The LaTeX Tutorial

INDIAN TeX USERS GROUP

THE LaTeX Tutorial
A PRIMER

THE TUTORIAL TEAM
INDIAN TEX USERS GROUP

SAYAHNA FOUNDATION
Trivandrum, India
http://www.sayahna.org
2020

THE LaTeX TUTORIAL — A PRIMER
Indian TeX Users Group
EDITOR: E. Krishnan

The PDF was generated from sources marked up in LaTeX in a computer running GNU/LINUX operating system. It was typeset using pdfTeX from TeXLive 2019. The base font used was Sabon at ten point, sans-serif font was Frutiger and math font was from newpx-math by Michael Sharpe.

Cover: Revolution of the Viaduct, a painting by Paul Klee (1879–1940). The image has been taken from Wikimedia and is gratefully acknowledged.

Published by **Sayahna Foundation**
JWRA 34, Jagathy, Trivandrum, India 695014
URL: www.sayahna.org

PREFACE TO THE FIRST EDITION

The ideal situation occurs when the things that we regard as beautiful are also regarded by other people as useful.

— Donald Knuth

For us who wrote the following pages, TEX is something beautiful and also useful. We enjoy TEX, sharing the delights of newly discovered secrets amongst ourselves and wondering ever a new at the infinite variety of the program and the ingenuity of its creator. We also lend a helping hand to the new initiates to this art. Then we thought of extending this help to a wider group and The Net being the new medium, we started an online tutorial. This was well received and now the Free Software Foundation has decided to publish these lessons as a book. It is a fitting gesture that the organization which upholds the rights of the user to study and modify a software publish a book on one of the earliest programs which allows this right.

Dear reader, read the book, enjoy it and if possible, try to add to it.

Trivandrum
1 Nov 2002

The TUG*India* Tutorial Team

Contents

1. The Basics 1

1.1 What is LaTeX? – 1 • 1.2 Simple typesetting – 3 • 1.3 Fonts – 10 • 1.4 Type size – 12

2. The Document 15

2.1 Document class – 15 • 2.2 Page style – 17 • 2.3 Page numbering – 19 • 2.4 Formatting lengths – 20 • 2.5 Parts of a document – 20 • 2.6 Dividing the document – 22 • 2.7 What next? – 25

3. Bibliography 27

3.1 Introduction – 27 • 3.2 natbib – 29

4. Bibliographic Databases 35

4.1 The bibTeX program – 35 • 4.2 BibTeX style files – 36 • 4.3 Creating a bibliographic database – 37

5. Table of contents, Index and Glossary 43

5.1 Table of contents – 43 • 5.2 Index – 47 • 5.3 Glossary – 52

6. Displayed Text 55

6.1 Borrowed words – 55 • 6.2 Poetry in typesetting – 57 • 6.3 Making lists – 57 • 6.4 When order matters – 61 • 6.5 Descriptions and definitions – 64

7. Rows and Columns 69

7.1 Keeping tabs – 69 • 7.2 Tables – 76

8. Typesetting Mathematics 95

8.1 The basics – 95 • 8.2 Custom commands – 101 • 8.3 More on mathematics – 102 • 8.4 Mathematics miscellany – 112 • 8.5 New operators – 126 • 8.6 The many faces of mathematics – 127 • 8.7 And that is not all! – 129 • 8.8 Symbols – 130

9. Typesetting Theorems 137
9.1 Theorems in LaTeX – 137 • 9.2 Designer theorems—The amsthm package – 140 • 9.3 Housekeeping – 150

10. Several Kinds of Boxes 151
10.1 Boxes—with frames – 151 • 10.2 Frames of any length – 154 • 10.3 Special lengths – 155 • 10.4 Boxes without frames – 157 • 10.5 Multiline boxes – 159 • 10.6 Black boxes and invisible lines – 162 • 10.7 Moving boxes around – 164

11. Floats 167
11.1 The `figure` environment – 167 • 11.2 The `table` environment – 171

12. Graphics Inclusion 179
12.1 The graphicx package – 179

13. Cross References in LaTeX 187
13.1 Why cross references? – 187 • 13.2 Let LaTeX do it – 187 • 13.3 Pointing to a page—the package varioref – 191 • 13.4 Pointing outside—the package xr – 194 • 13.5 Lost the keys? Use `lablst.tex` – 195

14. Footnotes, Marginpars, and Endnotes 197
14.1 Footnotes – 197 • 14.2 Marginal notes – 203 • 14.3 Endnotes – 205

CHAPTER I

THE BASICS

1.1. WHAT IS LaTeX?

THE SHORT and simple answer is that LaTeX is a typesetting program and is an extension of the original program TeX written by Donald Knuth. But then what is a *typesetting program*? To answer this, let us look at the various stages in the preparation of a document using computers.

1. The text is *entered* into the computer.
2. The input text is *formatted* into lines, paragraphs and pages.
3. The output text is *displayed* on the computer screen.
4. The final output is *printed*.

In most *word processors* all these operations are integrated into a single application package. But a typesetting program like TeX is concerned only with the second stage above. So to typeset a document using TeX, we type the text of the document and the necessary formatting commands in a *text editor* (such as `Emacs` in GNU/Linux) and then compile it. After that the document can be viewed using a *previewer* or printed using a *printer driver*.

TeX is also a *programming language*, so that by learning this language, people can write code for additional features. In fact LaTeX itself is such a (large) collection of extra features. And the collective effort is continuing, with more and more people writing extra *packages*.

1.1.1. A SMALL EXAMPLE

Let us see LaTeX in action by typesetting a short (really short) document. Start your favorite text editor and type in the lines below *exactly* as shown

```
\documentclass{article}
\begin{document}
This is my \emph{first} document prepared in \LaTeX.
\end{document}
```

Be especially careful with the `\` character (called the *backslash*) and note that this is different from the more familiar `/` (the *slash*) in `and/or` and save the file onto the hard disk as `myfile.tex`. (Instead of `myfile` you can use any name you wish, but be sure to have `.tex` at the end as the *extension*.) The process of compiling this and viewing the output depends on your operating system. We describe below the process of doing this in GNU/Linux.

At the shell prompt type

```
latex myfile
```

You will see a number of lines of text scroll by in the screen and then you get the prompt back. To view the output in screen, you must have the X Window running. So, start `X` if you have not done so, and in a terminal window, type

```
xdvi myfile
```

A window comes up showing the output below

> This is my *first* document prepared in LaTeX.

Now let us take a closer look at the *source file* (that is, the file you have typed). The first line `\documentclass{article}` tells LaTeX that what we want to produce is an article. If you want to write a book, this must be changed to `\documentclass{book}`. The whole document we want to typeset should be included between `\begin{document}` and `\end{document}`. In our example, this is just one line. Now compare this line in the source and the output. The first three words are produced as typed. Then `\emph{first}`, becomes *first* in the output (as you have probably noticed, it is a common practice to *emphasize* words in print using italic letters). Thus `\emph` is a *command* to LaTeX to typeset the text within the braces in *italic*.[1] Again, the next three

[1] This is not really true. For the real story of the command, see the section on fonts.

words come out without any change in the output. Finally, the input `\LaTeX` comes out in the output as LaTeX.

Thus our source is a mixture of text to be typeset and a couple of LaTeX *commands* `\emph` and `\LaTeX`. The first command changes the input text in a certain way and the second one generates new text. Now call up the file again and add one more sentence given below.

```
This is my \emph{first} document prepared in \LaTeX. I
typed it on \today.
```

What do you get in the output? What new text does the command `\today` generate?

1.1.2. WHY LaTeX?

So, why all this trouble? Why not simply use a word processor? The answer lies in the motivation behind TeX. Donald Knuth says that his aim in creating TeX is to *beautifully* typeset *technical documents* especially those containing *a lot of Mathematics*. It is very difficult (sometimes even impossible) to produce complex mathematical formulas using a word processor. Again, even for ordinary text, if you want your document to look *really beautiful* then LaTeX is the natural choice.

1.2. SIMPLE TYPESETTING

We have seen that to typeset something in LaTeX, we type in the text to be typeset together with some LaTeX commands. Words must be separated by spaces (does not matter how many) and lines maybe broken arbitrarily.

The end of a paragraph is specified by a *blank line* in the input. In other words, whenever you want to start a new paragraph, just leave a blank line and proceed. For example, the first two paragraphs above were produced by the input

```
We have seen that to typeset something in \LaTeX, we
type in the text to be typeset together with some
\LaTeX\ commands. Words must be separated by spaces
(does not matter how many) and lines maybe broken
arbitrarily.

The end of a paragraph is specified by a \emph{blank
```

```
line} in the input. In other words, whenever you want
to start a new paragraph, just leave a blank line and
proceed.
```

Note that the first line of each paragraph starts with an *indentation* from the left margin of the text. If you do not want this indentation, just type `\noindent` at the start of each paragraph for example, in the above input, `\noindent We have seen...` and `\noindent The end of ...` (come on, try it!) There is an easier way to suppress paragraph indentation for *all* paragraphs of the document in one go, but such tricks can wait.

1.2.1. SPACES

You might have noticed that even though the length of the lines of text we type in a paragraph are different, in the output, all lines are of equal length, aligned perfectly on the right and left. TEX does this by adjusting the space between the words.

In traditional typesetting, a little extra space is added to periods which end sentences and TEX also follows this custom. But how does TEX know whether a period ends a sentence or not? It assumes that every period *not following an upper case letter* ends a sentence. But this does not always work, for there are instances where a sentence does end in an upper case letter. For example, consider the following

> Carrots are good for your eyes, since they contain Vitamin A. Have you ever seen a rabbit wearing glasses?

The right input to produce this is

```
Carrots are good for your eyes, since they contain Vitamin
A\@. Have you ever seen a rabbit wearing glasses?
```

Note the use of the command `\@` *before* the period to produce the extra space after the period. (Remove this from the input and see the difference in the output.)

On the other hand, there are instances where a period following a lowercase letter does not end a sentence. For example

> The numbers 1, 2, 3, etc. are called natural numbers. According to Kronecker, they were made by God; all else being the work of Man.

To produce this (without extra space after `etc.`) the input should be

```
The numbers 1, 2, 3, etc.\ are called natural numbers.
According to Kronecker, they were made by God; all else
being the works of Man.
```

Here, we use the command `\␣` (that is, a backslash and a *space*—here and elsewhere, we sometimes use ␣ to denote a space in the input, especially when we draw attention to the space).

There are other situations where the command `\␣` (which always produce a space in the output) is useful. For example, type the following line and compile it.

```
I think \LaTeX is fun.
```

You get

> I think LaTeXis fun.

What happened to the *space* you typed between `\LaTeX` and `is`? You see, TeX gobbles up all spaces after a command. To get the required sequence in the output, change the input as

```
I think \LaTeX\ is fun.
```

Again, the command `\␣` comes to the rescue.

1.2.2. QUOTES

Have you noticed that in typesetting, opening quotes are different from closing quotes? Look at the TeX output below

> Note the difference in right and left quotes in ‘single quotes’ and “double quotes”.

This is produced by the input

```
Note the difference in right and left quotes in `single
quotes' and ``double quotes''.
```

Modern computer keyboards have a key to type the symbol ` which produces a left quote in TeX. (In our simulated inputs, we show this symbol as `` ` ``.) Also, the key ' (the usual ‘typewriter’ quote key, which also doubles as the apostrophe key) produces a right quote in TeX. Double quotes are produced by typing the corresponding single quote twice. The ‘usual’ double quote key " can also be used to produce a *closing* double quote in TeX.

If your keyboard does not have a left quote key, you can use `\lq` command to produce it. The corresponding command

\rq produces a right quote. Thus the output above can also be produced by

```
Note the difference in right and left quotes in \lq
single quotes\rq\ and \lq\lq double quotes\rq\rq.
```

(Why the command \␣ after the first \rq?)

1.2.3. DASHES

In text, dashes are used for various purposes and they are distinguished in typesetting by their lengths; thus short dashes are used for hyphens, slightly longer dashes are used to indicate number ranges and still longer dashes used for parenthetical comments. Look at the following TEX output

> X-rays are discussed in pages 221–225 of Volume 3—the volume on electromagnetic waves.

This is produced from the input

```
X-rays are discussed in pages 221--225 of Volume 3---the
volume on electromagnetic waves.
```

Note that a single dash character in the input - produces a hyphen in the output, two dashes -- produces a longer dash (–) in the output and three dashes --- produce the longest dash (—) in the output.

1.2.4. ACCENTS

Sometimes, especially when typing foreign words in English, we need to put different types of accents over the letters. The table below shows the accents available in LATEX. Each column shows some of the accents and the inputs to generate them.

ò	\‘o	ó	\’o	ô	\^o	õ	\~o
ō	\=o	ȯ	\.o	ö	\"o	ç	\c c
ǒ	\u o	ő	\v o	ő	\H o	ọ	\d o
o͟	\b o	o͡o	\t oo				

The letters i and j need special treatment with regard to accents, since they should not have their customary dots when accented. The commands \i and \j produce dot-less i and j as ı and ȷ. Thus to get

Él está aquí

you must type

```
\'{E}l est\'{a} aqu\'{\i}
```

Some symbols from non-English languages are also available in LaTeX, as shown in the table below:

œ	\oe	Œ	\OE	æ	\ae	Æ	\AE
å	\aa	Å	\AA				
ø	\o	Ø	\O	ł	\l	Ł	\L
ß	\ss						
¡	!`	¿	?`				

1.2.5. SPECIAL SYMBOLS

We have seen that the input `\LaTeX` produces LaTeX in the output and `\␣` produces a space. Thus TeX uses the symbol `\` for a special purpose—to indicate the program that what follows is not text to be typeset but an instruction to be carried out. So what if you want to get \ in your output (improbable as it may be)? The command `\textbackslash` produces \ in the output.

Thus `\` is a symbol which has a special meaning for TeX and cannot be produced by direct input. As another example of such a special symbol, see what is obtained from the input below

```
Maybe I have now learnt about 1% of \LaTeX.
```

You only get

Maybe I have now learnt about 1

What happened to the rest of the line? You see, TeX uses the per cent symbol (`%`) as the *comment* character; that is a symbol which tells TeX to consider the text following as 'comments' and not as text to be typeset. This is especially useful for a TeX programmer to explain a particularly sticky bit of code to others (and perhaps to himself). Even for ordinary users, this comes in handy, to keep a 'to do' list within the document itself for example.

But then, how do you get a percent sign in the output? Just type `\%` as in

```
Maybe I have now learnt about 1\% of \LaTeX.
```

The symbols `\` and `%` are just two of the ten charcaters TEX reserves for its internal use. The complete list is

```
~ # $ % ^ & _ \ { }
```

We have seen how TEX uses two of these symbols (or is it four? Did not we use `{ }` in one of our examples?) The use of others we will see as we proceed.

