.2 [Colored text], page 123. The third returns the background to normal, which is a transparent background. (If that is not supported use `\pagecolor{white}`, although that will make a white background rather than the default transparent background.)

```
 ...
\pagecolor{cyan}
 ...
\nopagecolor
```

22 Graphics

You can use graphics such as PNG or PDF files in your LaTeX document. You need an additional package, which comes standard with LaTeX. This example is the short how-to.

```
\include{graphicx}  % goes in the preamble
  ...
\includegraphics[width=0.5\linewidth]{plot.pdf}
```

To use the commands described here your document preamble must contain either `\usepackage{graphicx}` or `\usepackage{graphics}`. Most of the time, `graphicx` is the better choice.

Graphics come in two main types, raster and vector. LaTeX can use both. In raster graphics the file contains an entry for each location in an array, describing what color it is. An example is a photograph, in JPG format. In vector graphics, the file contains a list of instructions such as '`draw a circle with this radius and that center`'. An example is a line drawing produced by the Asymptote program, in PDF format. Generally vector graphics are more useful because you can rescale their size without pixelation or other problems, and because they often have a smaller size.

There are systems particularly well-suited to make graphics for a LaTeX document. For example, these allow you to use the same fonts as in your document. LaTeX comes with a `picture` environment (see Section 8.19 [picture], page 54) that has simple capabilities. Besides that, there are other ways to include the graphic-making commands in the document. Two such systems are the PSTricks and TikZ packages. There are also systems external to LaTeX, that generate a graphic that you include using the commands of this chapter. Two that use a programming language are Asymptote and MetaPost. One that uses a graphical interface is Xfig. Full description of these systems is outside the scope of this document; see their documentation.

22.1 Graphics package options

Synopsis (must be in the document preamble):

```
\usepackage[comma-separated option list]{graphics}
```

or

```
\usepackage[comma-separated option list]{graphicx}
```

The `graphicx` package has a format for optional arguments to the `\includegraphics` command that is convenient (it is the key-value format), so it is the better choice for new documents. When you load the `graphics` or `graphicx` package with `\usepackage` there are two kinds of available options.

The first is that LaTeX does not contain information about different output systems but instead depends on information stored in a *printer driver* file. Normally you should not specify the driver option in the document, and instead rely on your system's default. One advantage of this is that it makes the document portable across systems.

For completeness here is a list of the drivers. The currently relevant ones are: `dvipdfmx`, `dvips`, `dvisvgm`, `luatex`, `pdftex`, `xetex`. The two `xdvi` and `oztex` are essentially aliases for `dvips` (and `xdvi` is monochrome). Ones that should not be used for new systems are: `dvipdf`, `dvipdfm`, `dviwin`, `dvipsone`, `emtex`, `pctexps`, `pctexwin`, `pctexhp`, `pctex32`,

`truetex`, `tcidvi`, `vtex` (and `dviwindo` is an alias for `dvipsone`). These are stored in files with a `.def` extension, such as `pdftex.def`.

The second kind of options are below.

`demo` Instead of an image file, LaTeX puts in a 150 pt by 100 pt rectangle (unless another size is specified in the `\includegraphics` command).

`draft` For each graphic file, it is not shown but instead the file name is printed in a box of the correct size. In order to determine the size, the file must be present.

`final` (Default) Override any previous `draft` option, so that the document shows the contents of the graphic files.

`hiderotate`
Do not show rotated text. (This allows for the possibility that a previewer does not have the capability to rotate text.)

`hidescale`
Do not show scaled text. (This allows for the possibility that a previewer does not have the capability to scale.)

`hiresbb` In a PS or EPS file the graphic size may be specified in two ways. The `%%BoundingBox` lines describe the graphic size using integer multiples of a PostScript point, that is, integer multiples of 1/72 inch. A later addition to the PostScript language allows decimal multiples, such as 1.23, in `%%HiResBoundingBox` lines. This option has LaTeX to read the size from the latter.

22.2 Graphics package configuration

These commands configure the way LaTeX searches the file system for the graphic.

The behavior of file system search code is necessarily platform dependent. In this document we cover Linux, Macintosh, and Windows, as those systems are typically configured. For other situations consult the documentation in `grfguide.pdf`, or the LaTeX source, or your TeX distribution's documentation.

22.2.1 \graphicspath

Synopsis:

```
\graphicspath{list of dir names inside curly brackets}
```

Declare a list of directories to search for graphics files. This allows you to later say something like `\includegraphics{lion.png}` instead of having to give its path.

LaTeX always looks for graphic files first in the current directory. The declaration below tells the system to then look in the subdirectory `pix`, and then `../pix`.

```
\usepackage{graphicx}   % or graphics; put in preamble
  ...
\graphicspath{ {pix/} {../pix/} }
```

The `\graphicspath` declaration is optional. If you don't include it then LaTeX's default is to search all of the places that it usually looks for a file (it uses LaTeX's `\input@path`). In particular, in this case one of the places it looks is the current directory.

Enclose each directory name in curly braces; for example, above it says '`{pix}`'. Do this even if there is only one directory. Each directory name must end in a forward slash, `/`. This is true even on Windows, where good practice is to use forward slashes for all the directory separators since it makes the document portable to other platforms. If you have spaces in your directory name then use double quotes, as with `{"my docs/"}`. Getting one of these rules wrong will cause LaTeX to report `Error: File 'filename' not found`.

Basically, the algorithm is that with this example, after looking in the current directory,

```
\graphicspath{ {pix/} {../pix/} }
...
\usepackage{lion.png}
```

for each of the listed directories, LaTeX concatenates it with the file name and searches for the result, checking for `pix/lion.png` and then `../pix/lion.png`. This algorithm means that the `\graphicspath` command does not recursively search subdirectories: if you issue `\graphicspath{{a/}}` and the graphic is in `a/b/lion.png` then LaTeX will not find it. It also means that you can use absolute paths such as `\graphicspath{{/home/jim/logos/}}` or `\graphicspath{{C:/Users/Albert/Pictures/}}`. However, using these means that the document is not portable. (You could preserve portability by adjusting your TeX system settings configuration file parameter `TEXINPUTS`; see the documentation of your system.)

You can use `\graphicspath` in the preamble or in the document body. You can use it more than once. For debugging, show its value with `\makeatletter\typeout{\Ginput@path}\makeatother`.

The directories are taken with respect to the base file. That is, suppose that you are working on a document based on `book/book.tex` and it contains `\include{chapters/chap1}`. If in `chap1.tex` you put `\graphicspath{{plots/}}` then LaTeX will not search for graphics in `book/chapters/plots`, but instead in `book/plots`.

22.2.2 \DeclareGraphicsExtensions

Synopses:

```
\DeclareGraphicsExtensions{comma-separated list of file extensions}
```

Declare the filename extensions to try. This allows you to specify the order in which to choose graphic formats when you include graphic files by giving the filename without the extension, as in `\includegraphics{functionplot}`.

In this example, LaTeX will find files in the PNG format before PDF files.

```
\DeclareGraphicsExtensions{.png,PNG,.pdf,.PDF}
  ...
\includegraphics{lion}   % will find lion.png before lion.pdf
```

Because the file name `lion` does not have a period, LaTeX uses the extension list. For each directory in the graphics path (see Section 22.2.1 [\graphicspath], page 127), LaTeX will try the extensions in the order given. If it does not find such a file after trying all the directories and extensions then it reports '`! LaTeX Error: File 'lion' not found`'. Note that you must include the periods at the start of the extensions.

Because Linux and Macintosh filenames are case sensitive, the list of file extensions is case sensitive on those platforms. The Windows platform is not case sensitive.

You are not required to include `\DeclareGraphicsExtensions` in your document; the printer driver has a sensible default. For example, the most recent `pdftex.def` has the extension list '`.png,.pdf,.jpg,.mps,.jpeg,.jbig2,.jb2,.PNG,.PDF,.JPG,.JPEG,.JBIG2,.JB2`'.

You can use this command in the preamble or in the document body. You can use it more than once. For debugging, show its value with `\makeatletter\typeout{\Gin@extensions}\makeatother`.

22.2.3 `\DeclareGraphicsRule`

Synopsis:

```
\DeclareGraphicsRule{extension}{type}{size-file extension}{command}
```

Declare how to handle graphic files whose names end in *extension*.

This example declares that all files with names have the form `filename-without-dot.mps` will be treated as output from MetaPost, meaning that the printer driver will use its MetaPost-handling code to input the file.

```
\DeclareGraphicsRule{.mps}{mps}{.mps}{}
```

This

```
\DeclareGraphicsRule{*}{mps}{*}{}
```

tells LaTeX that it should handle as MetaPost output any file with an extension not covered by another rule, so it covers `filename.1`, `filename.2`, etc.

This describes the four arguments.

extension The file extension to which this rule applies. The extension is anything after and including the first dot in the filename. Use the Kleene star, `*`, to denote the default behaviour for all undeclared extensions.

type The type of file involved. This type is a string that must be defined in the printer driver. For instance, files with extensions `.ps`, `.eps`, or `.ps.gz` may all be classed as type `eps`. All files of the same type will be input with the same internal command by the printer driver. For example, the file types that `pdftex` recognizes are: `jpg`, `jbig2`, `mps`, `pdf`, `png`, `tif`.

size-file extension
The extension of the file to be read to determine the size of the graphic, if there is such a file. It may be the same as *extension* but it may be different.

As an example, consider a PostScript graphic. To make it smaller, it might be compressed into a `.ps.gz` file. Compressed files are not easily read by LaTeX so you can put the bounding box information in a separate file. If *size-file extension* is empty then you must specify size information in the arguments of `\includegraphics`.

If the driver file has a procedure for reading size files for `type` then that will be used, otherwise it will use the procedure for reading `.eps` files. (Thus you may specify the size of bitmap files in a file with a PostScript style `%%BoundingBox` line if no other format is available.)

command A command that will be applied to the file. This is very often left blank. This command must start with a single backward quote. Thus,

`\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{‘gunzip -c #1}` specifies that any file with the extension `.eps.gz` should be treated as an `eps` file, with the the BoundingBox information stored in the file with extension `.eps.bb`, and that the command `gunzip -c` will run on your platform to decompresses the file.

Such a command is specific to your platform. In addition, your TEX system must allow you to run external commands; as a security measure modern systems restrict running commands unless you explicitly allow it. See the documentation for your TEX distribution.

22.3 Commands for graphics

These are the commands available with the `graphics` and `graphicx` packages.

22.3.1 \includegraphics

Synopses for `graphics` package:

```
\includegraphics{filename}
\includegraphics[urx,ury]{filename}
\includegraphics[llx,lly][urx,ury]{filename}
\includegraphics*{filename}
\includegraphics*[urx,ury]{filename}
\includegraphics*[llx,lly][urx,ury]{filename}
```

Synopses for `graphicx` package:

```
\includegraphics{filename}
\includegraphics[key-value list]{filename}
\includegraphics*{filename}
\includegraphics*[key-value list]{filename}
```

Include a graphics file. The starred form `\includegraphics*` will clip the graphic to the size specified, while for the unstarred form any part of the graphic that is outside the box of the specified size will over-print the surrounding area.

This

```
\usepackage{graphicx}  % in preamble
  ...
\begin{center}
  \includegraphics{plot.pdf}
\end{center}
```

will incorporate into the document the graphic in `plot.pdf`, centered and at its nominal size. You can also give a path to the file, as with `\includegraphics{graphics/plot.pdf}`. To specify a list of locations to search for the file, see Section 22.2.1 [\graphicspath], page 127.

If your filename includes spaces then put it in double quotes, as with `\includegraphics{"sister picture.jpg"}`.

The `\includegraphics{filename}` command decides on the type of graphic by splitting *filename* on the first dot. You can use *filename* with no dot, as in `\includegraphics{turing}` and then LATEX tries a sequence of extensions such

as `.png` and `.pdf` until it finds a file with that extension (see Section 22.2.2 [\DeclareGraphicsExtensions], page 128).

If your file name contains dots before the extension then you can hide them with curly braces, as in `\includegraphics{{plot.2018.03.12.a}.pdf}`. Or, if you use the `graphicx` package then you can use the options `type` and `ext`; see below. This and other filename issues are also handled with the package `grffile`.