Also, we have noted that `\` is produced in the output by the command `\textbackslash` and `%` is produced by `\%`. What about the other symbols? The table below gives the inputs to produce these symbols.

~	\textasciitilde	&	\&
#	\#	_	_
$	\$	\	\textbackslash
%	\%	{	\{
^	\textasciicircum	}	\}

You can see that except for three, all special symbols are produced by preceding them with a `\`. Of the exceptional three, we have seen that `\~` and `\^` are used for producing accents. So what does `\\` do? It is used to break lines. For example,

```
This is the first line.\\ This is the second line
```

produces

This is the first line.
This is the second line

We can also give an *optional* argument to `\\` to increase the vertical distance between the lines. For example,

```
This is the first line.\\[10pt]
This is the second line
```

gives

This is the first line.

This is the second line

Now there is an extra 10 points of space between the lines (1 point is about $1/72^{nd}$ of an inch).

1.2.6. TEXT POSITIONING

We have seen that TEX aligns text in its own way, regardless of the way text is formatted in the input file. Now suppose you want to typeset something like this

The TEXnical Institute

Certificate

This is to certify that Mr. N. O. Vice has undergone a course at this institute and is qualified to be a TEXnician.

The Director
The TEXnical Institute

This is produced by

```
\begin{center}
  The \TeX nical Institute\\[.75cm]
    Certificate
\end{center}
\noindent This is to certify that Mr. N. O. Vice has
 undergone a course at this institute and is qualified to
 be a \TeX nician.
\begin{flushright}
  The Director\\
  The \TeX nical Institute
\end{flushright}
```

Here, the commands

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

typesets the text between them exactly at the center of the page and the commands

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

typesets text flush with the right margin. The corresponding commands

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

places the enclosed text flush with the *left* margin. (Change the `flushright` to `flushleft` and see what happens to the output.)

These examples are an illustration of a LATEX construct called an *environment*, which is of the form

Table 1.1:

	STYLE	COMMAND
FAMILY	roman	`\textrm{roman}`
	sans serif	`\textsf{sans serif}`
	typewriter	`\texttt{typewriter}`
SERIES	medium	`\textmd{medium}`
	boldface	`\textbf{boldface}`
SHAPE	upright	`\textup{upright}`
	italic	`\textit{italic}`
	slanted	`\textsl{slanted}`
	SMALL CAP	`\textsc{small cap}`

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

where *name* is the name of the environment. We have seen an example of an environment at the very beginning of this chapter (though not identified as such), namely the `document` environment.

1.3. FONTS

The actual letters and symbols (collectively called *type*) that LaTeX (or any other typesetting system) produces are characterized by their *style* and *size*. For example, in this book emphasized text is given in *italic* style and the example inputs are given in `typewriter` style. We can also produce smaller and bigger type. A set of types of a particular style and size is called a *font*.

1.3.1. TYPE STYLE

In LaTeX, a type style is specified by family, series and shape. They are shown in the table 1.1.

Any type style in the output is a combination of these three characteristics. For example, by default we get roman family, medium series, upright shape type style in a LaTeX output. The `\textit` command produces roman family, medium series, italic

shape type. Again, the command \textbf produces roman family, boldface series, upright shape type.

We can combine these commands to produce a wide variety of type styles. For example, the input

```
\textsf{\textbf{sans serif family,
     boldface series, upright shape}}
\textrm{\textsl{roman family,
    medium series, slanted shape}}
```

produces the output shown below:

sans serif family, boldface series, upright shape
roman family, medium series, slanted shape

Some of these type styles may not be available in your computer. In that case, LaTeX gives a warning message on compilation and substitutes another available type style which it thinks is a close approximation to what you had requested.

We can now tell the whole story of the \emph command. We have seen that it usually, that is when we are in the middle of normal (upright) text, it produces italic shape. But if the current type shape is slanted or italic, then it switches to upright shape. Also, it uses the family and series of the current font. Thus

```
\textit{A polygon of three sides is called a \emph{triangle}
 and a polygon of four sides is called a \emph{quadrilateral}}
```

gives

A polygon of three sides is called a triangle *and a polygon of four sides is called a* quadrilateral

while the input

```
\textbf{A polygon of three sides is called a
\emph{triangle} and a polygon of four sides is called a
\emph{quadrilateral}}
```

produces

A polygon of three sides is called a *triangle* and a polygon of four sides is called a *quadrilateral*

Each of these type style changing commands has an alternate form as a *declaration*.

For example, instead of `\textbf{boldface}` you can also type `{\bfseries boldface}` to get **boldface**. Note that that not only the name of the command, but its usage also is different. For example, to typeset

> By a **triangle,** we mean a polygon of three sides.

if you type

```
By a \bfseries{triangle}, we mean a polygon of three sides.
```

you will end up with

> By a **triangle, we mean a polygon of three sides.**

Thus to make the declaration act upon a specific piece of text (and no more), the declaration and the text should be enclosed in braces.

The table below completes the one given earlier, by giving also the declarations to produce type style changes.

	STYLE	COMMAND	DECLARATION
SHAPE	upright	`\textup{upright}`	`{\upshape upright}`
	italic	`\textit{italic}`	`{\itshape italic}`
	slanted	`\textsl{slanted}`	`{\slshape slanted}`
	SMALL CAP	`\textsc{small cap}`	`{\scshape small cap}`
SERIES	medium	`\textmd{medium}`	`{\mdseries medium}`
	boldface	`\textbf{boldface}`	`{\bfseries boldface}`
FAMILY	roman	`\textrm{roman}`	`{\rmfamily roman}`
	sans serif	`\textsf{sans serif}`	`{\sffamily sans serif}`
	typewriter	`\texttt{typewriter}`	`{\ttfamily typewriter}`

These declaration names can also be used as environment names. Thus to typeset a long passage in, say, sans serif, just enclose the passage within the commands:

`\begin{sffmily} ... \end{sffamily}`.

1.4. TYPE SIZE

Traditionally, type size is measured in (printer) points. The default type that TEX produces is of 10 pt size. There are some *declarations* (ten, to be precise) provided in LATEX for changing the type size. They are given in the following table:

size	{\tiny size}	size	{\large size}
size	{\scriptsize size}	size	{\Large size}
size	{\footnotesize size}	size	{\LARGE size}
size	{\small size}	size	{\huge size}
size	{\normalsize size}	size	{\Huge size}

Note that the \normalsize corresponds to the size we get by default and the sizes form an ordered sequence with \tiny producing the smallest and \Huge producing the largest. Unlike the style changing commands, there are no *command-with-one-argument* forms for these declarations.

We can combine style changes with size changes. For example, the "certificate" we typed earlier can now be 'improved' as follows

```
\begin{center}
  {\bfseries\huge The \TeX nical Institute}\\[1cm]
    {\scshape\LARGE Certificate}
\end{center}

\noindent This is to certify that Mr. N. O. Vice has
undergone a course at this institute and is qualified
to be a \TeX nical Expert.

\begin{flushright}
  {\sffamily The Director\\
  The \TeX nical Institute}
\end{flushright}
```

and this produces

The TEXnical Institute

CERTIFICATE

This is to certify that Mr. N. O. Vice has undergone a course at this institute and is qualified to be a TEXnical Expert.

The Director
The TEXnical Institute

CHAPTER II

THE DOCUMENT

2.1. DOCUMENT CLASS

We now describe how an entire document with chapters and sections and other embellishments can be produced with LaTeX. We have seen that all LaTeX files should begin by specifying the kind of document to be produced, using the command `\documentclass`. We've also noted that for a short article (which can actually turn out to be quite long!) we write `\documentclass{article}` and for books, we write `\documentclass{book}`. There are other *document classes* available in LaTeX such as `report` and `letter`. All of them share some common features and there are features specific to each.

In addition to specifying the type of document (which we *must* do, since LaTeX has no default document class), we *can* also specify some options which modify the default format.Thus the actual syntax of the `\documentclass` command is

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

Note that options are given in *square brackets* and not braces. (This is often the case with LaTeX commands—options are specified within square brackets, after which mandatory arguments are given within braces.)

2.1.1. FONT SIZE

We can select the size of the font for the normal text in the entire document with one of the options

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

Thus we can say

```
\documentclass[11pt]{article}
```

to set the normal text in our document in 11 pt size. The default is `10pt` and so this is the size we get, if we do not specify any font-size option.

2.1.2. PAPER SIZE

We know that LaTeX has its own method of breaking lines to make paragraphs. It also has methods to make vertical breaks to produce different pages of output. For these breaks to work properly, it must know the width and height of the paper used. The various options for selecting the paper size are given below:

`letterpaper`	11×8.5 in	`a4paper`	20.7×21 in
`legalpaper`	14×8.5 in	`a5paper`	21×14.8 in
`executivepaper`	10.5×7.25 in	`b5paper`	25×17.6 in

Normally, the longer dimension is the vertical one—that is, the height of the page. The default is `letterpaper`.

2.1.3. PAGE FORMATS

There are options for setting the contents of each page in a single column (as is usual) or in two columns (as in most dictionaries). This is set by the options

`onecolumn` `twocolumn`

and the default is `onecolumn`.

There is also an option to specify whether the document will be finally printed on just one side of each paper or on both sides. The names of the options are

`oneside` `twoside`

One of the differences is that with the `twoside` option, page numbers are printed on the right on odd-numbered pages and on the left on even numbered pages, so that when these printed back to back, the numbers are always on the outside, for better visibility. (Note that LaTeX has no control over the actual *printing*. It only makes the *formats* for different types of printing.) The default is `oneside` for `article`, `report` and `letter` and `twoside` for `book`.

In the `report` and `book` class there is a provision to specify the different chapters (we will soon see how). Chapters always

begin on a new page, leaving blank space in the previous page, if necessary. With the `book` class there is the additional restriction that chapters begin only on odd-numbered pages, leaving an entire page blank, if need be. Such behavior is controlled by the options,

```
openany     openright
```

The default is `openany` for `report` class (so that chapters begin on "any" *new* page) and `openright` for the `book` class (so that chapters begin only on *new* `right`, that is, odd numbered, page).

There is also a provision in LaTeX for formatting the "title" (the name of the document, author(s) and so on) of a document with special typographic consideration. In the `article` class, this part of the document is printed along with the text following on the first page, while for `report` and `book`, a *separate* title page is printed. These are set by the options

```
notitlepage     titlepage
```

As noted above, the default is `notitlepage` for `article` and `titlepage` for `report` and `book`. As with the other options, the default behavior can be overruled by explicitly specifying an option with the `documentclass` command.

There are some other options to the `documentclass` which we will discuss in the relevant context.

2.2. PAGE STYLE

Having decided on the overall appearance of the document through the `\documentclass` command with its various options, we next see how we can set the style for the individual pages. In LaTeX parlance, each page has a "head" and "foot" usually containing such information as the current page number or the current chapter or section. Just what goes where is set by the command

```
\pagestyle{...}
```

where the mandatory argument can be any one of the following *styles*

```
plain     empty     headings     myheadings
```

The behavior pertaining to each of these is given below:

plain The page head is empty and the foot contains just the page number, centered with respect to the width of the text. This is the default for the `article` class if no `\pagestyle` is specified in the preamble.

empty Both the head and foot are empty. In particular, no page numbers are printed.

headings This is the default for the `book` class. The foot is empty and the head contains the page number and names of the chapter section or subsection, depending on the document class and its options as given below:

CLASS	OPTION	LEFT PAGE	RIGHT PAGE
`book, report`	one-sided	—	*chapter*
	two-sided	*chapter*	*section*
`article`	one-sided	—	*section*
	two-sided	*section*	*subsection*

myheadings The same as `headings`, except that the 'section' information in the head are not predetermined, but to be given explicitly using the commands like `\markright` or `\markboth` as described below.

Moreover, we can customize the style for the *current page* only using the command

`\thispagestyle{`*style*`}`

where *style* is the name of one of the styles above. For example, the page number may be suppressed for the current page alone by the command `\thispagestyle{empty}`. Note that only the *printing* of the page number is suppressed. The next page will be numbered with the next number and so on.

2.2.1. HEADING DECLARATIONS

As we mentioned above, in the page style `myheadings`, we have to specify the text to appear on the head of every page. It is done with one of the commands

`\markboth{`*left head*`}{`*right head*`}`
`\markright{`*right head*`}`

where *left head* is the text to appear in the head on left-hand pages and *right head* is the text to appear on the right-hand

pages.

The `\markboth` command is used with the `twoside` option with even numbered pages considered to be on the left and odd numbered pages on the right. With `oneside` option, all pages are considered to be right-handed and so in this case, the command `\markright` can be used. These commands can also be used to override the default head set by the `headings` style.

Note that these give only a limited control over the head and foot. since the general format, including the font used and the placement of the page number, is fixed by LaTeX. Better customization of the head and foot are offered by the package fancyhdr, which is included in most LaTeX distributions.

2.3. PAGE NUMBERING

The style of page numbers can be specified by the command

```
\pagenumbering{...}
```

The possible arguments to this command and the resulting style of the numbers are given below:

`arabic`	Indo-Arabic numerals
`roman`	lowercase Roman numerals
`Roman`	upper case Roman numerals
`alph`	lowercase English letters
`Alph`	uppercase English letters

The default value is `arabic`. This command resets the page *counter*. Thus for example, to number all the pages in the 'Preface' with lowercase Roman numerals and the rest of the document with Indo-Arabic numerals, introduce a declaration `\pagenumbering{roman}` at the beginning of the Preface and issue the command `\pagestyle{arabic}` immediately after the first `\chapter` command. (The `\chapter{...}` command starts a new chapter. We will come to it soon.)

We can make the pages start with any number we want by the command

```
\setcounter{page}{number}
```

where *number* is the page number we wish the current page to have.

2.4. FORMATTING LENGTHS

Each page that LaTeX produces consists not only of a *head* and *foot* as discussed above but also a *body* (surprise!) containing the actual text. In formatting a page, LaTeX uses the width and heights of these parts of the page and various other lengths such as the left and right margins. The values of these lengths are set by the paper size options and the page format and style commands. For example, the page layout with values of these lengths for an odd page and even in this book are separately shown below. These lengths can all be changed with the command `\setlength`. For example,

```
\setlength{\textwidth}{15cm}
```

makes the width of text 15 cm. The package geometry gives easier interfaces to customize page format.

2.5. PARTS OF A DOCUMENT

We now turn our attention to the contents of the document itself. Documents (especially longer ones) are divided into chapters, sections and so on. There may be a title part (sometimes even a separate title page) and an abstract. All these require special typographic considerations and LaTeX has a number of features which automate this task.

2.5.1. TITLE

The "title" part of a document usually consists of the name of the document, the name of author(s) and sometimes a date. To produce a title, we make use of the commands

```
\title{document name}
\author{author names}
\date{date text}

\maketitle
```

Note that after specifying the arguments of `\title`, `\author` and `\date`, we must issue the command `\maketitle` for this part to be typeset.

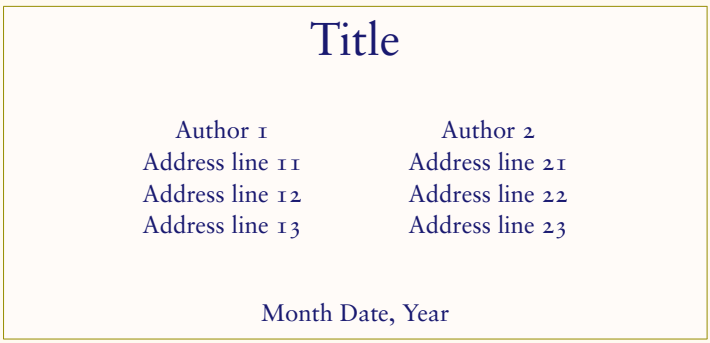
By default, all entries produced by these commands are centered on the lines in which they appear. If a title text is too long

to fit in one line, it will be broken automatically. However, we can choose the break points with the \\ command.

If there are several authors and their names are separated by the \and command, then the names appear side by side. Thus

```
\title{Title}
\author{Author 1\\
        Address line 11\\
        Address line 12\\
        Address line 13
        \and
        Author 2\\
        Address line 21\\
        Address line 22\\
        Address line 23}
 \date{Month Date, Year}
```

produces



If instead of \and, we use (plain old) \\, the names are printed one below another.

We may leave some of these arguments empty; for example, the command \date{ } prints no date. Note, however, that if you simply omit the \date command itself, the current date will be printed. The command

```
\thanks{footnote text}
```

can be given at any point within the \title, \author or \date. It puts a marker at this point and places the *footnote text* as a footnote. (The general method of producing a footnote is to type \footnote{*footnote text*} at the point we want to refer to.)

As mentioned earlier, the "title" is printed in a separate page for the document classes book and report and in the first page of

the document for the class `article`. (Also recall that this behavior can be modified by the options `titlepage` or `notitlepage`.)

2.5.2. ABSTRACT

In the document classes `article` and `report`, an abstract of the document in special format can be produced by the commands

```
\begin{abstract}
   Abstract Text
\end{abstract}
```

Note that we have to type the abstract ourselves. (There is a limit to what even LaTeX can do.) In the `report` class this appears on the separate title page and in the `article` class it appears below the title information on the first page (unless overridden by the title page option). This command is not available in the `book` class.

2.6. DIVIDING THE DOCUMENT

A book is usually divided into chapters and (if it is technical one), chapters are divided into sections, sections into subsections and so on. LaTeX provides the following hierarchy of *sectioning* commands in the `book` and `report` class:

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

Except for `\chapter` all these are available in `article` class also. For example, the heading at the beginning of this chapter was produced by

```
\chapter{The Document}
```

and the heading of this section was produced by

```
\section{Dividing the document}
```

To see the other commands in action, suppose at this point of text I type

```
\subsection{Example}
In this example, we show how subsections and subsubsections
```

```
are produced (there are no subsubsubsections). Note how the
subsections are numbered.

\subsubsection{Subexample}
Did you note that subsubsections are not numbered? This is
so in the \texttt{book} and \texttt{report} classes. In
the \texttt{article} class they too have numbers.
(Can you figure out why?)

\paragraph{Note}
Paragraphs  and subparagraphs do not have numbers. And
they have \textit{run-in} headings.

Though named ``paragraph'' we can have several paragraphs
of text within this.

\subparagraph{Subnote}
Subparagraphs have an additional indentation too.

And they  can also contain more than one paragraph of
text.
```

We get

2.6.1. EXAMPLE

In this example, we show how subsections and subsubsections are produced (there are no subsubsubsections). Note how the subsections are numbered.

Subexample

Did you note that subsubsections are not numbered? This is so in the `book` and `report` classes. In the `article` class they too have numbers. (Can you figure out why?)

Note Paragraphs and subparagraphs do not have numbers. And they have *run-in* headings. Though named "paragraph" we can have several paragraphs of text within this.

Subnote Subparagraphs have an additional indentation too. And they can also contain more than one paragraph of text.

2.6.2. MORE ON SECTIONING COMMANDS

In the `book` and the `report` classes, the `\chapter` command shifts to the beginning of a new page and prints the word "Chapter" and a number and beneath it, the name we have given in the argument of the command. The `\section` command produces two numbers (separated by a dot) indicating the chapter number and the section number followed by the name we have given. It does not produce any text like "Section". Subsections have three numbers indicating the chapter, section and subsection. Subsubsections and commands below it in the hierarchy do not have any numbers.

In the `article` class, `\section` is highest in the hierarchy and produces single number like `\chapter` in `book`. (It does not produce any text like "Section", though.) In this case, subsubsections also have numbers, but none below have numbers.

Each sectioning command also has a "starred" version which does not produce numbers. Thus `\section*{`*name*`}` has the same effect as `\section{`*name*`}`, but produces no number for this section.

Some books and longish documents are divided into parts also. LaTeX also has a `\part` command for such documents. In such cases, `\part` is the highest in the hierarchy, but it does not affect the numbering of the lesser sectioning commands.

You may have noted that LaTeX has a specific format for typesetting the section headings, such as the font used, the positioning, the vertical space before and after the heading and so on. All these can be customized, but it requires some TeXpertise and cannot be addressed at this point. However, the package sectsty provided some easy interfaces for tweaking some of these settings.

2.7. WHAT NEXT?

The task of learning to create a document in LaTeX is far from over. There are other things to do such as producing a bibliography and a method to refer to it and also at the end of it all to produce a table of contents and perhaps an index. All these can be done efficiently (and painlessly) in LaTeX, but they are matters for other chapters.

CHAPTER III

BIBLIOGRAPHY

3.1. INTRODUCTION

BIBLIOGRAPHY is the environment which helps the author to cross-reference one publication from the list of sources at the end of the document. LaTeX helps authors to write a well structured bibliography, because this is how LaTeX works—by specifying structure.

It is easy to convert the style of bibliography to that of a publisher's requirement, without touching the code inside the bibliography. We can maintain a bibliographic data base using the program BibTeX. While preparing the articles, we can extract the needed references in the required style from this data base. harvard and natbib are widely used packages for generating bibliography.

To produce bibliography, we have the environment **thebibliography**,[1] which is similar to the **enumerate** environment. Here we use the command `\bibitem` to separate the entries in the bibliography and use `\cite` to refer to a specific entry from this list in the document. This means that at the place of citation, it will produce number or author-year code connected with the list of references at the end.

```
\begin{thebibliography}{widest-label}
  \bibitem{key1}
  \bibitem{key2}
```

[1] Bibiliography environment need two compilations. In the first compilation it will generate file with `aux` extension, where `citation` and `bibcite` will be marked and in the second compilation `\cite` will be replaced by numeral or author-year code.

```
\end{thebibliography}
```

The `\begin{thebibliography}` command requires an argument that indicates the width of the widest label in the bibliography. If you know you would have between 10 and 99 citations, you should start with

```
\begin{thebibliography}{99}
```

You can use any two digit number in the argument, since all numerals are of the same width. If you are using customized labels, put the longest label in argument, for example,

`\begin{thebibliography}{Long-name}`.

Each entry in the environment should start with

`\bibitem{`*key1*`}`

If the author name is `Alex` and year `1991`, the key can be coded as `{ale91}` or some such mnemonic string[2]. This *key* is used to cite the publication within the document text. To cite a publication from the bibliography in the text, use the `\cite` command, which takes with the corresponding key as the argument. However, the argument to `\cite` can also be two or more keys, separated by commas.

`\cite{`*key1*`}` `\cite{`*key1,key2*`}`

In bibliography, numbering of the entries is generated automatically. You may also add a note to your citation, such as page number, chapter number etc. by using an optional argument to the `\cite` command. Whatever text appears in this argument will be placed within square brackets, after the label.

`\cite[page~25]{`*key1*`}`

See below an example of bibliography and citation. The following code

```
It is  hard to write unstructured and disorganised documents
using \LaTeX~\cite{les85}.It is interesting to typeset one
equation~\cite[Sec 3.3]{les85} rather than setting ten pages
of running matter~\cite{don89,rondon89}.
```

[2] Key can be any sequence of letters, digits and punctuation characters, except that it may not contain a comma (maximum 256 characters).

```
\begin{thebibliography}{9}
\bibitem{les85}Leslie Lamport, 1985. \emph{\LaTeX---A
 Document Preparation System---User's Guide and
 Reference Manual}, Addison-Wesley, Reading.

\bibitem{don89}Donald E. Knuth, 1989. \emph{Typesetting
 Concrete Mathematics}, TUGBoat, 10(1):31-36.

\bibitem{rondon89}Ronald L. Graham, Donald E. Knuth, and
 Ore Patashnik, 1989. \emph{Concrete Mathematics: A
 Foundation for Computer Science}, Addison-Wesley, Reading.
\end{thebibliography}
```

produces the following output:

It is hard to write unstructured and disorganised documents using LaTeX [1]. It is interesting to typeset one equation [1, Sec 3.3] rather than setting ten pages of running matter [2,3].

Bibliography

[1] Leslie Lamport, 1985. *LaTeX—A Document Preparation System—User's Guide and Reference Manual*, Addison-Wesley, Reading.

[2] Donald E. Knuth, 1989. *Typesetting Concrete Mathematics*, TUGBoat, 10(1):31-36.

[3] Ronald L. Graham, Donald E. Knuth, and Ore Patashnik, 1989. *Concrete Mathematics: A Foundation for Computer Science*, Addison-Wesley, Reading.

3.2. NATBIB

The natbib package is widely used for generating bibliography, because of its flexible interface for most of the available bibliographic styles. The natbib package is a re-implementation of the LaTeX `\cite` command, to work with both author–year and numerical citations. It is compatible with the standard bibliographic style files, such as plain.bst, as well as with those for harvard, apalike, chicago, astron, authordate, and of course natbib. To load the package; use the command.

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

3.2.1. OPTIONS FOR NATBIB

`round`	(default) for round parentheses
`square`	for square brackets
`curly`	for curly braces
`angle`	for angle brackets
`colon`	(default) to separate multiple citations with colons
`comma`	to use commas as separators
`authoryear`	(default) for author–year citations
`numbers`	for numerical citations
`super`	for superscripted numerical citations, as in *Nature*
`sort`	orders multiple citations into the sequence in which they appear in the list of references
`sort&compress`	as `sort` but in addition multiple numerical citations are compressed if possible (as 3–6, 15)
`longnamesfirst`	makes the first citation of any reference the equivalent of the starred variant (full author list) and subsequent citations normal (abbreviated list)
`sectionbib`	redefines `\thebibliography` to issue `\section*` instead of `\chapter*`; valid only for classes with a `\chapter` command; to be used with the `chapterbib` package
`nonamebreak`	keeps all the authors' names in a citation on one line; causes overfull hboxes but helps with some `hyperref` problems.

You can set references in the *Nature style* of citations (superscripts) as follows

```
\documentclass{article}
  \usepackage{natbib}
  \citestyle{nature}
\begin{document}
  . . . . . .
  . . . . . .
\end{document}
```

3.2.2. BASIC COMMANDS

The natbib package has two basic citation commands, `\citet` and `\citep` for *textual* and *parenthetical* citations, respectively. There also exist the starred versions `\citet*` and `\citep*` that print the full author list, and not just the abbreviated one. All of these may take one or two optional arguments to add some text before and after the citation.

Normally we use author name and year for labeling the bibliography.

```
\begin{thebibliography}{widest-label}
  \bibitem[Lamport(1985)]{les85}Leslie Lamport, 1985.
    \emph{\LaTeX---A Document Preparation}...
  \bibitem[Knuth(0000)]{don89}Donald E. Knuth, 1989.
    \emph{Typesetting Concrete Mathematics},...
  \bibitem[Ronald, Donald and Ore(1989)]{rondon89}Ronald
     L. Graham, ...
\end{thebibliography}
```

Year in parentheses is mandatory in optional argument for bibitem. If year is missing in any of the bibitem, the whole author–year citation will be changed to numerical citation. To avoid this, give '(0000)' for year in optional argument and use partial citations (`\citeauthor`) in text.

Don't put 'space character' before opening bracket of year in optional argument.

`\citet{ale91}`	⇒ Alex et al. (1991)
`\citet[chap.~4]{ale91}`	⇒ Alex et al. (1991, chap. 4)
`\citep{ale91}`	⇒ (Alex et al., 1991)
`\citep[chap.~4]{ale91}`	⇒ (Alex et al., 1991, chap. 4)
`\citep[see][]{ale91}`	⇒ (see Alex et al., 1991)
`\citep[see][chap.~4]{ale91}`	⇒ (see Alex et al., 1991, chap. 4)
`\citet*{ale91}`	⇒ Alex, Mathew and Ravi (1991)
`\citep*{ale91}`	⇒ (Alex, Mathew and Ravi, 1991)

3.2.3. MULTIPLE CITATIONS

Multiple citations may be made as usual, by including more than one citation key in the `\cite` command argument.

```
\citet{ale91,rav92}    ⇒ Alex et al. (1991); Ravi et al. (1992)
\citep{ale91,rav92}    ⇒ (Alex et al., 1991; Ravi et al. 1992)
\citep{ale91,ale92}    ⇒ (Alex et al., 1991, 1992)
\citep{ale91a,ale91b} ⇒ (Alex et al., 1991a,b)
```

3.2.4. NUMERICAL MODE

These examples are for author–year citation mode. In numerical mode, the results are different.

```
\citet{ale91}                  ⇒  Alex et al. [5]
\citet[chap.~4]{ale91}         ⇒  Alex et al. [5, chap. 4]
\citep{ale91}                  ⇒  [5]
\citep[chap.~4]{ale91}         ⇒  [5, chap. 4]
\citep[see][]{ale91}           ⇒  [see 5]
\citep[see][chap.~4]{ale91}    ⇒  [see 5, chap. 4]
\citep{ale91a,ale91b}          ⇒  [5, 12]
```

3.2.5. SUPPRESSED PARENTHESES

As an alternative form of citation, `\citealt` is the same as `\citet` but *without any parentheses*. Similarly, `\citealp` is `\citep` with the parentheses turned off. Multiple references, notes, and the starred variants also exist.

```
\citealt{ale91}          ⇒ Alex et al. 1991
\citealt*{ale91}         ⇒ Alex, Mathew, and Ravi 1991
\citealp{ale91}          ⇒ Alex., 1991
\citealp*{ale91}         ⇒ Alex, Mathew, and Ravi, 1991
\citealp{ale91,ale92}    ⇒ Alex et al., 1991; Alex et al., 1992
\citealp[pg.~7]{ale91}   ⇒ Alex., 1991, pg. 7
\citetext{short comm.}   ⇒ (short comm.)
```

The `\citetext` command allows arbitrary text to be placed in the current citation parentheses. This may be used in combination with `\citealp`.