This example puts a graphic in a figure environment so LaTeX can move it to the next page if fitting it on the current page is awkward (see Section 8.10 [figure], page 44).

```
\begin{figure}
  \centering
  \includegraphics[width=3cm]{lungxray.jpg}
  \caption{The evidence is overwhelming: don't smoke.}  \label{fig:xray}
\end{figure}
```

This places a graphic that will not float, so it is sure to appear at this point in the document even if makes LaTeX stretch the text or resort to blank areas on the page. It will be centered and will have a caption.

```
\usepackage{caption}  % in preamble
  ...
\begin{center}
  \includegraphics{pix/nix.png}
  \captionof{figure}{The spirit of the night} \label{pix:nix}  % if you want a caption
\end{center}
```

This example puts a box with a graphic side by side with one having text, with the two vertically centered.

```
\newcommand*{\vcenteredhbox}[1]{\begingroup
                                \setbox0=\hbox{#1}\parbox{\wd0}{\box0}\endgroup}
  ...
\begin{center}
  \vcenteredhbox{\includegraphics[width=0.4\textwidth]{plot}}
  \hspace{1em}
  \vcenteredhbox{\begin{minipage}{0.4\textwidth}
                   \begin{displaymath}
                     f(x)=x\cdot \sin (1/x)
                   \end{displaymath}
                 \end{minipage}}
\end{center}
```

If you use the `graphics` package then the only options involve the size of the graphic (but see Section 22.3.2 [\rotatebox], page 135, and Section 22.3.3 [\scalebox], page 136). When one optional argument is present then it is `[`*`urx`*`,`*`ury`*`]` and it gives the coordinates of the top right corner of the image, as a pair of TeX dimensions (see Section 14.1 [Units of length], page 90). If the units are omitted they default to `bp`. In this case, the lower left corner of the image is assumed to be at (0,0). If two optional arguments are present then the leading one is `[`*`llx`*`,`*`lly`*`]`, specifying the coordinates of the image's lower left. Thus, `\includegraphics[1in,0.618in]{...}` calls for the graphic to be placed so it is 1 inch wide and 0.618 inches tall and so its origin is at (0,0).

The `graphicx` package gives you many more options. Specify them in a key-value form, as here.

```
\begin{center}
  \includegraphics[width=1in,angle=90]{lion}
  \hspace{2em}
  \includegraphics[angle=90,width=1in]{lion}
\end{center}
```

The options are read left-to-right. So the first graphic above is made one inch wide and then rotated, while the second is rotated and then made one inch wide. Thus, unless the graphic is perfectly square, the two will end with different widths and heights.

There are many options. The primary ones are listed first.

Note that a graphic is placed by LaTeX into a box, which is traditionally referred to as its bounding box (distinct from the PostScript BoundingBox described below). The graphic's printed area may go beyond this box, or sit inside this box, but when LaTeX makes up a page it puts together boxes and this is the box allocated for the graphic.

`width` The graphic will be shown so its bounding box is this width. An example is `\includegraphics[width=1in]{plot}`. You can use the standard TeX dimensions (see Section 14.1 [Units of length], page 90) and also convenient is `\linewidth`, or in a two-column document, `\columnwidth` (see Section 5.5 [Page layout parameters], page 25). An example is that by using the `calc` package you can make the graphic be 1 cm narrow than the width of the text with `\includegraphics[width=\linewidth-1.0cm]{hefferon.jpg}`.

`height` The graphic will be shown so its bounding box is this height. You can use the standard TeX dimensions (see Section 14.1 [Units of length], page 90), and also convenient are `\pageheight` and `\textheight` (see Section 5.5 [Page layout parameters], page 25). For instance, `\includegraphics[height=0.25\textheight]{godel}` will make the graphic be a quarter of the height of the text area.

`totalheight`
The graphic will be shown so its bounding box has this height plus depth. This differs from the height if the graphic was rotated. For instance, if it has been rotated by -90 then it will have zero height but a large depth.

`keepaspectratio`
If set to `true`, or just specified as with `\includegraphics[...,keepaspectratio,...]{...}` and you give as options both `width` and `height` (or `totalheight`), then LaTeX will make the graphic is as large as possible without distortion. That is, LaTeX will ensure that neither is the graphic wider than `width` nor taller than `height` (or `totalheight`).

`scale` Factor by which to scale the graphic. Specifying `\includegraphics[scale=2.0]{...}` makes the graphic twice its nominal size. This number may be any value; a number between 1 and 0 will shrink the graphic and a negative number will reflect it.

`angle` Rotate the picture. The angle is taken in degrees and counterclockwise. The graphic is rotated about its `origin`; see that option. For a complete description of how rotated material is typeset, see Section 22.3.2 [\rotatebox], page 135.

`origin` The point of the graphic about which the rotation happens. Possible values are any string containing one or two of: `l` for left, `r` for right, `b` for bottom, `c` for center, `t` for top, and `B` for baseline. Thus, `\includegraphics[angle=180,origin=c]{moon}` will turn the picture upside down from the center, while `\includegraphics[angle=180,origin=lB]{LeBateau}` will turn its picture upside down about its left baseline. (The character `c` gives the horizontal center in `bc` or `tc`, but gives the vertical center in `lc` or `rc`.) The default is `lB`.

To rotate about an arbitrary point, see Section 22.3.2 [\rotatebox], page 135.

These are lesser-used options.

`viewport` Pick out a subregion of the graphic to show. Takes four arguments, separated by spaces and given in TEX dimensions, as with `\includegraphics[.., viewport=0in 0in 1in 0.618in]{...}`. The dimensions default to big points, `bp`. They are taken relative to the origin specified by the bounding box. See also the `trim` option.

`trim` Gives parts of the graphic to not show. Takes four arguments, separated by spaces, that are given in TEX dimensions, as with `\includegraphics[.., trim= 0in 0.1in 0.2in 0.3in, ...]{...}`. These give the amounts of the graphic not to show, that is, LATEX will crop the picture by 0 inches on the left, 0.1 inches on the bottom, 0.2 inches on the right, and 0.3 inches on the top. See also the `viewport` option.

`clip` If set to `true`, or just specified as with `\includegraphics[...,clip,...]{...}`, then the graphic is cropped to the bounding box. You can get this effect by instead using the starred form of the command, as `\includegraphics*[...]{...}`.

`page` Give the page number of a multi-page PDF file. The default is `page=1`.

`pagebox` Specifies which bounding box to use for PDF files from among `mediabox`, `cropbox`, `bleedbox`, `trimbox`, or `artbox`. PDF files do not have the BoundingBox that PostScript files have, but may specify up to four predefined rectangles. The MediaBox gives the boundaries of the physical medium. The CropBox is the region to which the contents of the page are to be clipped when displayed. The BleedBox is the region to which the contents of the page should be clipped in production. The TrimBox is the intended dimensions of the finished page. The ArtBox is the extent of the page's meaningful content. The driver will set the image size based on CropBox if present, otherwise it will not use one of the others, with a driver-defined order of preference. MediaBox is always present.

`interpolate`
Enable or disable interpolation of raster images by the viewer. Can be set with `interpolate=true` or just specified as with `\includegraphics[...,interpolate,...]{...}`.

`quiet` Do not write information to the log. You can set it with `quiet=true` or just specified it with `\includegraphics[...,quite,...]{...}`,

`draft` If you set it with `draft=true` or just specified it with `\includegraphics[...,draft,...]{...}`, then the graphic will not appear in the document, possibly saving color printer ink. Instead, LaTeX will put an empty box of the correct size with the filename printed in it.

These options address the bounding box for Encapsulated PostScript graphic files, which have a size specified with a line `%%BoundingBox` that appears in the file. It has four values, giving the lower x coordinate, lower y coordinate, upper x coordinate, and upper y coordinate. The units are PostScript points, equivalent to TeX's big points, 1/72 inch. For example, if an `.eps` file has the line `%%BoundingBox 10 20 40 80` then its natural size is 30/72 inch wide by 60/72 inch tall.

`bb` Specify the bounding box of the displayed region. The argument is four dimensions separated by spaces, as with `\includegraphics[.., bb= 0in 0in 1in 0.618in]{...}`. Usually `\includegraphics` reads the BoundingBox numbers from the EPS file automatically, so this option is only useful if the bounding box is missing from that file or if you want to change it.

`bbllx, bblly, bburx, bbury`
Set the bounding box. These four are obsolete, but are retained for compatibility with old packages.

`natwidth, natheight`
An alternative for `bb`. Setting `\includegraphics[...,natwidth=1in,natheight=0.618in,...]` is the same as setting `bb=0 0 1in 0.618in`.

`hiresbb` If set to `true`, or just specified as with `\includegraphics[...,hiresbb,...]{...}`, then LaTeX will look for `%%HiResBoundingBox` lines instead of `%%BoundingBox` lines. (The `BoundingBox` lines use only natural numbers while the `HiResBoundingBox` lines use decimals; both use units equivalent to TeX's big points, 1/72 inch.) To override a prior setting of `true`, you can set it to `false`.

These following options allow a user to override LaTeX's method of choosing the graphic type based on the filename extension. An example is that `\includegraphics[type=png,ext=.xxx,read=.xxx]{lion}` will read the file `lion.xxx` as though it were `lion.png`. For more on these, see Section 22.2.3 [\DeclareGraphicsRule], page 129.

`type` Specify the graphics type.

`ext` Specify the graphics extension. Only use this in conjunction with the option `type`.

`read` Specify the file extension of the read file. Only use this in conjunction with the option `type`.

`command` Specify a command to be applied to this file. Only use this in conjunction with the option `type`.

22.3.2 \rotatebox

Synopsis for `graphics` package:

```
\rotatebox{angle}{material}
```

Synopses for `graphicx` package:

```
\rotatebox{angle}{material}
\rotatebox[key-value list]{angle}{material}
```

Put *material* in a box and rotate it *angle* degrees counterclockwise.

This example rotates the table column heads forty five degrees.

```
\begin{tabular}{ll}
  \rotatebox{45}{Character} &\rotatebox{45}{NATO phonetic}   \\
  A                         &AL-FAH  \\
  B                         &BRAH-VOH
\end{tabular}
```

The *material* can be anything that goes in a box, including a graphic.

```
\rotatebox[origin=c]{45}{\includegraphics[width=1in]{lion}}
```

To place the rotated material, the first step is that LaTeX sets *material* in a box, with a reference point on the left baseline. The second step is the rotation, by default about the reference point. The third step is that LaTeX computes a box to bound the rotated material. Fourth, LaTeX moves this box horizontally so that the left edge of this new bounding box coincides with the left edge of the box from the first step (they need not coincide vertically). This new bounding box, in its new position, is what LaTeX uses as the box when typesetting this material.

If you use the `graphics` package then the rotation is about the reference point of the box. If you use the `graphicx` package then then these are the options that can go in the *key-value list*, but note that you can get the same effect without needing this package, except for the `x` and `y` options (see Section 22.3.1 [\includegraphics], page 130).

`origin` The point of the *material*'s box about which the rotation happens. Possible values are any string containing one or two of: `l` for left, `r` for right, `b` for bottom, `c` for center, `t` for top, and `B` for baseline. Thus, `\includegraphics[angle=180,origin=c]{moon}` will turn the picture upside down from the center, while `\includegraphics[angle=180,origin=lB]{LeBateau}` will turn its picture upside down about its left baseline. (The character `c` gives the horizontal center in `bc` or `tc` but gives the vertical center in `lc` or `rc`.) The default is `lB`.

`x, y` Specify an arbitrary point of rotation with `\rotatebox[x=`*TeX dimension*`,y=`*TeX dimension*`]{...}` (see Section 14.1 [Units of length], page 90). These give the offset from the box's reference point.

`units` This key allows you to change the default of degrees counterclockwise. Setting `units=-360` changes the direction to degrees clockwise and setting `units=6.283185` changes to radians counterclockwise.

22.3.3 \scalebox

Synopses:

```
\scalebox{horizontal factor}{material}
\scalebox{horizontal factor}[vertical factor]{material}
\reflectbox{material}
```

Scale the *material*.

This example halves the size, both horizontally and vertically, of the first text and doubles the size of the second.

```
\scalebox{0.5}{DRINK ME} and \scalebox{2.0}{Eat Me}
```

If you do not specify the optional *vertical factor* then it defaults to the same value as the *horizontal factor*.

You can use this command to resize a graphic, as with `\scalebox{0.5}{\includegraphics{lion}}`. If you use the `graphicx` package then you can accomplish the same thing with optional arguments to `\includegraphics` (see Section 22.3.1 [\includegraphics], page 130).

The `\reflectbox` command abbreviates `\scalebox{-1}[1]{material}`. Thus, `Able was I\reflectbox{Able was I}` will show the phrase '`Able was I`' immediately followed by its mirror reflection.

22.3.4 \resizebox

Synopses:

```
\resizebox{horizontal length}{vertical length}{material}
\resizebox*{horizontal length}{vertical length}{material}
```

Given a size, such as `3cm`, transform *material* to make it that size. If either *horizontal length* or *vertical length* is an exclamation point `!` then the other argument is used to determine a scale factor for both directions.

This example makes the graphic be a half inch wide and scales it vertically by the same factor to keep it from being distorted.

```
\resizebox{0.5in}{!}{\includegraphics{lion}}
```

The unstarred form `\resizebox` takes *vertical length* to be the box's height while the starred form `\resizebox*` takes it to be height+depth. For instance, make the text have a height+depth of a quarter inch with `\resizebox*{!}{0.25in}{\parbox{1in}{This box has both height and depth.}}`.

You can use `\depth`, `\height`, `\totalheight`, and `\width` to refer to the original size of the box. Thus, make the text two inches wide but keep the original height with `\resizebox{2in}{\height}{Two inches}`.

23 Special insertions

LaTeX provides commands for inserting characters that have a special meaning do not correspond to simple characters you can type.

23.1 Reserved characters

LaTeX sets aside the following characters for special purposes (for example, the percent sign `%` is for comments) so they are called *reserved characters* or *special characters*.

```
# $ % & { } _ ~ ^ \
```

If you want a reserved character to be printed as itself, in the text body font, for all but the final three characters in that list simply put a backslash `\` in front of the character. Thus, `\$1.23` will produce `$1.23` in your output.

As to the last three characters, to get a tilde in the text body font use `\~{}` (omitting the curly braces would result in the next character receiving a tilde accent). Similarly, to get a get a text body font circumflex use `\^{}`. A text body font backslash results from `\textbackslash{}`.

To produce the reserved characters in a typewriter font use `\verb!!`, as below.

```
\begin{center}
  \# \$ \% \& \{ \} \_ \~{} \^{} \textbackslash \\
  \verb!# $ % & { } _ ~ ^ \!
\end{center}
```

In that example the double backslash `\\` is only there to split the lines.

23.2 Upper and lower case

Synopsis:

```
\uppercase{text}
\lowercase{text}
\MakeUppercase{text}
\MakeLowercase{text}
```

Change the case of characters. The TeX primitives commands `\uppercase` and `\lowercase` only work for American characters. The LaTeX commands `\MakeUppercase` and `\MakeLowercase` commands also change characters accessed by commands such as `\ae` or `\aa`. The commands `\MakeUppercase` and `\MakeLowercase` are robust but they have moving arguments (see Section 12.9 [\protect], page 84).

These commands do not change the case of letters used in the name of a command within *text*. But they do change the case of every other Latin letter inside the argument *text*. Thus, `\MakeUppercase{Let $y=f(x)$}` produces '`LET Y=F(X)`'. Another example is that the name of an environment will be changed, so that `\MakeUppercase{\begin{tabular} ... \end{tabular}}` will produce an error because the first half is changed to `\begin{TABULAR}`.

LaTeX uses the same fixed table for changing case throughout a document, The table used is designed for the font encoding T1; this works well with the standard TeX fonts for all Latin alphabets but will cause problems when using other alphabets.

To change the case of text that results from a macro inside *text* you need to do expansion. Here the `\Schoolname` produces '`COLLEGE OF MATHEMATICS`'.

```
\newcommand{\schoolname}{College of Mathematics}
\newcommand{\Schoolname}{\expandafter\MakeUppercase
                         \expandafter{\schoolname}}
```

The `textcase` package brings some of the missing feature of the standard LaTeX commands `\MakeUppercase` and `\MakeLowerCase`.

To uppercase only the first letter of words, you can use the package `mfirstuc`.

23.3 Symbols by font position

You can access any character of the current font using its number with the `\symbol` command. For example, the visible space character used in the `\verb*` command has the code decimal 32, so it can be typed as `\symbol{32}`.

You can also specify numbers in octal (base 8) by using a `'` prefix, or hexadecimal (base 16) with a `"` prefix, so the previous example could also be written as `\symbol{'40}` or `\symbol{"20}`.

23.4 Text symbols

LaTeX provides commands to generate a number of non-letter symbols in running text. Some of these, especially the more obscure ones, are not available in OT1; you may need to load the `textcomp` package.

`\copyright`
`\textcopyright`
: The copyright symbol, ©.

`\dag`
: The dagger symbol (in text).

`\ddag`
: The double dagger symbol (in text).

`\LaTeX`
: The LaTeX logo.

`\LaTeXe`
: The LaTeX2e logo.

`\guillemotleft` («)
`\guillemotright` (»)
`\guilsinglleft` (‹)
`\guilsinglright` (›)
: Double and single angle quotation marks, commonly used in French: «, », ‹, ›.

`\ldots`
`\dots`
`\textellipsis`
: An ellipsis (three dots at the baseline): '. . .'. `\ldots` and `\dots` also work in math mode.

`\lq`
: Left (opening) quote: '.

`\P`
`\textparagraph`
: Paragraph sign (pilcrow): ¶.

`\pounds`
`\textsterling`
: English pounds sterling: £.

`\quotedblbase` („)
`\quotesinglbase` (‚)
: Double and single quotation marks on the baseline: „ and ‚.

`\rq`
: Right (closing) quote: ’.

`\S`
: \itemx \textsection Section sign: §.

`\TeX`
: The TEX logo.

`\textasciicircum`
: ASCII circumflex: ^.

`\textasciitilde`
: ASCII tilde: ~.

`\textasteriskcentered`
: Centered asterisk: *.

`\textbackslash`
: Backslash: \.

`\textbar`
: Vertical bar: |.

`\textbardbl`
: Double vertical bar.

`\textbigcircle`
: Big circle symbol.

`\textbraceleft`
: Left brace: {.

`\textbraceright`
: Right brace: }.

`\textbullet`
: Bullet: •.

`\textcircled{`*letter*`}`
: *letter* in a circle, as in ®.

`\textcompwordmark`
`\textcapitalcompwordmark`
`\textascendercompwordmark`
: Composite word mark (invisible). The `\textcapital...` form has the cap height of the font, while the `\textascender...` form has the ascender height.

`\textdagger`
: Dagger: †.

`\textdaggerdbl`
: Double dagger: ‡.

`\textdollar` (or `\$`)
: Dollar sign: $.

`\textemdash` (or `---`)
: Em-dash: — (for punctuation).

`\textendash` (or `--`)
: En-dash: – (for ranges).

`\texteuro`
: The Euro symbol: €.

`\textexclamdown` (or `!‘`)
: Upside down exclamation point: ¡.

`\textgreater`
: Greater than: >.

`\textless`
: Less than: <.

`\textleftarrow`
: Left arrow.

`\textordfeminine`
`\textordmasculine`
: Feminine and masculine ordinal symbols: $\underline{\text{a}}$, $\underline{\text{o}}$.

`\textperiodcentered`
: Centered period: ·.

`\textquestiondown` (or `?‘`)
: Upside down question mark: ¿.

`\textquotedblleft` (or ` ‘‘`)
: Double left quote: “.

`\textquotedblright` (or `’’`)
: Double right quote: ”.

`\textquoteleft` (or ` ‘`)
: Single left quote: ‘.

`\textquoteright` (or `’`)
: Single right quote: ’.

`\textquotesingle`
: Straight single quote. (From TS1 encoding.)

`\textquotestraightbase`
`\textquotestraightdblbase`
: Single and double straight quotes on the baseline.

`\textregistered`
: Registered symbol: ®.

`\textrightarrow`
: Right arrow.

`\textthreequartersemdash`
: "Three-quarters" em-dash, between en-dash and em-dash.

`\texttrademark`
: Trademark symbol: TM.

`\texttwelveudash`
: "Two-thirds" em-dash, between en-dash and em-dash.

`\textunderscore`
: Underscore: _.

`\textvisiblespace`
: Visible space symbol.

23.5 Accents

LaTeX has wide support for many of the world's scripts and languages, through the `babel` package and related support. This section does not attempt to cover all that support. It merely lists the core LaTeX commands for creating accented characters.

The `\capital...` commands produce alternative forms for use with capital letters. These are not available with OT1.

`\"`
`\capitaldieresis`
: Produces an umlaut (dieresis), as in ö.

`\'`
`\capitalacute`
: Produces an acute accent, as in ó. In the `tabbing` environment, pushes current column to the right of the previous column (see Section 8.21 [tabbing], page 59).

`\.`
: Produces a dot accent over the following, as in ȯ.

`\=`
`\capitalmacron`
: Produces a macron (overbar) accent over the following, as in ō.

`\^`
`\capitalcircumflex`
: Produces a circumflex (hat) accent over the following, as in ô.

`\``
`\capitalgrave`
: Produces a grave accent over the following, as in ò. In the `tabbing` environment, move following text to the right margin (see Section 8.21 [tabbing], page 59).

`\~`
`\capitaltilde`
: Produces a tilde accent over the following, as in ñ.

`\b`
: Produces a bar accent under the following, as in o̲. See also `\underbar` hereinafter.

`\c`
`\capitalcedilla`
: Produces a cedilla accent under the following, as in ç.

`\d`
`\capitaldotaccent`
: Produces a dot accent under the following, as in ọ.

`\H`
`\capitalhungarumlaut`
: Produces a long Hungarian umlaut accent over the following, as in ő.

`\i` : Produces a dotless i, as in 'ɪ'.

`\j` : Produces a dotless j, as in 'ɉ'.

`\k`
`\capitalogonek`
: Produces a letter with ogonek, as in 'ǫ'. Not available in the OT1 encoding.

`\r`
`\capitalring`
: Produces a ring accent, as in 'o͚'.

`\t`
`\capitaltie`
`\newtie`
`\capitalnewtie`
: Produces a tie-after accent, as in 'o͡o'. The `\newtie` form is centered in its box.

`\u`
`\capitalbreve`
: Produces a breve accent, as in 'ŏ'.

`\underbar`
: Not exactly an accent, this produces a bar under the argument text. The argument is always processed in horizontal mode. The bar is always a fixed position under the baseline, thus crossing through descenders. See also `\underline` in Section 16.6 [Math miscellany], page 107. See also `\b` above.

`\v`
`\capitalcaron`
: Produces a háček (check, caron) accent, as in 'ǒ'.

23.6 Additional Latin letters

Here are the basic LaTeX commands for inserting letters (beyond A–Z) extending the Latin alphabet, used primarily in languages other than English.

`\aa`
`\AA` : å and Å.

`\ae`
`\AE` : æ and Æ.

`\dh`
`\DH` Icelandic letter eth: ð and Ð. Not available with OT1 encoding, you need the `fontenc` package to select an alternate font encoding, such as T1.

`\dj`
`\DJ` Crossed d and D, a.k.a. capital and small letter d with stroke. Not available with OT1 encoding, you need the `fontenc` package to select an alternate font encoding, such as T1.

`\ij`
`\IJ` ij and IJ (except somewhat closer together than appears here).

`\l`
`\L` ł and Ł.

`\ng`
`\NG` Lappish letter eng, also used in phonetics.

`\o`
`\O` ø and Ø.

`\oe`
`\OE` œ and Œ.

`\ss`
`\SS` ß and SS.