3.2.6. PARTIAL CITATIONS

In author–year schemes, it is sometimes desirable to be able to refer to the authors without the year, or vice versa. This is provided with the extra commands

`\citeauthor{ale91}`	⇒	Alex et al.
`\citeauthor*{ale91}`	⇒	Alex, Mathew, and Ravi
`\citeyear{ale91}`	⇒	1991
`\citeyearpar{ale91}`	⇒	(1991)

3.2.7. CITATIONS ALIASING

Sometimes one wants to refer to a reference with a special designation, rather than by the authors, i.e. as Paper I, Paper II. Such aliases can be defined and used, textually and/or parenthetically with:

```
\defcitealias{jon90}{Paper~I}
```

`\citetalias{ale91}`	⇒	Paper I
`\citepalias{ale91}`	⇒	(Paper I)

These citation commands function much like `\citet` and `\citep`: they may take multiple keys in the argument, may contain notes, and are marked as hyperlinks.

3.2.8. SELECTING CITATION STYLE AND PUNCTUATION

Use the command `\bibpunct` with one optional and six mandatory arguments:

1. The opening bracket symbol, `default = (`
2. The closing bracket symbol, `default = )`
3. The punctuation between multiple citations, `default = ;`
4. The letter 'n' for numerical style, or 's' for numerical superscript style, any other letter for author–year, `default = author-year;`
5. The punctuation that comes between the author names and the year
6. The punctuation that comes between years or numbers when common author lists are suppressed (`default = ,`);

The optional argument is the character preceding a postnote, default is a comma plus space. In redefining this character, one must include a space if that is what one wants.

Example 1

```
\bibpunct{[}{]}{,}{a}{}{;}
```

changes the output of

```
\citep{jon90,jon91,jam92}
```

into

[Jones et al. 1990; 1991, James et al. 1992].

Example 2

```
\bibpunct[;]{(}{)}{,}{a}{}{;}
```

changes the output of

```
\citep[and references therein]{jon90}
```

into

(Jones et al. 1990; and references therein).

CHAPTER IV

BIBLIOGRAPHIC DATABASES

IBLIOGRAPHIC database is a database in which all the useful bibliographic entries can be stored. The information about the various publications is stored in one or more files with the extension `.bib`. For each publication, there is a *key* that identifies it and which may be used in the text document to refer to it. And this is available for all documents with a list of reference in the field. This database is useful for the authors/researchers who are constantly referring to the same publications in most of their works. This database system is possible with the BibTeX program supplied with the LaTeX package.

4.1. THE BIBTEX PROGRAM

BibTeX is an auxiliary program to LaTeX that automatically constructs a bibliography for a LaTeX document from one or more databases. To use BibTeX, you must include in your LaTeX input file a `\bibliography` command whose argument specifies one or more files that contain the database. For example

```
\bibliography{database1,database2}
```

The above command specifies that the bibliographic entries are obtained from *database1.bib* and *database2.bib*. To use BibTeX, your LaTeX input file must contain a `\bibliographystyle` command. This command specifies the *bibliography style*, which determines the format of the source list. For example, the command

```
\bibliographystyle{plain}
```

specifies that entries should be formatted as specified by the `plain` bibliography style (`plain.bst`). We can put `\bibliographystyle` command anywhere in the document after the `\begin{document}` command.

4.2. BIBTEX STYLE FILES

plain Standard BIBTEX style. Entries sorted alphabetically with numeric labels.

unsrt Standard BIBTEX style. Similar to plain, but entries are printed in order of citation, rather than sorted. Numeric labels are used.

alpha Standard BIBTEX style. Similar to plain, but the labels of the entries are formed from the author's name and the year of publication.

abbrv Standard BIBTEX style. Similar to plain, but entries are more compact, since first names, month, and journal names are abbreviated.

acm Alternative BIBTEX style, used for the journals of the Association for Computing Machinery. It has the author name (surname and first name) in small caps, and numbers as labels.

apalike Alternative BIBTEX style, used by the journals of the American Psychology Association. It should be used together with the LATEX apalike package. The bibliography entries are formatted alphabetically, last name first, each entry having a hanging indentation and no label.

Examples of some other style files are:

abbrv.bst, abstract.bst, acm.bst, agsm.bst, alpha.bst, amsalpha.bst, authordate*i*.bst, authordate1-4.sty, bbs.bst, cbe.bst, cell.bst, dcu.bst, harvard.sty, ieeetr.bst, jtb.bst, kluwer.bst, named.bst, named.sty, natbib.sty, natbib.bst, nature.sty, nature.bst, phcpc.bst, phiaea.bst, phjcp.bst, phrmp.bst, plainyr.bst, siam.bst

Various organisations or individuals have developed style files that correspond to the house style of particular journals or

editing houses. We can also customise a bibliography style, by making small changes to any of the `.bst` file, or else generate our own using the `makebst` program.

4.2.1. STEPS FOR RUNNING BIBTEX WITH LATEX

1. Run LaTeX, which generates a list of `\cite` references in its auxiliary file, `.aux`.
2. Run BibTeX, which reads the auxiliary file, looks up the references in a database (one or more `.bib` files, and then writes a file (the `.bbl` file) containing the formatted references according to the format specified in the style file (the `.bst` file). Warning and error messages are written to the log file (the `.blg` file). It should be noted that BibTeX never reads the original LaTeX source file.
3. Run LaTeX again, which now reads the `.bbl` reference file.
4. Run LaTeX a third time, resolving all references.

Occasionally the bibliography is to include publications that were *not* referenced in the text. These may be added with the command

`\nocite{`*key*`}`

given anywhere within the main document. It produces no text at all but simply informs BibTeX that this reference is also to be put into the bibliography. With `\nocite{*}`, *every* entry in all the databases will be included, something that is useful when producing a list of all entries and their keys.

After running BibTeX to make up the `.bbl` file, it is necessary to process LaTeX *at least twice* to establish both the bibliography and the in-text reference labels. The bibliography will be printed where the `\bibliography` command is issued; it infact inputs the `.bbl` file.

4.3. CREATING A BIBLIOGRAPHIC DATABASE

Though bibliographic database creation demands more work than typing up a list of references with the environment; it has a great advantage that, the entries need to be included in the database only once and are then available for all future publications even if a different bibliography style is demanded in later works, all the information is already on hand in the

database for BibTeX to write a new `thebibliography` environment in another format. Given below is a specimen of an entry in bibliographic database:

```
@BOOK{knuth:86a,
  AUTHOR           = "Donald E. Knuth",
  TITLE            = {The \TeX{}book},
  EDITION          = "third",
  PUBLISHER        = "Addison-Wesley",
  ADDRESS          = {Reading, MA},
  YEAR             = 1986 }
```

The first word, prefixed `@`, determines the *entry_type*. The *entry_type* is followed by the reference information for that entry enclosed in curly braces `{ }`. The very first entry is the *key* for the whole reference by which it is referred to in the `\cite` command. In the above example it is `knuth:86a`. The actual reference information is then entered in various *fields*, separated from one another by commas. Each *field* consists of a *field_name*, an = sign, with optional spaces on either side, and the *field text*. The *field_names* shows above are `AUTHOR`, `TITLE`, `PUBLISHER`, `ADDRESS`, and `YEAR`. The *field text* must be enclosed either in curly braces or in double quotation marks. However, if the text consists solely of a number, as for `YEAR` above, the braces or quotation marks may be left off.

For each entry type, certain fields are *required*, others are *optional*, and the rest are *ignored*. These are listed with the description of the various entry types below. If a required field is omitted, an error message will appear during the BibTeX run. Optional fields will have their information included in the bibliography if they are present, but they need not be there. Ignored fields are useful for including extra information in the database that will not be output, such as a comment or an abstract of a paper. Ignored fields might also be ones that are used by other database programs.

The general syntax for entries in the bibliographic database reads

```
@entry_type{key,
  field_name = {field text},
  ....
  field_name = {field text} }
```

The names of the *entry_types* as well as the *field_names* may be written in capitals or lower case letters, or in a combination of both. Thus `@BOOK`, `@book`, and `@bOOk` are all acceptable variations.

The outermost pair of braces for the entire entry may be either curly braces `{ }`, as illustrated, or parentheses `( )`. In the latter case, the general syntax reads

```
@entry_type(key, ... ..)
```

However, the *field text* may only be enclosed within curly braces `{...}` or double quotation marks `"..."` as shown in the example above.

The following is a list of the standard entry types in alphabetical order, with a brief description of the types of works for which they are applicable, together with the required and optional fields that they take.

`@article`:	Entry for an article from a journal or magazine.
required:	author, title, journal, year.
optional:	volume, number, pages, month, note.
`@book`:	Entry for a book with a definite publisher.
required:	author or editor, title, publisher, year.
optional:	volume or number, series, address, edition, month, note.
`@booklet`:	Entry for a printed and bound work without the name of a publisher or sponsoring organisation.
required:	title.
optional:	author, howpublished, address, month, year, note.
`@conference`:	Entry for an article in conference proceedings.
required:	author, title, booktitle, year.
optional:	editor, volume or number, series, pages, address, month, organisation, publisher, note.
`@inbook`:	Entry for a part (chapter, section, certain pages) of a book.
required:	author or editor, title, chapter and/or pages, publisher, year.
optional:	volume or number, series, type, address, edition, month, note.
`@incollection`:	Entry for part of a book that has its own title.

required: author, title, booktitle, publisher, year.
optional: editor, volume or number, series, type, chapter, pages, address, edition, month, note.

`@inproceedings`: Entry for an article in conference proceedings.
required: author, title, booktitle, year.
optional: editor, volume or number, series, pages, address, month, organisation, publisher, note.

`@manual`: Entry for technical documentation.
required: title.
optional: author, organisation, address, edition, month, year, note.

`@masterthesis`: Entry for a Master's thesis.
required: author, title, school, year.
optional: type, address, month, note.

`@misc`: Entry for a work that does not fit under any of the others.
required: none.
optional: author, title, howpublished, month, year, note.

`@phdthesis`: Entry for a PhD thesis.
required: author, title, school, year.
optional: type, address, month, note.

`@proceedings`: Entry for conference proceedings.
required: title, year.
optional: editor, volume or number, series, address, month, organisation, publisher, note.

`@unpublished`: Entry for an unpublished work with an author and title.
required: author, title, note.
optional: month, year.

4.3.1. EXAMPLE OF A LaTeX FILE (SAMPLE.TEX) USING BIBLIOGRAPHICAL DATABASE (BSAMPLE.BIB)

```
\documentclass{article}
\pagestyle{empty}
\begin{document}

\section*{Example of Citations of Kind \texttt{plain}}
Citation of a normal book~\cite{Eijkhout:1991} and an
```

```
edited book~\cite{Roth:postscript}. Now we cite an article
written by a single~\cite{Felici:1991} and by multiple
authors~\cite{Mittlebatch/Schoepf:1990}. A reference to
an article inside proceedings~\cite{Yannis:1991}.  We
refer to a manual~\cite{Dynatext} and a technical
report~\cite{Knuth:WEB}. A citation of an unpublished
work~\cite{EVH:Office}. A reference to a chapter in
a book~\cite{Wood:color} and to a PhD thesis
\cite{Liang:1983}.  An example of multiple
citations~\cite{Eijkhout:1991,Roth:postscript}.

\bibliographystyle{plain} %% plain.bst
\bibliography{bsample}    %% bsample.bib
\end{document}
```

4.3.2. PROCEDURE FOR PRODUCING REFERENCES FOR THE ABOVE FILE SAMPLE.TEX WHICH USES BIBLIOGRAPHIC DATA BASE BSAMPLE.BIB

```
$ latex sample     % 1st run of LaTeX

$ bibtex sample    % BibTeX run
                   % Then sample.bbl file will
                   % be produced

$ latex sample     % 2nd run of LaTeX
```

If still unresolved citation references

```
$ latex sample     % 3rd run of LaTeX
```

CHAPTER V

TABLE OF CONTENTS, INDEX AND GLOSSARY

5.1. TABLE OF CONTENTS

TABLE OF CONTENTS is a special list which contains the section numbers and corresponding headings as given in the standard form of the sectioning commands, together with the page numbers on which they begin. Similar lists exist containing reference information about the floating elements in a document, namely, the *list of tables* and *list of figures*. The structure of these lists is simpler, since their contents, the captions of the floating elements, all are on the same level.

Standard LaTeX can automatically create these three contents lists. By default, LaTeX enters text generated by one of the arguments of the sectioning commands into the `.toc` file. Similarly, LaTeX maintains two more files, one for the list of figures (`.lof`) and one for the list of tables (`.lot`), which contain the text specified as the argument of the `\caption` command for figures and tables.

`\tableofcontents` produces a table of contents. `\listoffigures` and `\listoftables` produce a list of figures and list of tables respectively. These lists are printed at the point where these commands are issued. Occasionally, you may find that you do not like the way LaTeX prints a table of contents or a list of figures or tables. You can fine-tune an individual entry by using the optional arguments to the sectioning command or `\caption` command that generates it. Formatting commands can also be introduced with the `\addtocontents`. If all else fails, you can edit the `.toc`, `lof`, `lot` files yourself. Edit these files only when preparing the final version of your document, and use a

\nofiles command to suppress the writing of new versions of the files.

5.1.1. ADDITIONAL ENTRIES

The *-form sectioning commands are not entered automatically in the table of contents. LaTeX offers two commands to insert such information directly into a contents file:

```
\addtocontents{file}{text}
\addcontentsline{file}{type}{text}
```

file The extension of the contents file, usually toc, lof or lot.

type The type of the entry. For the toc file the *type* is normally the same as the heading according to the format of which an entry must be typeset. For the lof or lot files, figure or table is specified.

text The actual information to be written to the *file* mentioned. LaTeX commands should be protected by \protect to delay expansion

The \addtocontents command does not contain a *type* parameter and is intended to enter *user-specific* formatting information. For example, if you want to generate additional spacing in the middle of a table of contents, the following command can be issued:

```
\addtocontents{toc}{\protect\vspace{2ex}}
```

The \addcontentsline instruction is usually invoked *automatically* by the document sectioning commands, or by the \caption commands. If the entry contains numbered text, then \numberline must be used to separate the section number (*number*) from the rest of the text for the entry (*heading*) in the *text* parameter:

```
\protect\numberline{number}{heading}
```

For example, a \caption command inside a figure environment saves the text annotating the figure as follows:

```
\addcontentsline{lof}{figure}
  {\protect\numberline{\thefigure} captioned text}
```

Sometimes \addcontentsline is used in the source to complement the actions of standard LaTeX. For instance, in the case

of the starred form of the section commands, no information is written to the .toc file. So if you do not want a heading number (starred form) but an entry in the .toc file you can write something like:

```
\chapter*{Forward}
\addcontentsline{toc}{chapter}{\numberline{}Forward}
```

This produces an indented "chapter" entry in the table of contents, leaving the space where the chapter number would go free. Omitting the \numberline command would typeset the word "Forward" flush left instead.