`\th`
`\TH` Icelandic letter thorn: þ and Þ. Not available with OT1 encoding, you need the `fontenc` package to select an alternate font encoding, such as T1.

23.7 `\rule`

Synopsis:

```
\rule[raise]{width}{thickness}
```

The `\rule` command produces *rules*, that is, lines or rectangles. The arguments are:

raise How high to raise the rule (optional).

width The length of the rule (mandatory).

thickness The thickness of the rule (mandatory).

23.8 `\today`

The `\today` command produces today's date, in the format '*`month dd, yyyy`*'; for example, '`July 4, 1976`'. It uses the predefined counters `\day`, `\month`, and `\year` (see Section 13.8 [\day \month \year], page 89) to do this. It is not updated as the program runs.

Multilingual packages like `babel` or classes like `lettre`, among others, will localize `\today`. For example, the following will output '`4 juillet 1976`':

```
\year=1976 \month=7 \day=4
\documentclass{minimal}
\usepackage[french]{babel}
```

```
\begin{document}
\today
\end{document}
```

The `datetime` package, among others, can produce a wide variety of other date formats.

24 Splitting the input

A large document requires a lot of input. Rather than putting the whole input in a single large file, it's more efficient to split it into several smaller ones. Regardless of how many separate files you use, there is one that is the *root file*; it is the one whose name you type when you run LaTeX.

See Section 8.11 [filecontents], page 45, for an environment that allows bundling an external file to be created with the main document.

24.1 \include

Synopsis:

```
\include{file}
```

If no `\includeonly` command is present, the `\include` command executes `\clearpage` to start a new page (see Section 10.2 [\clearpage], page 73), then reads *file*, then does another `\clearpage`.

Given an `\includeonly` command, the `\include` actions are only run if *file* is listed as an argument to `\includeonly`. See Section 24.2 [\includeonly], page 145.

The `\include` command may not appear in the preamble or in a file read by another `\include` command.

24.2 \includeonly

Synopsis:

```
\includeonly{file1,file2,...}
```

The `\includeonly` command controls which files will be read by subsequent `\include` commands. The list of filenames is comma-separated. Each element *file1*, *file2*, ... must exactly match a filename specified in a `\include` command for the selection to be effective.

This command can only appear in the preamble.

24.3 \input

Synopsis:

```
\input{file}
```

The `\input` command causes the specified *file* to be read and processed, as if its contents had been inserted in the current file at that point.

If *file* does not end in '`.tex`' (e.g., '`foo`' or '`foo.bar`'), it is first tried with that extension ('`foo.tex`' or '`foo.bar.tex`'). If that is not found, the original *file* is tried ('`foo`' or '`foo.bar`').

25 Front/back matter

25.1 Tables of contents

A table of contents is produced with the `\tableofcontents` command. You put the command right where you want the table of contents to go; LaTeX does the rest for you. A previous run must have generated a `.toc` file.

The `\tableofcontents` command produces a heading, but it does not automatically start a new page. If you want a new page after the table of contents, write a `\newpage` command after the `\tableofcontents` command.

The analogous commands `\listoffigures` and `\listoftables` produce a list of figures and a list of tables (from `.lof` and `.lot` files), respectively. Everything works exactly the same as for the table of contents.

The command `\nofiles` overrides these commands, and *prevents* any of these lists from being generated.

25.1.1 `\addcontentsline`

Synopsis:

```
\addcontentsline{ext}{unit}{text}
```

The `\addcontentsline` command adds an entry to the specified list or table where:

ext The filename extension of the file on which information is to be written, typically one of: `toc` (table of contents), `lof` (list of figures), or `lot` (list of tables).

unit The name of the sectional unit being added, typically one of the following, matching the value of the *ext* argument:

- `toc` The name of the sectional unit: `part`, `chapter`, `section`, `subsection`, `subsubsection`.
- `lof` For the list of figures: `figure`.
- `lot` For the list of tables: `table`.

text The text of the entry.

What is written to the `.ext` file is the command `\contentsline{unit}{text}{num}`, where *num* is the current value of counter *unit*.

25.1.2 `\addtocontents`

The `\addtocontents{ext}{text}` command adds text (or formatting commands) directly to the `.ext` file that generates the table of contents or lists of figures or tables.

ext The extension of the file on which information is to be written, typically one of: `toc` (table of contents), `lof` (list of figures), or `lot` (list of tables).

text The text to be written.

25.2 Glossaries

The command `\makeglossary` enables creating glossaries.

The command `\glossary{`*text*`}` writes a glossary entry for *text* to an auxiliary file with the `.glo` extension.

Specifically, what gets written is the command `\glossaryentry{`*text*`}{`*pageno*`}`, where *pageno* is the current `\thepage` value.

The `glossary` package on CTAN provides support for fancier glossaries.

25.3 Indexes

The command `\makeindex` enables creating indexes. Put this in the preamble.

The command `\index{`*text*`}` writes an index entry for *text* to an auxiliary file named with the `.idx` extension.

Specifically, what gets written is the command `\indexentry{`*text*`}{`*pageno*`}`, where *pageno* is the current `\thepage` value.

To generate a index entry for 'bar' that says 'See foo', use a vertical bar: `\index{bar|see{foo}}`. Use `seealso` instead of `see` to make a 'See also' entry.

The text 'See' is defined by the macro `\seename`, and 'See also' by the macro `\alsoname`. These can be redefined for other languages.

The generated `.idx` file is then sorted with an external command, usually either `makeindex` (`http://mirror.ctan.org/indexing/makeindex`) or (the multi-lingual) `xindy` (`http://xindy.sourceforge.net`). This results in a `.ind` file, which can then be read to typeset the index.

The index is usually generated with the `\printindex` command. This is defined in the `makeidx` package, so `\usepackage{makeidx}` needs to be in the preamble.

The rubber length `\indexspace` is inserted before each new letter in the printed index; its default value is '`10pt plus5pt minus3pt`'.

The `showidx` package causes each index entries to be shown in the margin on the page where the entry appears. This can help in preparing the index.

The `multind` package supports multiple indexes. See also the TEX FAQ entry on this topic, `http://www.tex.ac.uk/cgi-bin/texfaq2html?label=multind`.

26 Letters

Synopsis:

```
\documentclass{letter}
\address{sender address}
\signature{sender name}
\begin{document}
\begin{letter}{recipient address}
\opening{salutation}
  letter body
\closing{closing text}
\end{letter}
...  more letters ...
\end{document}
```

Produce one or more letters.

Each letter is in a separate `letter` environment, whose argument *recipient address* often contains multiple lines separated with a double backslash (`\\`). For example, you might have:

```
\begin{letter}{Mr. Joe Smith \\
     2345 Princess St. \\
     Edinburgh, EH1 1AA}
  ...
\end{letter}
```

The start of the `letter` environment resets the page number to 1, and the footnote number to 1 also.

The *sender address* and *sender name* are common to all of the letters, whether there is one or more, so these are best put in the preamble. As with the recipient address, often *sender address* contains multiple lines separated by a double backslash (`\\`). LaTeX will put the *sender name* under the closing, after a vertical space for the traditional hand-written signature; it also can contain multiple lines.

Each `letter` environment body begins with a required `\opening` command such as `\opening{Dear Madam or Sir:}`. The *letter body* text is ordinary LaTeX so it can contain everything from enumerated lists to displayed math, except that commands such as `\chapter` that make no sense in a letter are turned off. Each `letter` environment body typically ends with a `\closing` command such as `\closing{Yours,}`.

Additional material may come after the `\closing`. You can say who is receiving a copy of the letter with a command like `\cc{the Boss \\ the Boss's Boss}`. There's a similar `\encl` command for a list of enclosures. And, you can add a postscript with `\ps`.

LaTeX's default is to indent the signature and the `\closing` above it by a length of `\longindentation`. By default this is `0.5\textwidth`. To make them flush left, put `\setlength{\longindentation}{0em}` in your preamble.

To set a fixed date use something like `\renewcommand{\today}{2015-Oct-12}`. If put in your preamble then it will apply to all the letters.

This example shows only one `letter` environment. The three lines marked as optional are typically omitted.

```
\documentclass{letter}
\address{Sender's street \\ Sender's town}
\signature{Sender's name \\ Sender's title}
% optional: \location{Mailbox 13}
% optional: \telephone{(102) 555-0101}
\begin{document}
\begin{letter}{Recipient's name \\ Recipient's address}
\opening{Sir:}
% optional: \thispagestyle{firstpage}
I am not interested in entering a business arrangement with you.
\closing{Your most humble, etc.,}
\end{letter}
\end{document}
```

These commands are used with the `letter` class.

26.1 \address

Synopsis:

```
\address{senders address}
```

Specifies the return address as it appears on the letter and on the envelope. Separate multiple lines in *senders address* with a double backslash `\\`.

Because it can apply to multiple letters this declaration is often put in the preamble. However, it can go anywhere, including inside an individual `letter` environment.

This command is optional: without the `\address` declaration the letter is formatted with some blank space on top, for copying onto pre-printed letterhead paper. (See Chapter 2 [Overview], page 3, for details on your local implementation.) With the `\address` declaration, it is formatted as a personal letter.

Here is an example.

```
\address{Stephen Maturin \\
         The Grapes of the Savoy}
```

26.2 \cc

Synopsis:

```
\cc{first name \\
    ... }
```

Produce a list of names to which copies of the letter were sent. This command is optional. If it appears then typically it comes after `\closing`. Separate multiple lines with a double backslash `\\`, as in:

```
\cc{President \\
    Vice President}
```

26.3 \closing

Synopsis:

```
\closing{text}
```

Usually at the end of a letter, above the handwritten signature, there is a `\closing` (although this command is optional). For example,

```
\closing{Regards,}
```

26.4 \encl

Synopsis:

```
\encl{first enclosed object \\
      ... }
```

Produce a list of things included with the letter. This command is optional; when it is used, it typically is put after `\closing`. Separate multiple lines with a double backslash `\\`.

```
\encl{License \\
      Passport }
```

26.5 \location

Synopsis:

```
\location{text}
```

The *text* appears centered at the bottom of the each page. It only appears if the page style is `firstpage`.

26.6 \makelabels

Synopsis:

```
\makelabels
```

Create a sheet of address labels from the recipient addresses, one for each letter. This sheet will be output before the letters, with the idea that you can copy it to a sheet of peel-off labels. This command goes in the preamble.

Customize the labels by redefining the commands `\startlabels`, `\mlabel`, and `\returnaddress` in the preamble. The command `\startlabels` sets the width, height, number of columns, etc., of the page onto which the labels are printed. The command `\mlabel{`*sender address*`}{`*recipient address*`}` produces the two labels (or one, if you choose to ignore the *sender address*). The *sender address* is the value returned by the macro `\returnaddress` while *recipient address* is the value passed in the argument to the `letter` environment. By default `\mlabel` ignores the first argument, the *sender address*.

26.7 \name

Synopsis:

```
\name{name}
```

Sender's name, used for printing on the envelope together with the return address.

26.8 \opening

Synopsis:

```
\opening{text}
```

This command is required. It starts a letter, following the `\begin{letter}{...}`. The mandatory argument *text* is the text that starts your letter. For instance:

```
\opening{Dear John:}
```

26.9 \ps

Synopsis:

```
\ps{text}
```

Add a postscript. This command is optional and usually is used after `\closing`.

```
\ps{P.S. After you have read this letter, burn it. Or eat it.}
```

26.10 \signature

Synopsis:

```
\signature{first line \\
           ... }
```

The sender's name. This command is optional, although its inclusion is usual.

The argument text appears at the end of the letter, after the closing and after a vertical space for the traditional hand-written signature. Separate multiple lines with a double backslash `\\`. For example:

```
\signature{J Fred Muggs \\
           White House}
```

LaTeX's default for the vertical space from the `\closing` text down to the `\signature` text is `6\medskipamount`, which is six times 0.7 em.

This command is usually in the preamble, to apply to all the letters in the document. To have it apply to one letter only, put it inside a `letter` environment and before the `\closing`.

You can include a graphic in the signature, for instance with `\signature{\vspace{-6\medskipamount}\inc My name}` (this requires writing `\usepackage{graphicx}` in the preamble).