5.1.2. TYPESETTING A CONTENTS LIST

As discussed above, contents lists consist of entries of different types, corresponding to the structural units that they represent. Apart from these standard entries, these lists may contain any commands. A standard entry is specified by the command:

\contentsline{*type*}{*text*}{*page*}

type — Type of the entry, e.g. section, or figure.

text — Actual text as specified in the argument of the sectioning or \caption commands.

page — Page number.

Note that section numbers are entered as a parameter of the \numberline command to allow formatting with the proper indentation. It is also possible for the user to create a table of contents by hand with the help of the command \contentsline. For example:

```
\contentsline {section}
  {\numberline {2.4}Structure of the Table of Contents}{31}
```

To format an entry in the table of contents files, standard LaTeX makes use of the following command:

\@dottedtocline{*level*}{*indent*}{*numwidth*}{*text*}{*page*}

The last two parameters coincide with those of \contentsline, since the latter usually invokes \@dottedtocline command. The other parameters are the following:

level The nesting level of an entry. This parameter allows the user to control how many nesting levels will be displayed. Levels greater than the value of counter `tocdepth` will not appear in the table of contents.

indent This is total indentation from the left margin.

numwidth The width of the box that contains the number if *text* has a `\numberline` command. This is also the amount of extra indentation added to the second and later lines of a multiple line entry.

Additionally, the command `\@dottedtocline` uses the following formatting parameters, which specify the visual appearance of all entries:

`\@pnumwidth` The width of the box in which the page number is set.

`\@tocmarg` The indentation of the right margin for all but the last line of multiple line entries. Dimension, but changed with `\renewcommand`.

`\@dotsep` The separation between dots, in `mu` (math units). It is a pure number (like 1.7 or 2). By making this number large enough you can get rid of the dots altogether. Changed with `\renewcommand` as well.

5.1.3. MULTIPLE TABLES OF CONTENTS

The minitoc package, initially written by Nigel Ward and Dan Jurafsky and completely redesigned by Jean-Pierre Drucbert, creates a mini-table of contents (a "minitoc") at the beginning of each chapter when you use the book or report classes.

The mini-table of contents will appear at the beginning of a chapter, after the `\chapter` command. The parameters that govern the use of this package are discussed below:

Table 5.1: Summary of the `minitoc` parameters

`\dominitoc`	Must be put just in front of `\tableofcontents`, to initialize the minitoc system (Mandatory).

`\faketableofcontents`	This command replaces `\tableofcontents` when you want minitocs but not table of contents.
`\minitoc`	This command must be put right after each `\chapter` command where a minitoc is desired.
`\minitocdepth`	A LaTeX counter that indicates how many levels of headings will be displayed in the minitoc (default value is `2`).
`\mtcindent`	The length of the left/right indentation of the minitoc (default value is `24pt`).
`\mtcfont`	Command defining the font that is used for the minitoc entries (The default definition is a small roman font).

For each mini-table, an auxiliary file with extension `.mtc<N>` where `<N>` is the chapter number, will be created.

By default, these mini-tables contain only references to sections and subsections. The `minitocdepth` counter, similar to `tocdepth`, allows the user to modify this behaviour.

As the minitoc takes up room on the first page(s) of a chapter, it will alter the page numbering. Therefore, three runs normally are needed to get correct information in the mini-table of contents.

To turn off the `\minitoc` commands, merely replace the package `minitoc` with `minitocoff` on your `\usepackage` command. This assures that all `\minitoc` commands will be ignored.

5.2. INDEX

To find a topic of interest in a large document, book, or reference work, you usually turn to the table of contents or, more often, to the index. Therefore, an index is a very important part of a document, and most users' entry point to a source of information is precisely through a pointer in the index. The most generally used index preparation program is *MakeIndex*.

Each \index command causes LaTeX to write an entry in the .idx file. This command writes the text given as an argument, in the .idx file. This .idx will be generated only if we give \makeindex command in the preamble otherwise it will produce nothing.

\index{*index_entry*}

To generate index follow the procedure given below:

1. Tag the words inside the document, which needs to come as index, as an argument of \index command.
2. Include the makeidx package with an \usepackage command and put \makeindex command at the preamble.
3. Put a \printindex command where the index is to appear, normally before \end{document} command.
4. LaTeX *file*. Then a raw index (file.idx) will be generated.
5. Then run makeindex. (makeindex *file.idx* or makeindex *file*). Then two more files will be generated, *file.ind* which contains the index entries and *file.ilg*, a transcript file.
6. Then run LaTeX again. Now you can see in the dvi that the index has been generated in a new page.

5.2.1. SIMPLE INDEX ENTRIES

Each \index command causes LaTeX to write an entry in the .idx file. For example

\index{*index_entry*}

5.2.2. SUB ENTRIES

Up to three levels of index entries (main, sub, and subsub entries) are available with LaTeX-MakeIndex. To produce such entries, the argument of the \index command should contain both the main and subentries, separated by ! character.

Page 5: \index{*dimensions!rule!width*}

This will come out as

dimensions
    rule
        width, 5

```
Page vi:    \index{animal}
Page 5:     \index{animal}
Page 6:     \index{animal}
Page 7:     \index{animal}
Page 11:    \index{animalism|see{animal}}
Page 17:    \index{animal@\emph{animal}}
            \index{mammal|textbf}
Page 26:    \index{animal!mammal!cat}
Page 32:    \index{animal!insect}
```

(a) The input file

```
\indexentry{animal}{vi}
\indexentry{animal}{5}
\indexentry{animal}{6}
\indexentry{animal}{7}
\indexentry{animalism|seeanimal}{11}
\indexentry{animal@\emph{animal}}{17}
\indexentry{mammal|textbf}{17}
\indexentry{animal!mammal!cat}{26}
\indexentry{animal!insect}{32}
```

(b) The `.idx` file

Figure 5.1: Stepwise development of index processing (*Contd...*)

5.2.3. PAGE RANGES AND CROSS-REFERENCES

You can specify a page range by putting the command `\index{...|(}` at the beginning of the range and `\index{...|)}`

```
\begin{theindex}
  \item animal, vi, 5–7
    \subitem insect, 32
    \subitem mammal
      \subsubitem cat, 26
  \item \emph{animal}, 17
  \item animalism, \see{animal}{11}
\indexspace
  \item mammal, \textbf{17}
\end{theindex}
```

(c) The `.ind` file

Figure 5.1: Stepwise development of index processing (*contd...*)

animal, vi 5–7
    insect, 32
    mammal
        cat, 26

animal, 17
animalism, *see* animal
mammal, **17**

(d) The typeset output

Figure 5.1: Stepwise development of index processing

```
Page ii:    \index{table|(}
Page xi:    \index{table|)}
Page 5:     \index{fonts!PostScript|(}
            \index{fonts!PostScript|)}
Page 13     \index{fonts!Computer Modern |(}
Page 14:    \index{table}
Page 17:    \index{fonts!math|see{math, fonts}}
Page 21:    \index{fonts!Computer Modern}
Page 25:    \index{fonts!Computer Modern|)}
```

fonts
    Computer Modern, 13–25
    math, *see* math, fonts
    PostScript, 5
table, ii–xi, 14

Figure 5.2: Page range and cross-referencing

at the end of the range. Page ranges should span a homogeneous numbering scheme (e.g., Roman and Arabic page numbers cannot fall within the same range).

You can also generate cross-reference index entries without page numbers by using the see encapsulator. Since "see" entry does not print any page number, the commands `\index{...|see{...}}` can be placed anywhere in the input file after the `\begin{document}` command. For practical reasons, it is convenient to group all such cross-referencing commands in one place.

```
Page ii:        \index{tabular|textbf}
Page 5:         \index{ninety-five}
Page 7:         \index{tabbing}
Page 14:        \index{delta}
Page 16:        \index{delta wing}
Page 19:        \index{flower@\textbf{flower}}
Page 21:        \index{tabular|textit}
Page 22:        \index{tabular|nn}
Page 23:        \index{delta@$\delta$}
                \index{tabular@\texttt{tabular}
                  {environment}}
Page 26:        \index{ninety}
Page 28:        \index{ninety@xc}
Page 34:        \index{tabbing|(textit}
Page 36:        \index{tabbing|)}
```

Figure 5.3: Controlling the presentation form (contd. . .)

delta, 14
δ, 23
delta wing, 16
flower, 19
ninety, 26
xc, 28
ninety-five, 5
tabbing, *7*, *34–36*
tabular, **ii**, *21*, 22n
`tabular` environment, 23

Figure 5.3: Controlling the presentation form

5.2.4. CONTROLLING THE PRESENTATION FORM

Sometimes you may want to sort an entry according to a key, while using a different visual representation for the typesetting, such as Greek letters, mathematical symbols, or specific typographic forms. This function is available with the syntax: *key@visual*, where *key* determines the alphabetical position and the string *value* produces the typeset text of the entry.

For some indexes certain page numbers should be formatted specially, with an italic page number (for example) indicating a primary reference, and an *n* after a page number denoting that the item appears in a footnote on that page. *MakeIndex* allows you to format an individual page number in any way you want by using the encapsulator syntax specified | character. What follows the | sign will "encapsulate" or enclose the

```
          \index{bar@\texttt{"|}|see{vertical bar}}
Page 1:   \index{quote (\verb+""+)}
          \index{quote@\texttt{""}sign}
Page 2:   \index{atsign@\texttt{@"} sign}
Page 3:   \index{maedchen@M\"{+a}dchen}
Page 4:   \index{exclamation ("!)}
Page 5:   \index{exclamation ("!)!Ah"!}
```

@ sign, 2
|, *see* vertical bar
exclamation (!), 4
    Ah!, 5
Mädchen, 3
quote ("), 1
" sign, 1

Figure 5.4: Printing those special characters

page number associated with the index entry. For instance, the command `\index{keyword|xxx}` will produce a page number of the form `\xxx{n}`, where *n* is the page number in question. Similarly, the command `\index{keyword|(xxx)}` will generate a page range of the form `\xxx{n-m}`

```
\newcommand{\nn}[1]{#1n}
```

5.2.5. PRINTING THOSE SPECIAL CHARACTERS

To typeset one of the characters having a special meaning to *MakeIndex* (`!, ", @, or |`) in the index, precede it with a `"` character. More precisely, any character is said to be quoted if it follows an unquoted `"` that is not part of a `\"` command. The latter case is for allowing umlaut characters. Quoted `!, @, ", or |` characters are treated like ordinary characters, losing their special meaning. The `"` preceding a quoted character is deleted before the entries are alphabetised.

5.3. GLOSSARY

A 'glossary' is a special index of terms and phrases alphabetically ordered together with their explanations. To help set up a glossary, LaTeX offers the commands

`\makeglossary`	in the preamble and
`\glossary{`*glossary-entry*`}`	in the text part

which function just like the commands for making up an index register. The entries are written to a file with extension `.glo` after the command `\makeglossary` has been given in the preamble. The form of these file entries from each `\glossary` command is

```
\glossaryentry\textit{glossary-entry}{page number}
```

The information in the `.glo` file can be used to establish a glossary. However, there is no equivalent to the `theindex` environment for a glossary, but a recommended structure is the `description` environment or a special list environment.

CHAPTER VI

DISPLAYED TEXT

THERE are many instances in a document when we want to visually separate a portion of text from its surrounding material. One method of doing this is to typeset the distinguished text with added indentation. It is called *displaying*. LaTeX has various constructs for displaying text depending the nature of the displayed text.

6.1. BORROWED WORDS

Quotations are often used in a document, either to add weight to our arguments by referring to a higher authority or because we find that we cannot improve on the way an idea has been expressed by someone else. If the quote is a one-liner, we can simply include it within double-quotes and be done with it (remember how to use quotes in TeX?) But if the quotation is several lines long, it is better to display it. Look at the formatted example given in Figure 6.1: This was typeset as shown below

```
Some mathematicians elevate the spirit of Mathematics  to
a kind of intellectual aesthetics. It is best voiced  by
Bertrand Russell in the following lines.
 \begin{quote}
   The true spirit of .......................from which
   all great work springs.
 \end{quote}
```

Note that here we give instructions to TeX to typeset some material in a separate paragraph with additional indentation on either side and indicate the start and end of material requiring special treatment, by means of the commands

Some mathematicians elevate the spirit of Mathematics to a kind of intellectual aesthetics. It is best voiced by Bertrand Russell in the following lines.

> The true spirit of delight, the exaltation, the sense of being more than man, which is the touchstone of the highest excellence, is to be found in Mathematics as surely as in poetry... Real life is, to most men, a long second best, a perpetual compromise between the ideal and the possible; but the world of pure reason knows no compromise, no practical limitations, no barriers to the creative activity embodying in splendid edifices the passionate aspiration after the perfect, from which all great work springs.

Yes, to men like Russell, Mathematics is more of an art than science.

Figure 6.1: Quotation spanning several lines.

```
\begin{quote} ... \end{quote}
```

This is an example of what is known in LaTeX parlance as an *environment*. Environments are used to delimit passages requiring special typographic treatments and to give instructions to LaTeX on how to typeset it. The general form of an environment is of the form

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

where *name* is the name of the environment and signifies to LaTeX the type of typographic treatment required (deliberate attempt at a pun, that).

The quoted part in this example is a single paragraph. If the quotation runs into several paragraphs, we must use the `quotation` environment, by enclosing the quotation within

`\begin{quotation}` and `\end{quotation}`.

As usual, paragraphs are separated by blank lines while typing the source file.

6.2. POETRY IN TYPESETTING

LaTeX can write poetry...well almost; if you write poems, TeX can nicely typeset it for you. (I have also heard some TeX wizards saying Knuth's code is sheer poetry!) Look at the passage below:

> Contrary to popular belief, limericks are not always ribald. Some of them contain mathematical concepts:
>
> A mathematician once confided
> That a Möbius band is one sided
> You'll get quite a laugh
> If you cut it in half
> For it stays in one piece when divided
>
> There is an extension of this to Klein's bottle also.

This was typeset as follows:

```
Contrary to popular belief, ...  mathematical concepts:
\begin{verse}
  A mathematician once confided\\
  That a M\"obius band is one sided\\
  You'll get quite a laugh\\
  If you cut it in half\\
  For it stays in one piece when divided
\end{verse}
There is an extension of this to Klein's bottle also.
```

Note that line breaks are forced by the symbol `\\`. Different stanzas are separated in the input by one (or more) blank lines. If you do not want TeX to start a new page at a particular line break (if you want to keep rhyming couplets together in one page, for example), then use `\\*` instead of plain `\\`. Again, if you want more space between lines than what LaTeX deems fit, then use `\\` with an optional *length* as in `\\[5pt]` which adds an *extra* vertical space of 5 points between the lines. You can also type `\\*[5pt]`, whose intention should be obvious by now.

6.3. MAKING LISTS

Lists are needed to keep some semblance of order in a chaotic world and LaTeX helps us to typeset them nicely. Also, there

are different kinds of lists available by default and if none of them suits your need, there are facilities to tweak these or even design your own. Let us first have a look at the types of lists LaTeX provides.

6.3.1. SAYING IT WITH BULLETS

The `itemize` environment gives us a bullet-list. For example it produces something like this:

One should keep the following in mind when using TeX

- TeX is a typesetting language and not a word processor
- TeX is a program and not an application
- There is no meaning in comparing TeX to a word processor, since the design purposes are different

Being a program, TeX offers a high degree of flexibility.

The input which produces this is given below:

```
One should keep the following in mind when using \TeX
\begin{itemize}
\item \TeX\ is a typesetting language and not a word
      processor
\item \TeX\ is a program and not an application
\item There is no meaning in comparing \TeX\ to a
      word processor, since the design purposes are
      different
\end{itemize}
Being a program, \TeX\ offers a high degree of
flexibility.
```

The `\begin{itemize}` ... `\end{itemize}` pair signifies we want a bullet-list of the enclosed material. Each item of the list is specified by (what else?) an `\item` command.

We can have lists within lists. For example:

One should keep the following in mind when using TEX

- TEX is a typesetting language and not a word processor
- TEX is a program and not an application
- There is no meaning in comparing TEX to a word processor, since the design purposes are different
- TEX is the natural choice in one of these situations
  - If we want to typeset a document containing lot of Mathematics
  - If we want our typed document to look beautiful

Being a program, TEX offers a high degree of flexibility.

It is produced by the input below:

```
One should keep the following in mind when using \TeX
\begin{itemize}
\item \TeX\ is a typesetting language and not a  word
      processor
\item \TeX\ is a program and not an application
\item There is no meaning in comparing \TeX\ to a word
      processor, since the design purposes are different
\item \TeX\ is the natural choice in one of these
      situations
  \begin{itemize}
  \item If we want to typeset a document containing
        lot of Mathematics
  \item If we want our typed document to look beautiful
  \end{itemize}
\end{itemize}
Being a program, \TeX\ offers a high degree of flexibility.
```

The `itemize` environment supports four levels of nesting. The full list of *labels* for the items ('bullets' for the first level, 'dashes' for the second and so on) is as shown below

- The first item in the first level
- the second item in the first level
  - The first item in the second level
  - the second item in the second level
    - The first item in the third level
    - the second item in the third level
      - The first item in the fourth level
      - the second item in the fourth level

Not satisfied with these default labels? How about this one?

- First item of a new list
- Second item

It was produced by the following input:

```
{\renewcommand{\labelitemi}{$\triangleright$}
\begin{itemize}
\item First item of a new list
\item Second item
\end{itemize}}
```

Several things need explanation here. First note that the *first level labels* of the `itemize` environment are produced by the (internal and so invisible to the user) command `\labelitemi` and by default, this is set as `\textbullet` to produce the default 'bullets'. What we do here by issuing the `\renewcommand` is to override this by a choice of our own namely `$\triangleright$` which produces the little triangles in the above list. Why the braces `{` and `}` (did you notice them?) enclosing the whole input? They make the effect of the `\renewcommand` *local* in the sense that this change of labels is only for this specific list. Which means the next time we use an `itemize` environment, the labels revert back to the original 'bullets'. If we want the labels to be changed in the *entire document*, then remove the braces.

What if we want to change the second level labels? No problem, just change the `\labelitemii` command, using a symbol of our choice. The third and fourth level labels are set by the commands (can you guess?) `\labelitemiii` and `\labelitemiv`. Look at the following example.

- The first item in the first level
- the second item in the first level
  - The first item in the second level
  - the second item in the second level
    - The first item in the third level
    - the second item in the third level
      - The first item in the fourth level
      - the second item in the fourth level

Here the labels are chosen from the PostScript ZapfDingbats font. We will have to use the package pifont, by including the line `\usepackage{pifont}` in our document preamble to access them. The source of the above output is

```
\renewcommand{\labelitemi}{\ding{42}}
\renewcommand{\labelitemii}{\ding{43}}
\renewcommand{\labelitemiii}{\ding{44}}
\renewcommand{\labelitemiv}{\ding{45}}
 \begin{itemize}
  \item The first item in the first level
  \item the second item in the first level
   \begin{itemize}
    \item The first item in the second  level
    \item the second item in the second level
     \begin{itemize}
      \item The first item in the third  level
      \item the second item in the third level
       \begin{itemize}
        \item The first item in the fourth  level
        \item the second item in the fourth level
       \end{itemize}
     \end{itemize}
   \end{itemize}
  \end{itemize}}
```

6.4. WHEN ORDER MATTERS

When the *order* of the items in a list is important, we need a list which specifies this order. For example, consider this

> The three basic steps in producing a printed document using LaTeX are as follows
>
> 1. Prepare a source file with the extension `tex`
> 2. Compile it with LaTeX to produce a `dvi` file
> 3. Print the document using a `dvi` driver

Such a numbered list is produced by the `enumerate` environment in LaTeX. The above list was produced by the following source.

```
\begin{enumerate}
\item Prepare a source file with the extension "tex"
\item Compile it with \LaTeX to produce a "dvi" file
```

```
\item Print the document using a "dvi" driver
\end{enumerate}
```

As in the case of `itemize` environment, here also four levels of nesting are supports. The example below shows the labels used for different levels.

1. The first item in the first level
2. the second item in the first level
   (a) The first item in the second level
   (b) the second item in the second level
      i. The first item in the third level
      ii. the second item in the third level
         A. The first item in the fourth level
         B. the second item in the fourth level

How about customizing the labels? Here there is an additional complication in that the labels for items in the same level must follow a sequence (such as 1, 2, 3, ... for the first level, (a), (b), (c), ... for the second and so on, by default). There *is* a method for doing it, but it will take us into somewhat deeper waters. Fortunately, there is a package enumerate by David Carlisle, which makes it easy. So if we want

The three basic steps in producing a printed document using LaTeX are as follows:

Step 1. Prepare a source file with the extension `tex`
Step 2. Compile it with LaTeX to produce a `dvi` file
   i. Use a previewer (such as `xdvi` on X Window System) to view the output
   ii. Edit the source if needed
   iii. Recompile
Step 3. Print the document using a `dvi` driver (such as `dvips`)

just type the input as follows

```
The three basic steps in  producing a printed document
using \LaTeX\ are as follows:
\begin{enumerate}[\hspace{0.5cm}Step 1.]
\item Prepare a source file with the extension "tex"
\item Compile it with \LaTeX to produce a "dvi" file
  \begin{enumerate}[i.]
  \item Use a previewer (such as "xdvi" on
    \textsf{X Window System}) to view the output
```

```
  \item Edit the source if needed
  \item Recompile
  \end{enumerate}
\item Print the document using a "dvi" driver
  (such as "dvips")
\end{enumerate}
```

As you can see, the labels Step 1, Step 2 and Step 3 are produced by the *optional* argument `Step 1` within square brackets immediately following the first `\begin{enumerate}` command and the labels i, ii, iii for the second level enumeration are produced by the optional `[i]` following the second `\begin{enumerate}`. So, what is `\hspace{0.5cm}` doing in the first optional argument? It is to provide an indentation at the left margin of the first level items, which the **enumerate** environment does not produce by default.

We can add further embellishments. For example, if we want the labels in the first level of the above example to be in boldface, just change the optional argument `[\hspace{0.5cm} Step 1]` to `[\hspace{0.5cm}\bfseries Step 1]`. This produces:

The three basic steps in producing a printed document using LaTeX are as follows:

Step 1 Prepare a source file with the extension `tex`

Step 2 Compile it with LaTeX to produce a `dvi` file

(a) Use a previewer (such as `xdvi` on X Window System) to view the output

(b) Edit the source if needed

(c) Recompile

Step 3 Print the document using a `dvi` driver (such as `dvips`)

Some care must be taken when we give options like this. Suppose we want to produce something like this

Addition of numbers satisfies the following conditions:

(A1) It is commutative

(A2) It is associative

(A3) There is an additive identity

(A4) Each number has an additive inverse

If we give the option `[\hspace{1cm}(A1)]` as in

```
Addition of numbers satisfies the following conditions:
```

```
\begin{enumerate}[\hspace{1cm}(A1)]
\item It is commutative
\item It is associative
\item There is an additive identity
\item Each  number has an additive inverse
\end{enumerate}
```

Then we get the (somewhat surprising) output

Addition of numbers satisfies the following conditions:

(11) It is commutative
(22) It is associative
(33) There is an additive identity
(44) Each number has an additive inverse

What happened? In the enumerate package, the option `[A]` signifies that we want the labels to be named in the sequence A, B, C, ..., Z (the upper case Roman alphabet) and the option `[1]` signifies we want them as 1, 2, 3, ...(the Arabic numerals). Other signifiers are `[a]` for lowercase Roman letters, `[I]` for uppercase Roman numerals and `[i]` for lowercase Roman numerals. So, if we use any one of these in the optional argument with some other purpose in mind, then *enclose it in braces*. Thus the correct input to generate the above example is

```
Addition of numbers satisfies the following conditions:
 \begin{enumerate}[\hspace{1cm}({A}1)]
 \item It is commutative
 \item It is associative
 \item There is an additive identity
 \item Each  number has an additive inverse
 \end{enumerate}
```

with braces surrounding the `A`. (The mystery is not over, is it? How come we got 11, 22,... in the above example and not A1, B2,...? Work it out yourselves!)

6.5. DESCRIPTIONS AND DEFINITIONS

There is a third type of list available off-the-shelf in LaTeX which is used in typesetting lists like this

Let us take stock of what we have learnt

TEX A typesetting program

Emacs A text editor and also

a programming environment
a mailer
and a lot else besides

AbiWord A word processor

This is produced by the `description` environment as shown below:

```
Let us take stock of what we have learnt
  \begin{description}
  \item[\TeX] A typesetting program
  \item[Emacs] A text editor and also
    \begin{description}
    \item a programming environment
    \item a mailer
    \item and a lot else besides
    \end{description}
  \item[AbiWord] A word processor
  \end{description}
```

Note that this environment does not produce on its own any labels for the various items, but only produces as labels, whatever we give inside square brackets immediately after each `\item`. By default, the labels are typeset in boldface roman. Also, there is no indentation for the first level. As with the other list environments, these can be changed to suit your taste. For example, suppose we want labels to be typeset in sans-serif roman and also want an indentation even for the first level. The code below will do the trick (remember why we include the whole input within braces?):

```
\renewcommand{\descriptionlabel}[1]{\hspace{1cm}%
    \textsf{#1}}
Let us take stock of what we have learnt
 \begin{description}
  \item[\TeX] A typesetting program
  \item[Emacs] A text editor and also
    \begin{description}
    \item a programming environment
```

```
    \item and a lot else besides
    \end{description}
  \item[AbiWord] A word processor
\end{description}
```

and we get the output

Let us take stock of what we have learnt

TEX A typesetting program

Emacs A text editor and also

a programming environment
and a lot else besides

AbiWord A word processor

Now is perhaps the time to talk about a general feature of all the three list environments we have seen. In any of these, we can override the default labels (if any) produced by the environment by something of our own by including it within square brackets immediately after the `\item`. Thus the input

```
The real number $l$ is the least upper bound of the
set $A$ if it satisfies the following conditions
  \begin{enumerate}
  \item[(1)] $l$ is an upper bound of $A$
  \item[(2)] if $u$ is an upper bound of $A$, then
             $l\le u$
  \end{enumerate}
The second condition is equivalent to
  \begin{enumerate}
  \item[(2)$'$] If $a<l$, then $a$ is not an upper
                bound of $A$.
  \end{enumerate}
```

produces

The real number l is the least upper bound of the set A if it satisfies the following conditions

(1) l is an upper bound of A
(2) if u is an upper bound of A, then $l \le u$

The second condition is equivalent to

(2)′ If $a < l$, then a is not an upper bound of A.

This feature sometimes produces unexpected results. For example, if you type

```
Let's review the notation
 \begin{itemize}
 \item (0,1) is an \emph{open} interval
 \item [0,1] is a \emph{closed} interval
 \end{itemize}
```

you will get

Let's review the notation

- (0,1) is an *open* interval

0,1 is a *closed* interval

What happened? The `0,1` within *square brackets* in the second item is interpreted by LaTeX as the *optional label* for this item. The correct way to typeset this is

```
Let's review the notation
 \begin{itemize}
 \item $(0,1)$ is an \emph{open} interval
 \item $[0,1]$ is a \emph{closed} interval
 \end{itemize}
```

which produces

Let's review the notation

- $(0, 1)$ is an *open* interval
- $[0, 1]$ is a *closed* interval

So, why the dollars around `(0,1)` also? Since `(0,1)` and `[0,1]` are *mathematical entities*, the correct way to typeset them is to include them within braces in the input, even when there is no trouble such as with `\item` as seen above. (By the way, do you notice any difference between (0,1) produced by the input `(0,1)` and $(0, 1)$ produced by `$(0,1)$`?)

In addition to all these tweaks, there is also provision in LaTeX to design your own 'custom' lists. But that is another story.

CHAPTER VII

ROWS AND COLUMNS

THE various *list* environments allows us to format some text into visually distinct *rows*. But sometimes the logical structure of the text may require these rows themselves to be divided into vertically aligned columns. For example, consider the material below typeset using the `\description` environment (doesn't it look familiar?)

Let's take stock of what we've learnt

Abiword A word processor

Emacs A text editor

TEX A typesetting program

A nicer way to typeset this is

Let's take stock of what we've learnt

AbiWord	A word processor
Emacs	A text editor
TEX	A typesetting program

Here the three *rows* of text are visually separated into two *columns* of left aligned text. This was produced by the `tabbing` environment in LATEX.