26.11 \telephone

Synopsis:

```
\telephone{number}
```

The sender's telephone number. This is typically in the preamble, where it applies to all letters. This only appears if the `firstpage` pagestyle is selected. If so, it appears on the lower right of the page.

27 Terminal input/output

27.1 \typein[*cmd*]{*msg*}

Synopsis:

```
\typein[\cmd]{msg}
```

`\typein` prints *msg* on the terminal and causes LaTeX to stop and wait for you to type a line of input, ending with return. If the optional `\cmd` argument is omitted, the typed input is processed as if it had been included in the input file in place of the `\typein` command. If the `\cmd` argument is present, it must be a command name. This command name is then defined or redefined to be the typed input.

27.2 \typeout{*msg*}

Synopsis:

```
\typeout{msg}
```

Prints `msg` on the terminal and in the `log` file. Commands in `msg` that are defined with `\newcommand` or `\renewcommand` (among others) are replaced by their definitions before being printed.

LaTeX's usual rules for treating multiple spaces as a single space and ignoring spaces after a command name apply to `msg`. A `\space` command in `msg` causes a single space to be printed, independent of surrounding spaces. A `^^J` in `msg` prints a newline.

28 Command line

The input file specification indicates the file to be formatted; TEX uses `.tex` as a default file extension. If you omit the input file entirely, TEX accepts input from the terminal. You can also specify arbitrary LATEX input by starting with a backslash. For example, this processes `foo.tex` without pausing after every error:

```
latex '\nonstopmode\input foo.tex'
```

With many, but not all, implementations, command-line options can also be specified in the usual Unix way, starting with '`-`' or '`--`'. For a list of those options, try '`latex --help`'.

If LATEX stops in the middle of the document and gives you a '`*`' prompt, it is waiting for input. You can type `\stop` (and return) and it will prematurely end the document.

See Section 2.3 [TEX engines], page 4, for other system commands invoking LATEX.

Appendix A Document templates

Although not reference material, perhaps these document templates will be useful. Additional template resources are listed at `http://tug.org/interest.html#latextemplates`.

A.1 `beamer` template

The `beamer` class creates presentation slides. It has a vast array of features, but here is a basic template:

```
\documentclass{beamer}

\title{Beamer Class template}
\author{Alex Author}
\date{July 31, 2007}

\begin{document}

\maketitle

% without [fragile], any {verbatim} code gets mysterious errors.
\begin{frame}[fragile]
 \frametitle{First Slide}

\begin{verbatim}
  This is \verbatim!
\end{verbatim}

\end{frame}

\end{document}
```

One web resource for this: `http://robjhyndman.com/hyndsight/beamer/`.

A.2 `book` template

```
\documentclass{book}
\title{Book Class Template}
\author{Alex Author}

\begin{document}
\maketitle

\chapter{First}
Some text.

\chapter{Second}
Some other text.
```

```
\section{A subtopic}
The end.
\end{document}
```

A.3 tugboat template

TUGboat is the journal of the TEX Users Group, http://tug.org/TUGboat.

```
\documentclass{ltugboat}

\usepackage{graphicx}
\usepackage{ifpdf}
\ifpdf
\usepackage[breaklinks,hidelinks]{hyperref}
\else
\usepackage{url}
\fi

%%% Start of metadata %%%

\title{Example \TUB\ article}

% repeat info for each author.
\author{First Last}
\address{Street Address \\ Town, Postal \\ Country}
\netaddress{user (at) example dot org}
\personalURL{http://example.org/~user/}

%%% End of metadata %%%

\begin{document}

\maketitle

\begin{abstract}
This is an example article for \TUB{}.
Please write an abstract.
\end{abstract}

\section{Introduction}

This is an example article for \TUB, linked from
\url{http://tug.org/TUGboat/location.html}.

We recommend the \texttt{graphicx} package for image inclusions, and the
\texttt{hyperref} package if active urls are desired (in the \acro{PDF}
output).  Nowadays \TUB\ is produced using \acro{PDF} files exclusively.
```

```
The \texttt{ltugboat} class provides these abbreviations (and many more):
% verbatim blocks are often better in \small
\begin{verbatim}[\small]
\AllTeX \AMS \AmS \AmSLaTeX \AmSTeX \aw \AW
\BibTeX \CTAN \DTD \HTML
\ISBN \ISSN \LaTeXe
\mf \MFB
\plain \POBox \PS
\SGML \TANGLE \TB \TP
\TUB \TUG \tug
\UNIX \XeT \WEB \WEAVE

\, \bull \Dash \dash \hyph

\acro{FRED} -> {\small[er] fred}  % please use!
\cs{fred}   -> \fred
\meta{fred} -> <fred>
\nth{n}     -> 1st, 2nd, ...
\sfrac{3/4} -> 3/4
\booktitle{Book of Fred}
\end{verbatim}

For references to other \TUB\ issue, please use the format
\textsl{volno:issno}, e.g., ``\TUB\ 32:1'' for our \nth{100} issue.

This file is just a template.  The \TUB\ style documentation is the
\texttt{ltubguid} document at \url{http://ctan.org/pkg/tugboat}.  (For
\CTAN\ references, where sensible we recommend that form of url, using
\texttt{/pkg/}; or, if you need to refer to a specific file location,
\texttt{http://mirror.ctan.org/\textsl{path}}.)

Email \verb|tugboat@tug.org| if problems or questions.

\bibliographystyle{plain}  % we recommend the plain bibliography style
\nocite{book-minimal}      % just making the bibliography non-empty
\bibliography{xampl}       % xampl.bib comes with BibTeX

\makesignature
\end{document}
```

Concept Index

*

'*' prompt 153
*-form of environment commands 80
*-form of sectioning commands 31
*-form, defining new commands 78

.

`.glo` file 147
`.idx` file 147
`.ind` file 147

'

'see' and 'see also' index entries 147

A

abstracts 37
accents 141
accents, mathematical 106
accessing any character of a font 138
acute accent 141
acute accent, math 106
additional packages, loading 9
ae ligature 142
`algorithm2e` package 61
align environment, from `amsmath` 43
aligning equations 43
alignment via tabbing 59
`amsmath` package 38, 41
amsmath package, replacing `eqnarray` 43
appendix, creating 31
aring 142
arrays, math 38
arrow, left, in text 140
arrow, right, in text 140
ascender height 139
ASCII circumflex, in text 139
ASCII tilde, in text 139
asterisk, centered, in text 139
at clause, in font definitions 84
author, for titlepage 112
auxiliary file 4

B

`babel` package 67, 141
background, colored 125
backslash, in text 139
bar, double vertical, in text 139
bar, vertical, in text 139
bar-over accent 141
bar-over accent, math 106
bar-under accent 141
basics of LaTeX 3
`beamer` template and class 154
beginning of document hook 41
bibliography format, open 9
bibliography, creating (automatically) 68
bibliography, creating (manually) 67
bibTeX, using 68
big circle symbols, in text 139
Big point 90
black boxes, omitting 9
bold font 18
bold math 18
bold typewriter, avoiding 40
box, allocating new 80
box, colored 124
boxes 119
brace, left, in text 139
brace, right, in text 139
breaking lines 71
breaking pages 73
breve accent 142
breve accent, math 106
bug reporting 2
bullet symbol 96
bullet, in text 139
bulleted lists 47

C

calligraphic letters for math 18
cap height 139
caron accent 142
catcode 6
category code, character 6
cc list, in letters 149
cedilla accent 142
centered asterisk, in text 139
centered equations 9
centered period, in text 140
centering text, declaration for 39
centering text, environment for 39
Centimeter 90
character category code 6
characters, accented 141
characters, case 137

characters, non-English 142
characters, reserved 137
characters, special 137
check accent 142
check accent, math 106
Cicero 90
circle symbol, big, in text 139
circled letter, in text 139
circumflex accent 141
circumflex accent, math 107
circumflex, ASCII, in text 139
citation key 67
class and package commands 11
class and package difference 10
class and package structure 10
class file example 10
class file layout 10
class options 8, 10, 12
classes of documents 8
closing letters 150
closing quote 139
code, typesetting 69
color 122, 123, 124, 125
color models 122
color package commands 123
color package options 122
color, define 123
colored boxes 124
colored page 125
colored text 123
command line 153
command syntax 5
commands, class and package 11
commands, defining new ones 78, 79
commands, document class 10
commands, graphics package 130
commands, ignore spaces 85
commands, redefining 78
commands, star-variants 7
composite word mark, in text 139
computer programs, typesetting 69
configuration, graphics package 127
contents file 4
copyright symbol 138
counters, a list of 87
counters, defining new 79
counters, getting value of 88
counters, printing 87
counters, setting 89
creating pictures 54
creating tables 61
credit footnote 112
cross references 35
cross references, resolving 4
cross referencing with page number 36
cross referencing, symbolic 36
currency, dollar 140
currency, euro 140

D

dagger, double, in text 139
dagger, in text 138, 139
date, for titlepage 112
date, today's 143
`datetime` package 144
define color 123
defining a new command 78, 79
defining new environments 80
defining new fonts 84
defining new theorems 82
definitions 78
description lists, creating 40
design size, in font definitions 84
Didot point 90
dieresis accent 141
difference between class and package 10
discretionary hyphenation 72
discretionary multiplication 107
displaying quoted text with paragraph indentation 58
displaying quoted text without paragraph indentation 58
document class commands 10
document class options 8
document class, defined 3
document classes 8
document templates 154
dollar sign 140
dot accent 141
dot over accent, math 106
dot-over accent 141
dot-under accent 142
dotless i 142
dotless i, math 107
dotless j 142
dotless j, math 107
double angle quotation marks 138
double dagger, in text 138, 139
double dot accent, math 106
double guillemets 138
double left quote 140
double low-9 quotation mark 139
double quote, straight base 140
double right quote 140
double spacing 21
double vertical bar, in text 139

E

e-dash ... 140
e-TEX ... 4
ellipsis ... 138
em ... 90
em-dash ... 140
em-dash, three-quarters ... 141
em-dash, two-thirds ... 141
emphasis ... 18
enclosure list ... 150
end of document hook ... 42
ending and starting ... 3
engines, TEX ... 4
enlarge current page ... 73
`enumitem` package ... 48
environment ... 3
environment, theorem-like ... 82
environments ... 37
environments, defining ... 80
EPS files ... 127, 130
equation number, cross referencing ... 36
equation numbers, left vs. right ... 9
equation numbers, omitting ... 43
equations, aligning ... 43
equations, environment for ... 44
equations, flush left vs. centered ... 9
es-zet German letter ... 143
eth, Icelandic letter ... 143
`etoolbox` package ... 12
euro symbol ... 140
ex ... 90
exclamation point, upside-down ... 140
exponent ... 94
extended Latin ... 142
external files, writing ... 45

F

families, of fonts ... 19
`fancyvrb` package ... 61
feminine ordinal symbol ... 140
figure number, cross referencing ... 36
figures, footnotes in ... 54
figures, inserting ... 44
file, root ... 145
fixed-width font ... 18
`flafter` package ... 28
`float` package ... 28
float page ... 28
flush left equations ... 9
flushing floats and starting a page ... 73
font catalogue ... 19
font commands, low-level ... 19
font size ... 21
font sizes ... 19
font styles ... 17
font symbols, by number ... 138
fonts ... 17
fonts, new commands for ... 84
footer style ... 112
footer, parameters for ... 25
`footmisc` package ... 76
footnote number, cross referencing ... 36
footnote parameters ... 77
footnotes in figures ... 54
footnotes, creating ... 74
Footnotes, in a minipage ... 74
Footnotes, in a table ... 75
footnotes, in section headings ... 76
footnotes, symbols instead of numbers ... 74
formulas, environment for ... 44
formulas, math ... 94
forward reference ... 35
forward references, resolving ... 4
fragile commands ... 84
French quotation marks ... 138
functions, math ... 105