7.1. KEEPING TABS

7.1.1. BASICS

```
Let's take stock of what we've learnt
\begin{tabbing}
\hspace{1cm}\= \textbf{AbiWord}\quad\= word processor\\[5pt]
```

```
          \> \textbf{Emacs}       \> text editor\\[5pt]
          \> \textbf{\TeX}        \> typesetting program
\end{tabbing}
```

Let's analyze it line by line. In the first line the first tab is put at a distance of 1 cm. from the left margin so that the text following it ('AbiWord' in boldface roman) starts from this point. The second tab is put at a distance of one `\quad` (this is an in-built length specification in TEX roughly equal to one space) after the word 'Abiword' in boldface roman so that the text following it ('A word processor' in ordinary roman face) start from this point. The `\\[5pt]` command signifies the end of the first line and also asks for a vertical space of 5 points between the first and the second lines. In the second line, the first `\>` command makes the text following it ('Emacs' in boldface roman) to start from the first tab (already set in the first line), namely, 1 cm. from the left margin. The second `\>` line makes the text following it ('A text editor' in ordinary roman face) at the second tab already set, namely at a distance 1 cm plus *the length of the word 'AbiWord' in boldface roman* plus a `\quad`. The third line follows suit. The picture below will make this clear.

left margin

tab 1	*tab* 2
↓	↓
AbiWord	word processor
Emacs	text editor
TEX	typesetting program

One should be careful in setting tabs. For example to typeset

TEX	A typesetting program
Emacs	A text editor
AbiWord	A word processor

if you type

```
\begin{tabbing}
  \textbf{\TeX}\quad\= A typesetting program\\[5pt]
  \textbf{Emacs}\quad\> A text editor\\[5pt]
  \textbf{AbiWord}\quad\> A word processor
\end{tabbing}
```

then you end up with the output

TEX A typesetting program
Emacs A text editor
AbiWord A word processor

Do you see what happened? The first line set the first tab (the only tab in this example) at a distance of the length of the word 'TEX' in boldface roman plus a 'quad' from the left margin and the `\>` command in the second line makes the text following to start from this tab, which is right next to the word 'Emacs' in this line. The same thing happens in the third line, which is worse, since the *position* of the tab is at the 'o' of 'AbiWord' and the next word 'A word processor' starts from this point, *and overwrites the previous word*. The correct way to obtain the output we want is to use a *dummy line* to mark the tabs, without actually typesetting that line. This is achieved by the `\kill` command in the `tabbing` environment, as shown below

```
\begin{tabbing}
  \textbf{AbiWord}\quad\= A word processor\kill
  \textbf{\TeX}\quad   \> A typesetting program\\[5pt]
  \textbf{Emacs}\quad  \> A text editor\\[5pt]
  \textbf{AbiWord}\quad\> A word processor
\end{tabbing}
```

New tabs, in addition to the ones already set by the first line (dummy or otherwise), can be set in any subsequent line. Thus the output

TEX : A typesetting program
Emacs : A text editor
a programming environment
a mail reader
and a lot more besides
AbiWord : A word processor

is obtained from the source

```
\begin{tabbing}
```

```
\textbf{AbiWord}\quad\= : \= A word processor\kill\\
\textbf{\TeX}\quad   \> : \> A typesetting program\\[5pt]
\textbf{Emacs}\quad  \> : \> A text editor\\[5pt]
        \>  \> \quad\= a programming environment\\[5pt]

        \>  \> \> a mail reader\\[5pt]
        \>  \> \> and a lot more besides\\[5pt]
\textbf{AbiWord}\quad\> : \> A word processor
\end{tabbing}
```

Here the first line sets two tabs and the fourth line sets a third tab *after* these two. All the three tabs can then be used in the subsequent lines. New tab positions which *change* the ones set up by the first line, can also be introduced in any line by the `\=` command. Thus we can produce

Program : TEX
Author : Donald Knuth
Manuals :

Title	Author	Publisher
The TEXBook	Donald Knuth	Addison-Wesley
The Advanced TEX Book	David Salomon	Springer-Verlag

by the input

```
\begin{tabbing}
 Program\quad \= : \= \TeX\\[5pt]
 Author       \> : \> Donald Knuth\\[5pt]
 Manuals      \> :\\
 \quad\= The Advanced \TeX\ Book\quad
      \= David Salomon\quad
      \= Springer-Verlag\kill\\
 \> \textsf{Title}
     \> \textsf{Author}
     \> \textsf{Publisher}\\[8pt]
 \> The \TeX Book
     \> Donald Knuth
     \> Addison-Wesley\\[5pt]
 \> The Advanced \TeX\ Book
     \> David Salomon
     \> Springer-Verlag
\end{tabbing}
```

Here the first line sets two tabs and the next two lines use these tabs. The third line sets three new tabs which *replace* the original tab positions. The next three lines use these new tab positions.

7.1.2. PUSHING AND POPPING

What if you change the tab positions and then want the original settings back? Here's where the command pair

```
\pushtabs ... \poptabs
```

is useful. Thus to typeset

Program : TEX
Author : Donald Knuth
Manuals :

Title	Author	Publisher
The TEXBook	Donald Knuth	Addison-Wesley
The Advanced TEX Book	David Salomon	Springer-Verlag

Tutorial : http://tug.org.in/tutorial

we type

```
\begin{tabbing}
 Program\quad \= : \= \TeX\\[5pt]
 Author       \> : \> Donald Knuth\\[5pt]
 Manuals      \> : \\
 \pushtabs
 \quad\= The Advanced \TeX\ Book \quad
      \= David Salomon \quad
      \= Springer-Verlag\kill\\
 \>\textsf{Title}
      \> \textsf{Author}
      \> \textsf{Publisher}\\[8pt]
 \>The \TeX Book
      \> Donald Knuth
      \> Addison-Wesley\\[5pt]
 \>The Advanced \TeX\ Book
      \> David Salomon
      \> Springer-Verlag\\[8pt]
 \poptabs
```

```
  Tutorial
      \> :
      \> "http://tug.org.in/tutorial"
\end{tabbing}
```

Here the first three lines follow a tabbing scheme, the next three lines follow another tabbing scheme and the last line reverts back to the original scheme. Here the `\pushtabs` command stores the current tabbing scheme and removes it so that a new tabbing scheme can be set up; and the `\poptabs` commands reactivates the original scheme. These commands can be nested.

7.1.3. MORE COMMANDS

There are some more useful commands available in the `tabbing` environment. The `\+` command *given at the end of a line* makes *every subsequent line* start at the first tab; with `\+\+` at the end of a line, all subsequent lines start at the second tab and so on. The effect of each `\+` can be neutralized by one `\-` command at the end of a line. The command `\<` *at the beginning of a line* neutralizes the effect of one `\+` command for that particular line.

The command `\`` (left quote) puts the text following flush right against the right margin. Naturally we cannot use a `\=` or `\>` after this in a line.

Another interesting command is `\'` (right quote). Within the `tabbing` environment an input of the form `left_text\'right_text` puts the `right_text` at the current tab and the `left_text` just before this tab with a bit of spacing (preassigned by the parameter `\tabbingsep`).

The example below illustrates all the tabbing commands we've discussed

```
\begin{tabbing}
 Row 1 Column 1\hspace{2cm}
    \= Row 1 Column 2\\[5pt]
    \> Row 2 Column 2\hspace{.5cm}
      \= Row 2 Column 3\+\+\\[5pt]
        Row 3 Column 3\-\\[5pt]
 Row 4 Column 2
      \> Row 4 Column 3\\[5pt]
   \< Row 5 Column 1
      \> Row 5 Column 2
```

```
      \>Row 5 Column 3\\[5pt]
 Row 6 Column 2
      \> Row 6 Column 3\-\\[5pt]
 Row 7 Column 1
      \> Row 7 Column 2
      \> Row 7 Column 3\\[5pt]
 Row 8 Column 1 \`Right\\[5pt]
 Row 9 Column 1
 \> and\'Row 9 Column 2\\[5pt]
 \pushtabs
 \quad\= Row 10 New Column 1\hspace{1.5cm}
      \= Row 10 New Column 2\\[5pt]
      \> Row 11 New Column 2
      \> Row 11 New Column 2\\[5pt]
 \poptabs
 Row 12 Old Column 1
      \> Row 12 Old Col2
      \>Row 12 Old Col 3
\end{tabbing}
```

It produces the output in Figure 7.1.

Row 1 Column 1		Row 1 Column 2	
		Row 2 Column 2	Row 2 Column 3
			Row 3 Column3
		Row 4 Column 2	Row 4 Column 3
Row 5 Column 1		Row 5 Column 2	Row 5 Column 3
		Row 6 Column 2	Row 6 Column 3
Row 7 Column 1		Row 7 Column 2	Row 7 Column 3
Row 8 Column 1			Right
Row 9 Column 1	and	Row 9 Column 2	
Row 10 New Column 1		Row 10 New Column 2	
Row 11 New Column 2		Row 11 New Column 2	
Row 12 Old Column 1		Row 12 Old Col 2	Row 12 Old Col 3

Figure 7.1: Enhanced tabbing.

Recall that the commands `\=`. `\`` and `\'` are used for various accents outside the `tabbing` environment. If these are needed

within the tabbing environment, they can be produced with the commands \a=. \a` or \a' commands.

One final word. You might've noted in the examples above that we give a sort of 'formatting' to the sources also. This is not really necessary from the point of view of LaTeX since the output of the last example is the same even if we input

```
 \begin{tabbing}
 Row 1 Column 1\hspace{2cm}\=Row 1 Column 2\\[5pt]
 \>Row 2 Column 2\hspace{.5cm}\=Row 2 Column 3\+\+\\[5pt]
 Row 3 Column3\-\\[5pt]
 Row 4 Column 2\>Row 4 Column 3\\[5pt]
 \<Row 5 Column\>Row 5 Column 2\>Row 5 Column 3\\[5pt]
 Row 6 Column 2\>Row 6 Column 3\-\\[5pt]
 Row 7 Column 1\>Row 7 Column 2\>Row 7 Column 3\\[5pt]
 Row 8 Column 1\`\textbf{Flush right}\\[5pt]
 Row 9 Column 1\>and\'Row 9 Column 2\\[5pt]
\pushtabs
 Row 10 New Column 1\hspace{1.5cm}\=Row 10 New Column 2\\[5pt]
 Row 11 New Column 2\>Row 11 New Column 2\\[5pt]
\poptabs
 Row 12 Old Column 1\>Row 12 Old Col 2\>Row 12 Old Col 3
\end{tabbing}
```

LaTeX can make sense out of this, but we humans cannot. And such a jumble makes editing a hopeless task. The moral? Keep the source (humanly) readable.

7.2. TABLES

Another way to format text into columns and rows is to use the tabular environment. Let's see it in action by means of examples given in Figure 7.2 and Figure 7.3.

The \begin{center} ... \end{center} commands centralize the table. The table itself is produced by the

```
\begin{tabular} ...\end{tabular}
```

commands. The {lr} specification immediately after the \begin{tabular} indicates there are two *columns* in the table with the entries in the first column aligned on the *left* and the entries in the second column aligned on the *right*. The entries in each column are separated by the & symbol and the termination of each row is signalled by the \\ symbol. The \\[5pt] after the

Planet	Diameter(km)
Mercury	4878
Venus	12104
Earth	12756
Mars	6794
Jupiter	142984
Saturn	120536
Uranus	51118
Neptune	49532
Pluto	2274

Figure 7.2: Format text into columns

```
\begin{center}
  \begin{tabular}{lr}
  Planet  & Diameter(km)\\[5pt]
    Mercury & 4878\\
    Venus   & 12104\\
    Earth   & 12756\\
    Mars    & 6794\\
    Jupiter & 142984\\
    Saturn  & 120536\\
    Uranus  & 51118\\
    Neptune & 49532\\
    Pluto   & 2274
  \end{tabular}
\end{center}
```

Figure 7.3: Source of Figure 7.2.

first row specifies as usual, an additional vertical space of 5 points after this row in the output.

In addition to the column specifiers `l` and `r` we also have a specifier `c` which makes the entries in the corresponding column *centrally* aligned. For example the input in Figure 7.4 produces the output in Figure 7.5.

```
\begin{center}
  \begin{tabular}{cr}
  Planet  & Diameter(km)\\[5pt]
    Mercury & 4878\\
    Venus   & 12104\\
    Earth   & 12756\\
    Mars    & 6794\\
    Jupiter & 142984\\
    Saturn  & 120536\\
    Uranus  & 51118\\
    Neptune & 49532\\
    Pluto   & 2274
  \end{tabular}
\end{center}
```

Figure 7.4: Source of another tabular.

Planet	Diameter(km)
Mercury	4878
Venus	12104
Earth	12756
Mars	6794
Jupiter	142984
Saturn	120536
Uranus	51118
Neptune	49532
Pluto	2274

Figure 7.5: The output of source in Figure 7.4.

There's yet another column specifier `p` which allows us to set column entries in a *box* of specified width (technically a "parbox"—see Chapter **??**). Suppose you want the following

Planet	Features
Mercury	Lunar like crust, crustal faulting, small magnetic fields.
Venus	Shrouded in clouds, undulating surface with highlands, plains, lowlands and craters.
Earth	Ocens of water filling lowlands between continents, unique in supporting life, magnetic field.
Mars	Cratered uplands, lowland plains, volcanic regions.
Jupiter	Covered by clouds, dark ring of dust, magnetic field.
Saturn	Several cloud layers, magnetic field, thousands of rings.
Uranus	Layers of cloud and mist, magentic field, some rings.
Neptune	Unable to detect from earth.
Pluto	Unable to detect from earth

It is produced from the input

```
\begin{center}
  \begin{tabular}{lp{.8\linewidth}}
   Planet  & Features\\[5pt]
   Mercury & Lunar like crust, crustal faulting, small
             magnetic fields.\\
   Venus   & Shrouded in clouds, undulating surface
             with highlands, plains, lowlands and craters.\\
   Earth   & Ocens of water filling lowlands between
             continents, unique in supporting life,
             magnetic field.\\
   Mars    & Cratered uplands, lowland plains,
             volcanic regions.\\
   Jupiter & Covered by clouds, dark ring of dust,
             magnetic field.\\
   Saturn  & Several cloud layers, magnetic field,
             thousands of rings.\\
   Uranus  & Layers of cloud and mist, magentic
             field, some rings.\\
   Neptune & Unable to detect from earth.\\
   Pluto   & Unable to detect from earth
  \end{tabular}
\end{center}
```

Here the specification `p{6cm}` shows that in a "paragraph box" of width 6 cm. In a `p`-type column, if a `\raggedright` or `\centering` is given, then we can induce explicit line breaks *within that column* by the `\\` command. If such commands are used in the last column of a row, then the command `\tabularnewline` should be used to terminate that row as in this example:

```
\begin{center}
  \begin{tabular}{lp{6cm}}
    Planet  & Features\tabularnewline[8pt]
    Mercury & \raggedright Lunar like crust\\
              Crustal faulting\\
              Small magnetic fiels\tabularnewline[3pt]
    Venus   & \raggedright Shrouded in clouds\\
              Undulating surface\tabularnewline[3pt]
    Earth   & \raggedright Ocens of water\\
              Unique in supporting life\\
              Magnetic field\tabularnewline[3pt]
    Mars    & \raggedright Cratered uplands\\
              Lowland plains\\
              Volcanic regions\tabularnewline[3pt]
    Jupiter & \raggedright Covered by clouds\\
              Dark ring of dust\\
              Magnetic field\tabularnewline[3pt]
    Saturn  & \raggedright Several cloud layers
              Magnetic field\\
              Thousands of rings\tabularnewline[3pt]
    Uranus  & \raggedright Layers of cloud and mist\\
              Magentic field\\
              Some rings\tabularnewline[3pt]
    Neptune & Unable to detect
              from earth\tabularnewline[3pt]
    Pluto   & Unable to detect
              from earth\tabularnewline[3pt]
  \end{tabular}
\end{center}
```

This produces the output as in Table 7.1. Note that the last two lines don't need a `\raggedright` command, since there are no explicit linebreaks in them.