G

`geometry` package ... 9
global options ... 8, 10
glossaries ... 147
glossary package ... 147
glue register, plain TEX ... 80
graphics ... 126, 127, 130
graphics package ... 126, 127, 130
graphics package commands ... 130
graphics package options ... 126
graphics packages ... 56
graphics, resizing ... 136
graphics, scaling ... 136
grave accent ... 141
grave accent, math ... 106
greater than symbol, in text ... 140
greek letters ... 95
group, and environments ... 37

H

hacek accent ... 142
háček accent, math ... 106
hat accent ... 141
hat accent, math ... 107
header style ... 112
header, parameters for ... 25
hello, world ... 3
here, putting floats ... 28
hungarian umlaut accent ... 142
hyphenation, defining ... 72
hyphenation, discretionary ... 72
hyphenation, forcing ... 72
hyphenation, preventing ... 119

I

Icelandic eth ... 143
Icelandic thorn ... 143
ij letter, Dutch ... 143
implementations of TEX ... 4
importing graphics ... 130
in-line formulas ... 54
including graphics ... 130
indent, forcing ... 92
indent, suppressing ... 92
indentation of paragraphs, in minipage ... 54
index entries, 'see' and 'see also' ... 147
indexes ... 147
infinite horizontal stretch ... 114
infinite vertical stretch ... 117
input file ... 145
input/output, to terminal ... 152
inserting figures ... 44
insertions of special characters ... 137
italic correction ... 116
italic font ... 18

J

JPEG files ... 127, 130
JPG files ... 127, 130
justification, ragged left ... 46
justification, ragged right ... 46

K

Knuth, Donald E. ... 3

L

label ... 35
labelled lists, creating ... 40
Lamport TEX ... 3
Lamport, Leslie ... 3
landscape orientation ... 9
LATEX logo ... 138
LATEX overview ... 3
LATEX Project team ... 2
LATEX vs. LATEX2e ... 2
LATEX2e logo ... 138
Latin letters, additional ... 142
layout commands ... 23
layout, page parameters for ... 25
left angle quotation marks ... 138
left arrow, in text ... 140
left brace, in text ... 139
left quote ... 138
left quote, double ... 140
left quote, single ... 140
left-hand equation numbers ... 9
left-justifying text ... 46
left-justifying text, environment for ... 46
left-to-right mode ... 110
length command ... 91
lengths, adding to ... 91
lengths, allocating new ... 80
lengths, defining and using ... 90
lengths, predefined ... 91
lengths, setting ... 91
less than symbol, in text ... 140
letters, accented ... 141
letters, additional Latin ... 142
letters, ending ... 150
letters, starting ... 151
letters, writing ... 148
line break, forcing ... 71
line breaking ... 71
line breaks, forcing ... 72
line breaks, preventing ... 72
lines in tables ... 62
lining numerals ... 18
lining text up in tables ... 62
lining text up using tab stops ... 59
list items, specifying counter ... 88
list of figures file ... 4
list of tables file ... 4
`listings` package ... 61
lists of items ... 47
lists of items, generic ... 48
lists of items, numbered ... 42
loading additional packages ... 9
log file ... 4
logo, LATEX ... 138
logo, LATEX2e ... 138
logo, TEX ... 139
long command ... 11
low-9 quotation marks, single and double ... 139
low-level font commands ... 19
Lower case ... 137
LR mode ... 110
`ltugboat` class ... 155
LuaTEX ... 4

M

m-width ... 90
macro package, LATEX as ... 3
macron accent ... 141
macron accent, math ... 106
`macros2e` package ... 6
Madsen, Lars ... 43
`makeidx` package ... 147
`makeindex` program ... 147
making a title page ... 69
making paragraphs ... 92
marginal notes ... 92
masculine ordinal symbol ... 140
math accents ... 106
math formulas ... 94
math functions ... 105
math miscellany ... 107

math mode ... 110
math mode, entering ... 94
math mode, spacing ... 107
math symbols ... 95
math, bold ... 18
`mfirstuc` package ... 138
Millimeter ... 90
minipage, creating a ... 54
`minted` package ... 61
modes ... 110
monospace font ... 18
moving arguments ... 84
mpfootnote counter ... 74
mu, math unit ... 90
multicolumn text ... 23
multilingual support ... 141
`multind` package ... 147
multiplication symbol,
discretionary line break ... 107

N

nested `\include`, not allowed ... 145
new class commands ... 10
new command, check ... 11
new command, definition ... 12
new commands, defining ... 78, 79
new line, output as input ... 71
new line, starting ... 71
new line, starting (paragraph mode) ... 71
new page, starting ... 73
non-English characters ... 142
notes in the margin ... 92
null delimiter ... 108
numbered items, specifying counter ... 88
numerals, old-style ... 18

O

oblique font ... 18
oe ligature ... 143
ogonek ... 142
old-style numerals ... 18
one-column output ... 23
opening quote ... 138
OpenType fonts ... 4
options, class ... 12
options, color package ... 122
options, document class ... 8, 10
options, global ... 10
options, graphics package ... 126
options, package ... 10, 12
ordinals, feminine and masculine ... 140
oslash ... 143
overbar accent ... 141
overdot accent, math ... 106
overview of LaTeX ... 3

P

package file layout ... 10
package options ... 10, 12
package, `algorithm2e` ... 61
package, `amsmath` ... 38, 41
package, `babel` ... 67, 141
package, `datetime` ... 144
package, `enumitem` ... 48
package, `etoolbox` ... 12
package, `fancyvrb` ... 61
package, `flafter` ... 28
package, `float` ... 28
package, `footmisc` ... 76
package, `geometry` ... 9
package, `listings` ... 61
package, `macros2e` ... 6
package, `makeidx` ... 147
package, `mfirstuc` ... 138
package, `minted` ... 61
package, `multind` ... 147
package, `picture` ... 55
package, `setspace` ... 21
package, `showidx` ... 147
package, `textcase` ... 138
package, `textcomp` ... 18
package, `xspace` ... 116
packages, loading additional ... 9
page break, forcing ... 73
page break, preventing ... 73
page breaking ... 73
page layout parameters ... 25
page number, cross referencing ... 36
page numbering style ... 112
page styles ... 112
page, colored ... 125
paragraph indentation, in minipage ... 54
paragraph indentations in quoted text ... 58
paragraph indentations in quoted
text, omitting ... 58
paragraph mode ... 110, 120
paragraph symbol ... 138
paragraphs ... 92
parameters, for footnotes ... 77
parameters, page layout ... 25
PDF graphic files ... 127, 130
pdfTeX ... 4
pdfTeX engine ... 4
period, centered, in text ... 140
pica ... 90
`pict2e` package ... 56
`picture` package ... 55
pictures, creating ... 54
pilcrow ... 138
placement of floats ... 28
PNG files ... 127, 130
poetry, an environment for ... 70
Point ... 90
polish l ... 143

portrait orientation 9
position, in picture 55
positional parameter 78
postscript, in letters 151
pounds symbol 139
preamble, defined 3
predefined lengths 91
prompt, '*' 153
pronunciation 3

Q

quad 107
question mark, upside-down 140
quotation marks, French 138
quote, single straight 140
quote, straight base 140
quoted text with paragraph indentation, displaying 58
quoted text without paragraph indentation, displaying 58

R

ragged left text 46
ragged left text, environment for 46
ragged right text 46
ragged right text, environment for 46
redefining environments 80
reference, forward 35
references, resolving forward 4
registered symbol 140
remarks in the margin 92
reporting bugs 2
reserved characters 137
resizing 136
right angle quotation marks 138
right arrow, in text 140
right brace, in text 139
right quote 139
right quote, double 140
right quote, single 140
right-hand equation numbers 9
right-justifying text 46
right-justifying text, environment for 46
ring accent 142
ring accent, math 107
robust commands 84
roman font 18
root file 145
rotating graphics 135
rotating text 135
rotation 135
row, tabbing 59
rubber lengths, defining new 80
running header and footer 25
running header and footer style 112

S

sans serif font 18
Scaled point 90
scaling 136
script letters for math 18
section number, cross referencing 36
section numbers, printing 31
section symbol 139
section, redefining 31
sectioning commands 31
series, of fonts 20
`setspace` package 21
setting counters 89
shapes, of fonts 21
sharp S letters 143
`showidx` package 147
simulating typed text 69
single angle quotation marks 138
single guillemets 138
single left quote 140
single low-9 quotation mark 139
single quote, straight 140
single right quote 140
sizes of text 19
skip register, plain TEX 80
slanted font 18
small caps font 18
space, inserting vertical 117
space, vertical 118
spaces 114
spaces, ignore around commands 85
spacing within math mode 107
spacing, inter-sentence 115, 116
Spanish ordinals, feminine and masculine 140
special characters 137, 142
special insertions 137
specifier, float placement 28
splitting the input file 145
`stable` option to `footmisc` package 76
star-variants, commands 7
starred form, defining new commands 78
starting a new page 73
starting a new page and clearing floats 73
starting and ending 3
starting on a right-hand page 73
sterling symbol 139
straight double quote, base 140
straight quote, base 140
straight single quote 140
stretch, infinite horizontal 114
stretch, infinite vertical 117
stretch, omitting vertical 25
styles of text 17
styles, page 112
subscript 94
superscript 94
symbols, math 95
symbols, text 138

T

tab stops, using ... 59
table of contents entry, manually adding ... 146
table of contents file ... 4
table of contents, avoiding footnotes ... 76
table of contents, creating ... 146
tables, creating ... 61
template, `beamer` ... 154
template, `book` ... 154
template, TUGboat ... 155
templates, document ... 154
terminal input/output ... 152
TEX logo ... 139
text symbols ... 138
text, resizing ... 136
text, scaling ... 136
`textcase` package ... 138
`textcomp` package ... 18
thanks, for titlepage ... 112
theorem-like environment ... 82
theorems, defining ... 82
theorems, typesetting ... 69
thorn, Icelandic letter ... 143
three-quarters em-dash ... 141
tie-after accent ... 142
tilde accent ... 141
tilde accent, math ... 107
tilde, ASCII, in text ... 139
title page, separate or run-in ... 9
title pages, creating ... 69
title, for titlepage ... 112
titles, making ... 112
trademark symbol ... 141
transcript file ... 4
TrueType fonts ... 4
TUGboat template ... 155
two-column output ... 23
two-thirds em-dash ... 141
type styles ... 17
typed text, simulating ... 69
typeface sizes ... 19
typefaces ... 17
typewriter font ... 18
typewriter labels in lists ... 40

U

umlaut accent ... 141
underbar ... 142
underscore, in text ... 141
Unicode input, native ... 4
units, of length ... 90
unofficial nature of this manual ... 2
unordered lists ... 47
Upper case ... 137
using BibTEX ... 68
UTF-8 ... 4

V

variables, a list of ... 87
vector symbol, math ... 107
verbatim text ... 69
verbatim text, inline ... 70
vertical bar, double, in text ... 139
vertical bar, in text ... 139
vertical space ... 117, 118
vertical space before paragraphs ... 92
visible space ... 70
visible space symbol, in text ... 141

W

weights, of fonts ... 20
white space ... 114
wide hat accent, math ... 107
wide tilde accent, math ... 107
widths, of fonts ... 20
writing external files ... 45
writing letters ... 148

X

x-height ... 90
XeTEX ... 5
`xindy` program ... 147
`xspace` package ... 116

Command Index

$

$ 94

&

& 62

—

--help command-line option 153

.

.aux file 4
.dvi file 3
.fd file 84
.lof file 4, 146
.log file 4
.lot file 4, 146
.pdf file 4
.tex, default extension 153
.toc file 4, 146
.xdv file 5

[

[...] for optional arguments 5

^

^ 94

_

_ 94

\

\ character starting commands 5
\! 107
\" (umlaut accent) 141
\# 137
\$ 137
\% 137
\& 137
\' (acute accent) 141
\' (tabbing) 60
\(..... 94
\(SPACE) 115
\) 94
* 107
\+ 60
\, ... 107
\- 60
\- (hyphenation) 72
\. (dot-over accent) 141
\/ 116
\: 107
\; 107
\< 60
\= (macron accent) 141
\= (tabbing) 60
\> 60, 107
\> (tabbing) 60
\@ 115
\@beginparpenalty 52
\@endparpenalty 52
\@fnsymbol 74
\@ifstar 7
\@itempenalty 52
\@startsection 31
\[..... 94
\] 94
\^ 137
\^ (circumflex accent) 141
_ 137
\` (grave accent) 141
\` (tabbing) 60
\\ (for \shortstack objects) 58
\\ (for center) 39
\\ (for eqnarray) 43
\\ (for flushright) 46
\\ (tabbing) 60
\\ for \author 112
\\ for \title 112
\\ for flushleft 46
\\ for letters 148
\\ for tabular 62
\\ for verse 70
\\ force line break 71
* (for eqnarray) 43
\{ 137
\} 137
\| 95
\~ 137
\~ (tilde accent) 141
\a (tabbing) 60
\a' (acute accent in tabbing) 60
\a= (macron accent in tabbing) 60
\a` (grave accent in tabbing) 60
\aa (å) 142
\AA (Å) 142
\acute 106
\addcontentsline 146
\address 149
\addtocontents{*ext*}{*text*} 146
\addtocounter 89
\addtolength 91
\addvspace 117

\ae (æ) ... 142
\AE (Æ) ... 142
\aleph ... 95
\Alph example ... 43
\Alph{*counter*} ... 87
\alph{*counter*} ... 87
\alpha ... 95
\alsoname ... 147
\amalg ... 95
\and for \author ... 112
\angle ... 95
\appendix ... 31
\approx ... 95
\arabic{*counter*} ... 87
\arccos ... 105
\arcsin ... 105
\arctan ... 105
\arg ... 105
\arraycolsep ... 38
\arrayrulewidth ... 64
\arraystretch ... 64
\ast ... 95
\asymp ... 95
\AtBeginDocument ... 41
\AtBeginDvi ... 11
\AtEndDocument ... 42
\AtEndOfClass ... 11
\AtEndOfPackage ... 11
\author{*name* \and *name2*} ... 112
\b (bar-under accent) ... 141
\backslash ... 95
\bar ... 106
\baselineskip ... 21
\baselinestretch ... 21
\begin ... 37
\beta ... 95
\bf ... 18
\bfseries ... 17
\bibitem ... 67
\bibliography ... 68
\bibliographystyle ... 68
\bibname ... 67
\bigcap ... 95
\bigcirc ... 95
\bigcup ... 95
\bigodot ... 95
\bigoplus ... 95
\bigotimes ... 95
\bigskip ... 117
\bigskipamount ... 117
\bigsqcup ... 96
\bigtriangledown ... 95
\bigtriangleup ... 95
\biguplus ... 96
\bigvee ... 96
\bigwedge ... 96
\bmod ... 105
\boldmath ... 94
\bot ... 96
\bottomfraction ... 29
\bowtie ... 96
\Box ... 96
\breve ... 106
\bullet ... 96
\c (cedilla accent) ... 142
\cal ... 18
\cap ... 96
\capitalacute ... 141
\capitalbreve ... 142
\capitalcaron ... 142
\capitalcedilla ... 142
\capitalcircumflex ... 141
\capitaldieresis ... 141
\capitaldotaccent ... 142
\capitalgrave ... 141
\capitalhungarumlaut ... 142
\capitalmacron ... 141
\capitalnewtie ... 142
\capitalogonek ... 142
\capitalring ... 142
\capitaltie ... 142
\capitaltilde ... 141
\caption ... 45, 61
\cc ... 149
\cdot ... 96
\cdots ... 108
\centering ... 39
\chapter ... 31
\check ... 106
\CheckCommand ... 11
\CheckCommand* ... 11
\chi ... 96
\circ ... 96
\circle ... 55
\cite ... 68
\ClassError ... 11
\ClassInfo ... 11
\ClassInfoNoLine ... 11
\ClassWarning ... 11
\ClassWarningNoLine ... 11
\cleardoublepage ... 73
\clearpage ... 73
\cline ... 66
\closing ... 150
\clubsuit ... 96
\columnsep ... 23, 25
\columnseprule ... 23, 25
\columnwidth ... 23, 25
\complement ... 96
\cong ... 96
\contentsline ... 146
\coprod ... 96
\copyright ... 138
\cos ... 105
\cosh ... 105
\cot ... 105

\coth ... 105
\csc ... 105
\cup ... 96
\CurrentOption ... 12
\d (dot-under accent) ... 142
\dag ... 138
\dagger ... 96
\dashbox ... 56
\dashv ... 96
\date{*text*} ... 112
\day ... 89
\dblfloatpagefraction ... 24
\dblfloatsep ... 24
\dbltextfloatsep ... 24
\dbltopfraction ... 23
\dbltopnumber ... 24
\ddag ... 138
\ddagger ... 96
\ddot ... 106
\ddots ... 108
\DeclareGraphicsExtensions ... 128
\DeclareGraphicsRule ... 129
\DeclareOption ... 12
\DeclareOption* ... 12
\DeclareRobustCommand ... 12
\DeclareRobustCommand* ... 12
\deg ... 105
\Delta ... 96
\delta ... 96
\depth ... 91
\det ... 105
\dh (ð) ... 143
\DH (Ð) ... 143
\Diamond ... 96
\diamond ... 96
\diamondsuit ... 97
\dim ... 106
\displaystyle ... 94
\div ... 97
\dj ... 143
\DJ ... 143
\documentclass ... 8
\dot ... 106
\doteq ... 97
\dotfill ... 116
\dots ... 138
\doublerulesep ... 64
\downarrow ... 97
\Downarrow ... 97
\ell ... 97
\emph ... 18
\emptyset ... 97
\encl ... 150
\end ... 37
\enlargethispage ... 73
\enumi ... 43
\enumii ... 43
\enumiii ... 43
\enumiv ... 43
\epsilon ... 97
\equiv ... 97
\eta ... 97
\evensidemargin ... 9, 26
\ExecuteOptions ... 14
\exists ... 97
\exp ... 106
\extracolsep ... 63
\fbox ... 119
\fboxrule ... 56, 119
\fboxsep ... 56, 119
\fill ... 114
\flat ... 97
\floatpagefraction ... 29
\floatsep ... 29
\flushbottom ... 25
\fnsymbol, and footnotes ... 74
\fnsymbol{*counter*} ... 87
\fontencoding ... 19
\fontfamily ... 19
\fontseries ... 20
\fontshape ... 21
\fontsize ... 21
\footnote ... 74
\footnotemark ... 75
\footnoterule ... 77
\footnotesep ... 77
\footnotesize ... 19
\footnotetext ... 75
\footskip ... 25
\forall ... 97
\frac ... 108
\frac{*num*}{*den*} ... 108
\frame ... 56
\framebox ... 56, 119
\frenchspacing ... 115
\frown ... 97
\fussy ... 72
\gamma ... 97
\Gamma ... 97
\gcd ... 106
\ge ... 97
\geq ... 97
\gets ... 97
\gg ... 97
\glossary ... 147
\glossaryentry ... 147
\graphicspath ... 127
\grave ... 106
\guillemotleft («) ... 138
\guillemotright (») ... 138
\guilsinglleft (‹) ... 138
\guilsinglright (›) ... 138
\H (Hungarian umlaut accent) ... 142
\hat ... 107
\hbar ... 97
\headheight ... 25

\headsep ... 25
\heartsuit ... 97
\height ... 91
\hfill ... 114
\hline ... 67
\hom ... 106
\hookleftarrow ... 97
\hookrightarrow ... 97
\hrulefill ... 116
\hsize ... 27
\hspace ... 114
\Huge ... 19
\huge ... 19
\hyphenation ... 72
\i (dotless i) ... 142
\iff ... 97
\IfFileExists ... 13
\ignorespaces ... 85
\ignorespacesafterend ... 85
\ij (ij) ... 143
\IJ (IJ) ... 143
\Im ... 98
\imath ... 107
\in ... 98
\include ... 145
\includegraphics ... 130
\includeonly ... 145
\indent ... 92
\index ... 147
\indexentry ... 147
\indexspace ... 147
\inf ... 106
\infty ... 98
\input ... 145
\InputIfFileExists ... 13
\int ... 98
\intextsep ... 29
\iota ... 98
\it ... 18
\item ... 40, 42, 47
\itemindent ... 49
\itemsep ... 49
\itshape ... 17
\j (dotless j) ... 142
\jmath ... 107
\Join ... 98
\k (ogonek) ... 142
\kappa ... 98
\ker ... 106
\kill ... 60
\l (␣l) ... 143
\L (␣L) ... 143
\label ... 35
\labelenumi ... 43
\labelenumii ... 43
\labelenumiii ... 43
\labelenumiv ... 43
\labelitemi ... 47
\labelitemii ... 47
\labelitemiii ... 47
\labelitemiv ... 47
\labelsep ... 50
\labelwidth ... 50
\Lambda ... 98
\lambda ... 98
\land ... 98
\langle ... 98
\large ... 19
\Large ... 19
\LARGE ... 19
\LaTeX ... 138
\LaTeXe ... 138
\lbrace ... 98
\lbrack ... 98
\lceil ... 98
\ldots ... 138
\le ... 98
\leadsto ... 98
\left *delim1* ... \right *delim2* ... 108
\leftarrow ... 98
\Leftarrow ... 98
\lefteqn ... 44
\leftharpoondown ... 98
\leftharpoonup ... 98
\leftmargin ... 47, 50
\leftmargini ... 47
\leftmarginii ... 47
\leftmarginiii ... 47
\leftmarginiv ... 47
\leftmarginv ... 47
\leftmarginvi ... 47
\leftrightarrow ... 98
\Leftrightarrow ... 98
\leq ... 99
\lfloor ... 99
\lg ... 106
\lhd ... 99
\lim ... 106
\liminf ... 106
\limsup ... 106
\line ... 56
\linebreak ... 72
\linespread ... 21
\linethickness ... 57
\linewidth ... 26
\listoffigures ... 146
\listoftables ... 146
\listparindent ... 50
\ll ... 99
\ln ... 106
\lnot ... 99
\LoadClass ... 13
\LoadClassWithOptions ... 13
\location ... 150
\log ... 106
\longleftarrow ... 99

\longleftrightarrow ... 99
\longmapsto ... 99
\longrightarrow ... 99
\lor ... 99
\lq ... 138
\makebox ... 119
\makebox (for picture) ... 56
\makeglossary ... 147
\makeindex ... 147
\makelabel ... 49
\makelabels ... 150
\maketitle ... 112
\mapsto ... 99
\marginpar ... 92
\marginparpush ... 26, 93
\marginparsep ... 93
\marginparwidth ... 26, 93
\marginsep ... 26
\markboth{*left*}{*right*} ... 113
\markright{*right*} ... 113
\mathbf ... 18
\mathcal ... 18
\mathdollar ... 108
\mathellipsis ... 108
\mathnormal ... 18
\mathparagraph ... 108
\mathring ... 107
\mathrm ... 18
\mathsection ... 108
\mathsf ... 18
\mathsterling ... 108
\mathtt ... 18
\mathunderscore ... 108
\mathversion ... 18
\max ... 106
\mbox ... 119
\mbox, and LR mode ... 110
\mdseries ... 17
\medskip ... 117
\medskipamount ... 117
\medspace ... 107
\mho ... 99
\mid ... 99
\min ... 106
\models ... 99
\month ... 89
\mp ... 99
\mu ... 99
\multicolumn ... 65
\multiput ... 57
\nabla ... 99
\name ... 150
\natural ... 99
\ne ... 99
\nearrow ... 100
\NeedsTeXFormat ... 14
\neg ... 100
\neq ... 100
\newcommand ... 78
\newcounter ... 79
\newenvironment ... 80
\newfont ... 84
\newlength ... 80
\newline ... 71
\NEWLINE ... 115
\newpage ... 73
\newsavebox ... 80
\newtheorem ... 82
\newtie ... 142
\ng ... 143
\NG ... 143
\ni ... 100
\nocite ... 68
\nocorr ... 17
\nocorrlist ... 17
\nofiles ... 146
\noindent ... 92
\nolinebreak ... 72
\nonfrenchspacing ... 115
\nonumber ... 43
\nopagebreak ... 73
\normalfont ... 17
\normalmarginpar ... 92
\normalsfcodes ... 116
\normalsize ... 19
\not ... 100
\notin ... 100
\nu ... 100
\nwarrow ... 100
\o (ø) ... 143
\O (Ø) ... 143
\obeycr ... 71
\oddsidemargin ... 9, 26
\odot ... 100
\oe (œ) ... 143
\OE (Œ) ... 143
\oint ... 100
\oldstylenums ... 18
\Omega ... 100
\omega ... 100
\ominus ... 100
\onecolumn ... 23
\opening ... 151
\oplus ... 100
\OptionNotUsed ... 14
\oslash ... 100
\otimes ... 100
\oval ... 57
\overbrace{*math*} ... 108
\overline{*text*} ... 108
\owns ... 100
\PackageError ... 11
\PackageInfo ... 11
\PackageInfoNoLine ... 11
\PackageWarning ... 11
\PackageWarningNoLine ... 11

\pagebreak ... 73
\pagenumbering ... 112
\pageref ... 36
\pagestyle ... 112
\paperheight ... 26
\paperwidth ... 26
\paragraph ... 31
\parallel ... 100
\parbox ... 120
\parindent ... 54, 92
\parsep ... 50
\parskip ... 92
\parskip example ... 48
\part ... 31
\partial ... 100
\partopsep ... 50
\PassOptionsToClass ... 14
\PassOptionsToPackage ... 14
\pdfpageheight ... 9
\pdfpagewidth ... 9
\perp ... 100
\phi ... 100
\P ... 138
\Pi ... 100
\pi ... 100
\pm ... 100
\pmod ... 106
\poptabs ... 60
\pounds ... 139
\Pr ... 106
\prec ... 100
\preceq ... 100
\prime ... 101
\printindex ... 147
\ProcessOptions ... 15
\ProcessOptions* ... 15
\prod ... 101
\propto ... 101
\protect ... 84
\providecommand ... 79
\ProvidesClass ... 15
\ProvidesFile ... 16
\ProvidesPackage ... 15
\ps ... 151
\psi ... 101
\Psi ... 101
\pushtabs ... 60
\put ... 57
\qquad ... 107
\quad ... 107
\quotedblbase („) ... 139
\quotesinglbase (‚) ... 139
\r (ring accent) ... 142
\raggedbottom ... 25
\raggedleft ... 46
\raggedright ... 46
\raisebox ... 120
\rangle ... 101
\rbrace ... 101
\rbrack ... 101
\rceil ... 101
\Re ... 101
\ref ... 36
\reflectbox ... 136
\refname ... 67
\refstepcounter ... 89
\renewenvironment ... 80
\RequirePackage ... 16
\RequirePackageWithOptions ... 16
\resizebox ... 136
\restorecr ... 71
\restriction ... 101
\revemptyset ... 101
\reversemarginpar ... 92
\rfloor ... 101
\rhd ... 101
\rho ... 101
\right ... 108
\Rightarrow ... 101
\rightarrow ... 101
\rightharpoondown ... 101
\rightharpoonup ... 101
\rightleftharpoons ... 102
\rightmargin ... 51
\rm ... 18
\rmfamily ... 17
\roman{*counter*} ... 87
\Roman{*counter*} ... 87
\rotatebox ... 135
\rq ... 139
\rule ... 143
\savebox ... 121
\sbox ... 121
\sc ... 18
\scalebox ... 136
\scriptsize ... 19
\scshape ... 17
\searrow ... 102
\sec ... 106
\section ... 31
\seename ... 147
\selectfont ... 21
\setcounter ... 89
\setlength ... 91
\setminus ... 102
\settodepth ... 91
\settoheight ... 91
\settowidth ... 91
\sf ... 18
\sffamily ... 17
\sharp ... 102
\shortstack ... 58
\Sigma ... 102
\sigma ... 102
\signature ... 151
\sim ... 102

\simeq ... 102
\sin ... 106
\sinh ... 106
\sl ... 18
\sloppy ... 72
\slshape ... 17
\small ... 19
\smallint ... 102
\smallskip ... 117
\smallskipamount ... 117
\smile ... 102
\spacefactor ... 114
\spadesuit ... 102
\sqcap ... 102
\sqcup ... 102
\sqrt[*root*]{*arg*} ... 108
\sqsubset ... 102
\sqsubseteq ... 102
\sqsupset ... 102
\sqsupseteq ... 102
\ss (ß) ... 143
\SS (SS) ... 143
\stackrel{*text*}{*relation*} ... 109
\star ... 102
\stepcounter ... 89
\stop ... 153
\subparagraph ... 31
\subsection ... 31
\subset ... 102
\subseteq ... 102
\subsubsection ... 31
\succ ... 103
\succeq ... 103
\sum ... 103
\sup ... 106
\suppressfloats ... 28
\supset ... 103
\supseteq ... 103
\surd ... 103
\swarrow ... 103
\symbol ... 138
\S ... 139
\t (tie-after accent) ... 142
\tabbingsep ... 60
\tabcolsep ... 64
\tableofcontents ... 146
\TAB ... 115
\tan ... 106
\tanh ... 106
\tau ... 103
\telephone ... 151
\textascendercompwordmark ... 139
\textasciicircum ... 139
\textasciitilde ... 139
\textasteriskcentered ... 139
\textbackslash ... 137, 139
\textbar ... 139
\textbardbl ... 139
\textbf ... 17
\textbigcircle ... 139
\textbraceleft ... 139
\textbraceright ... 139
\textbullet ... 139
\textcapitalcompwordmark ... 139
\textcircled{*letter*} ... 139
\textcompwordmark ... 139
\textcopyright ... 138
\textdagger ... 139
\textdaggerdbl ... 139
\textdollar (or \$) ... 140
\textellipsis ... 138
\textemdash (or ---) ... 140
\textendash (or --) ... 140
\texteuro ... 140
\textexclamdown (or !‘) ... 140
\textfloatsep ... 29
\textfraction ... 29
\textgreater ... 140
\textheight ... 26
\textit ... 17
\textleftarrow ... 140
\textless ... 140
\textmd ... 17
\textnormal ... 17
\textordfeminine ... 140
\textordmasculine ... 140
\textparagraph ... 138
\textperiodcentered ... 140
\textquestiondown (or ?‘) ... 140
\textquotedblleft (or ‘‘) ... 140
\textquotedblright (or ’’) ... 140
\textquoteleft (or ‘) ... 140
\textquoteright (or ’) ... 140
\textquotesingle ... 140
\textquotestraightbase ... 140
\textquotestraightdblbase ... 140
\textregistered ... 140
\textrightarrow ... 140
\textrm ... 17
\textsc ... 17
\textsf ... 17
\textsl ... 17
\textsterling ... 139
\textthreequartersemdash ... 141
\texttrademark ... 141
\texttt ... 17
\texttwelveudash ... 141
\textunderscore ... 141
\textup ... 17
\textvisiblespace ... 141
\textwidth ... 27
\TeX ... 139
\th (þ) ... 143
\TH (Þ) ... 143
\thanks{*text*} ... 112
\theta ... 103

\thicklines ... 57
\thickspace ... 107
\thinlines ... 57
\thinspace ... 107, 116
\thispagestyle ... 113
\tilde ... 107
\times ... 103
\tiny ... 19
\title{*text*} ... 112
\to ... 103
\today ... 143
\top ... 103
\topfraction ... 29
\topmargin ... 27
\topsep ... 51
\topskip ... 27
\totalheight ... 91
\triangle ... 103
\triangleleft ... 103
\triangleright ... 103
\tt ... 18
\ttfamily ... 17
\twocolumn ... 23
\typein ... 152
\typeout ... 152
\u (breve accent) ... 142
\unboldmath ... 94
\underbar ... 142
\underbrace{*math*} ... 109
\underline{*text*} ... 109
\unitlength ... 54
\unlhd ... 103
\unrhd ... 103
\uparrow ... 103
\Uparrow ... 103
\updownarrow ... 104
\Updownarrow ... 103
\upharpoonright ... 104
\uplus ... 104
\upshape ... 17
\upsilon ... 104
\Upsilon ... 104
\usebox ... 121
\usecounter ... 88
\usefont ... 22
\usepackage ... 9
\v (breve accent) ... 142
\value ... 88
\vanothing ... 104
\varepsilon ... 104
\varphi ... 104
\varpi ... 104
\varrho ... 104
\varsigma ... 104
\vartheta ... 104
\vdash ... 104
\vdots ... 109
\vec ... 107
\vector ... 58
\vee ... 104
\verb ... 70
\vert ... 105
\Vert ... 104
\vfill ... 117
\vline ... 66
\vspace ... 118
\wedge ... 105
\widehat ... 107
\widetilde ... 107
\width ... 91
\wp ... 105
\wr ... 105
\Xi ... 105
\xi ... 105
\year ... 89
\zeta ... 105

{
{...} for required arguments ... 5

1

10pt option ... 8
11pt option ... 8
12pt option ... 8

A

a4paper option ... 8
a5paper option ... 8
abstract environment ... 37
array environment ... 38
article class ... 8

B

b5paper option ... 8
book class ... 8
bottomnumber ... 29
bp ... 90

C

cc ... 90
center environment ... 39
clock option to slides class ... 9
cm ... 90

D

dbltopnumber ... 29
dd ... 90
description environment ... 40
displaymath environment ... 41, 94
document environment ... 41
draft option ... 9
dvipdfmx command ... 3
dvips command ... 3
dvitype command ... 3

E

em ... 90
enumerate environment ... 42
environment, abstract ... 37
environment, array ... 38
environment, center ... 39
environment, description ... 40
environment, displaymath ... 41, 94
environment, document ... 41
environment, enumerate ... 42
environment, eqnarray ... 43
environment, equation ... 44, 94
environment, figure ... 44
environment, filecontents ... 45
environment, filecontents* ... 45
environment, flushleft ... 46
environment, flushright ... 46
environment, itemize ... 47
environment, letter ... 48
environment, list ... 48
environment, math ... 54, 94
environment, minipage ... 54
environment, picture ... 54
environment, quotation ... 58
environment, quote ... 58
environment, tabbing ... 59
environment, table ... 61
environment, tabular ... 62
environment, thebibliography ... 67
environment, theorem ... 69
environment, titlepage ... 69
environment, verbatim ... 69
environment, verse ... 70
eqnarray environment ... 43
equation environment ... 44, 94
etex command ... 4
ex ... 90
executivepaper option ... 8

F

figure environment ... 44
filecontents environment ... 45
filecontents* environment ... 45
final option ... 9
first-latex-doc document ... 2
fleqn option ... 9
flushleft environment ... 46
flushright environment ... 46

H

http://puszcza.gnu.org.ua/software/latexrefman/
home page ... 2

I

in ... 90
inch ... 90
itemize environment ... 47

L

landscape option ... 9
latex command ... 3
latex-doc-ptr document ... 2
latexrefman@tug.org email address ... 2
legalpaper option ... 8
leqno option ... 9
letter class ... 8
letter environment ... 48
letterpaper option ... 8
list environment ... 48
lR box ... 55
lrbox ... 119
lshort document ... 2
lualatex command ... 4

M

math environment ... 54, 94
minipage environment ... 54
mm ... 90
mu ... 90

N

notitlepage option ... 9

O

onecolumn option ... 9
oneside option ... 9
openany option ... 9
openbib option ... 9
openright option ... 9

P

pc ... 90
pdflatex command ... 4
picture environment ... 54
pt ... 90

Q

quotation environment ... 58
quote environment ... 58

R

report class ... 8

S

secnumdepth counter ... 31
slides class ... 8
sp ... 90

T

tabbing environment ... 59
table environment ... 61
tabular environment ... 62
textcomp package ... 138
thebibliography environment ... 67
theorem environment ... 69
titlepage environment ... 69
titlepage option ... 9
topmargin ... 27
topnumber ... 29
totalnumber ... 29
twocolumn option ... 9
twoside option ... 9

U

usrguide official documentation ... 2

V

verbatim environment ... 69
verse environment ... 70

X

xdvi command ... 3
xdvipdfmx ... 5
xelatex command ... 5