A table usually contains horizontal and vertical lines separating the rows and columns. These can also be produced in the `tabular` environment. For example, the first table we saw above can be typeset as

Planet	Diameter(km)
Mercury	4878
Venus	12104
Earth	12756
Mars	6794
Jupiter	142984
Saturn	120536

by the input

```
\begin{center}
  \begin{tabular}{|l|r|}
    \hline
    Planet  & Diameter(km)\\
    \hline
    Mercury & 4878\\
    ..............
    Pluto   & 2274\\
    \hline
  \end{tabular}
\end{center}
```

Do you see what produced the vertical and horizontal lines? Instead of the specification `{lr}` used earlier, we now have `{|l|r|}` The character `|` causes a vertical line to be drawn at the specified location, running down the entire height of the table. (Two `|`'s in succession produce a double vertical lines.) An `\hline` command after a row draws a horizontal line after that row, running along the entire width of the table. (Again, two `\hline`'s in succession produce a double horizontal line.) Note also that because of the last `\hline` , we should give a line termination command `\\` at the end of the last row also.

Now suppose we want to produce something like this

Planet	Distance from sun (km)	
	Maximum	Minimum
Mercury	69400000	46800000
Venus	109000000	107600000
Earth	152600000	147400000
Mars	249200000	207300000
Jupiter	817400000	741600000
Saturn	1512000000	1346000000

Here, there are three columns and the entry `Distance from the sun (km)` is to span the the last two columns below it. The command `\multicolumn` does the trick as shown below

```
\begin{center}
  \begin{tabular}{lrr}
    Planet  & \multicolumn{2}{c}{Distance from sun (km)}\\
            & Maximum    & Minimum\\
    Mercury & 69400000   & 46800000\\
    Venus   & 109000000  & 107600000\\
    Earth   & 152600000  & 147400000\\
    Mars    & 249200000  & 207300000\\
    Jupiter & 817400000  & 741600000\\
    Saturn  & 1512000000 & 1346000000\\
  \end{tabular}
\end{center}
```

The entry `\multicolumn{2}{c}{Distance from sun (km)}` indicates that the *item* within the last set of braces is to span *two* columns as specified by the `2` within the first set of braces. The entry `c` within the second set of braces indicates that this text is to be *centered* within the column. Thus the general form of the command is

`\multicolumn{`*num*`}{`*pos*`}{`*item*`}`

where *num* is the number of columns to be spanned, *pos* is the position of the item within the column and *item* is the text of the item. Note also that the input for the second row *starts* with an `&` character. This is because there is no entry in the first column of the second row.

Now what if you want

Planet	Distance from sun (km)	
	Maximum	Minimum
Mercury	69400000	46800000
Venus	109000000	107600000
Earth	152600000	147400000
Mars	249200000	207300000
Jupiter	817400000	741600000
Saturn	1512000000	1346000000
Uranus	3011000000	2740000000
Neptune	4543000000	4466000000
Pluto	7346000000	4461000000

Here the first few lines and the last lines of the input are as below (the other lines are the same as in the previous example).

```
\begin{center}
  \begin{tabular}{|l|r|r|}
    \hline
    Planet & \multicolumn{2}{c|}{Distance from sun (km)}\\
    \cline{2-3}
           & Maximum   & Minimum\\
    \hline

    ..............................

    \hline
  \end{tabular}
\end{center}
```

Table 7.1: The usage of `tabularnewline`.

Planet	Features
Mercury	Lunar like crust
	Crustal faulting
	Small magnetic fiels
Venus	Shrouded in clouds
	Undulating surface
Earth	Ocens of water
	Unique in supporting life
	Magnetic field
Mars	Cratered uplands
	Lowland plains
	Volcanic regions
Jupiter	Covered by clouds
	Dark ring of dust
	Magnetic field
Saturn	Several cloud layers
	Magnetic field
	Thousands of rings
Uranus	Layers of cloud and mist
	Magentic field
	Some rings
Neptune	Unable to detect from earth

Note that the position specifier in the `\multicolumn` command here is `c|`. This has to do with the way the environment splits the column specification into various columns. For example, the specification `|l|r|r|` in this exaple is split into `|l|`, `r|` and `r|` and the `\multicolumn{2}` command resets the last two columns. In particular, the final `|` gets reset and we'll have to explicitly supply it in the position specification of the `\multicolumn` command as `c|`.

Note also the command `\cline{2-3}` after the first row. This draws a horizontal line from the second to the third column. In general the command `\cline{`i`-`j`}` draws a horizontal line from the i^{th} column to the j^{th} column.

Another feature of the `\multicolumn` command is that with `\multicolumn{1}` we can override the position specification of any column set at the beginning of the environment. For example, consider the input below

```
\begin{center}
  \begin{tabular}{|l|r|r|}
    \hline
           & \multicolumn{2}{p{3.5cm}|}%
               {\centering Distance from sun \\ (million km)}\\
    \cline{2-3}
    \multicolumn{1}{|c|}{Planet}
       & \multicolumn{1}{c|}{Maximum}& \multicolumn{1}{c|}{Minimum}\\
    \hline
    Mercury & 69.4   & 46.8\\
    Venus   & 109.0  & 107.6\\
    Earth   & 152.6  & 147.4\\
    Mars    & 249.2  & 207.3\\
    Jupiter & 817.4  & 741.6\\
    Saturn  & 1512.0 & 1346.0\\
    Uranus  & 3011.0 & 2740.0\\
    Neptune & 4543.0 & 4466.0\\
    Pluto   & 7346.0 & 4461.0\\
    \hline
  \end{tabular}
\end{center}
```

It produces the output as in Table 7.2.

Note that even though `\centering` is used in the last column of the first row, no `\tabularnewline` is required to termi-

Table 7.2: Usage of `multicolumn`.

Planet	Distance from sun (million km)	
	Maximum	Minimum
Mercury	69.4	46.8
Venus	109.0	107.6
Earth	152.6	147.4
Mars	249.2	207.3
Jupiter	817.4	741.6
Saturn	1512.0	1346.0
Uranus	3011.0	2740.0
Neptune	4543.0	4466.0
Pluto	7346.0	4461.0

nate this row, since the scope of the `\centering` is limited by the `\multicolumn`.

By the way, do you feel that the tables we've been produced look a bit cramped? A bit crowded vertically? Well, you can create a bit more room between *rows* by redefining the value of `\arraystretch`. By default, it's value is 1 and if you set it to a number k, then the interrow space is increased k-fold. Thus the input of the last example with the command

```
\renewcommand{\arraystretch}{1.2}
```

after the `\begin{center}` produces output as in Table 7.3.

Next let's see how we produce a table like the one in Table 7.4. Here we want all the dashes in the second column to be vertically aligned, so that we must set them in a separate column; but then there should be no space between the numbers and the dashes connecting them. In such cases we can use the `@` command in the column specification as below

```
\begin{center}
  \begin{tabular}{|c|r@{--}l|}
    \hline
    Height & \multicolumn{2}{c|}{Ideal weight}\\
    (cm)   & \multicolumn{2}{c|}{(kg)}\\
    \hline
    155 & 53.5 & 64\\
```

```
    160 & 56   & 67\\
    ..............
    190 & 78   & 92.5\\
    \hline
  \end{tabular}
\end{center}
```

Table 7.3: Usage of `arraystretch`.

Planet	Distance from sun (million km)	
	Maximum	Minimum
Mercury	69.4	46.8
Venus	109.0	107.6
Earth	152.6	147.4
Mars	249.2	207.3
Jupiter	817.4	741.6
Pluto	7346.0	4461.0

Here the specification `r@{--}l` indicates that there should be a right aligned column and a left aligned column with a – in between each pair of entries in these columns *without the intercolumn space* the `tabular` environment leaves by default between every pair of columns. Note that this incidently saves us the trouble of repeatedly typing `--`. You can also add some space producing commands within the braces after the `@` command to produce that much space between the columns on either side of it.

Table 7.4: Usage of `array` features.

Height (cm)	Ideal weight (kg)
155	53.5–64
160	56–67
190	78–92.5

7.2.1. ENHANCEMENTS TO THE tabular

There are many packages which provide further facilities in forming tables. We'll discuss a couple of such packages here.

7.2.2. THE ARRAY PACKAGE

Look at the tables in Table 7.5. The one on the right looks nicer, doesn't it? It was produced using the column specifier `m` available in the array package. To produce this table, we must first load the array package by the usual `\usepackage{array}` in the preamble and then type

Table 7.5: Enhancements to the tabular

.

Planet	Mean distance from sun (km)	Planet	Mean distance from sun (km)
Mercury	58100000	Mercury	58100000
Venus	108300000	Venus	108300000
Earth	150000000	Earth	150000000
Mars	228250000	Mars	228250000
Jupiter	779500000	Jupiter	779500000
Saturn	1429000000	Saturn	1429000000
Uranus	2439000000	Uranus	2439000000
Neptune	4504500000	Neptune	4504500000
Pluto	5903500000	Pluto	5903500000

```
\begin{tabular}{|l|r|}
 \hline
 \multicolumn{1}{|m{1.5cm}|}{\centering Planet}
           &\multicolumn{1}{m{2.3cm}|}%
             {\centering Mean distance from sun \\ (km)}\\
  \hline
   Mercury & 58100000\\
   ..................
   Pluto   & 5903500000\\
   \hline
 \end{tabular}
```

The `m{`*wd*`}` specifier produces a column of width *wd* just like the `p` specifier, but with the text aligned vertically in the middle

unlike the p specifier which aligns the text with the topline. (The table on the left, incidently, was produced by the same input as above but with p instead of m).

Another interesting feature of the array package is the >{*decl*} command which can be used before a column specifier. It inserts *decl* directly in front of the column. For example look at the input below

```
\begin{center}
  \begin{tabular}{|>{\bfseries}l|r|}
    \hline
    \multicolumn{1}{|m{1.5cm}|}{\centering Planet}
          &\multicolumn{1}{m{2.3cm}|}%
            {\centering Mean distance from sun \\ (km)}\\
    \hline
    Mercury & 58100000\\
    Venus   & 108300000\\
    Earth   & 150000000\\
    Mars    & 228250000\\
    Jupiter & 779500000\\
    Saturn  & 1429000000\\
    Uranus  & 2439000000\\
    Neptune & 4504500000\\
    Pluto   & 5903500000\\
    \hline
  \end{tabular}
\end{center}
```

which produces the output

Planet	Mean distance from sun (km)
Mercury	58100000
Venus	108300000
Earth	150000000
Mars	228250000
Jupiter	779500000
Saturn	1429000000
Uranus	2439000000
Neptune	4504500000
Pluto	5903500000

The array package also has a ! command which works just like the @ command, but whch does not suppress the intercolumn space.

7.2.3. THE MULTIROW PACKAGE

Look again at the table in page 84. Wouldn't it be nice if the entry "Planet" in the first column is vertically aligned with the center of the two rows in the next column as below?

Planet	Distance from sun (million km)	
	Maximum	Minimum
Mercury	69.4	46.8
Venus	109.0	107.6
Earth	152.6	147.4
Mars	249.2	207.3
Jupiter	817.4	741.6
Saturn	1512.0	1346.0
Uranus	3011.0	2740.0
Neptune	4543.0	4466.0
Pluto	7346.0	4461.0

The package multirow is what we need to do this painlessly. It has a command

\multirow{*num*}{*wd*}{*item*}

where *num* is the number of rows to be spanned, *wd* is the width of this column and *item* is the text of the item in this column. This can be used as in the following example

```
\begin{center}
  \begin{tabular}{|l|r|r|}
  \hline
  \multirow{3}{1.5cm}{Planet}
          & \multicolumn{2}{p{3.5cm}|}%
               {\centering Distance from sun \\ (million km)}\\
  \cline{2-3}
          & \multicolumn{1}{c|}{Maximum}
                   & \multicolumn{1}{c|}{Minimum}\\
  \hline
  Mercury & 69.4   & 46.8\\
  Venus   & 109.0  & 107.6\\
```

```
    Earth   & 152.6  & 147.4\\
    Mars    & 249.2  & 207.3\\
    Jupiter & 817.4  & 741.6\\
    Saturn  & 1512.0 & 1346.0\\
    Uranus  & 3011.0 & 2740.0\\
    Neptune & 4543.0 & 4466.0\\
    Pluto   & 7346.0 & 4461.0\\
    \hline
  \end{tabular}
\end{center}
```

But this code does not produce the table above, but only

Planet	Distance from sun (million km)	
	Maximum	Minimum
Mercury	69.4	46.8
Venus	109.0	107.6
Earth	152.6	147.4
Mars	249.2	207.3
Jupiter	817.4	741.6
Saturn	1512.0	1346.0
Uranus	3011.0	2740.0
Neptune	4543.0	4466.0
Pluto	7346.0	4461.0

The trouble is that though the entry "Planet" is vertically centered in its column, it is not horizontally centered. The horizontal alignment is controlled by the command `\multirowsetup` and this is by default `\raggedright`. So all that is needed to get the beautiful table seen at the beginning of this section is to add the line

```
\renewcommand{\multirowsetup}{\centering}
```

at the beginning of the code above.

7.2.4. tabbing VS. tabular

Let's take a quick look at the pros and cons of the `tabbing` and `tabular` environments.

- The `tabbing` environment can be typeset only as a separate paragraph, while the `tabular` environment can be placed anywhere in text, even inside Mathematics.

- The `tabbing` environment can span multiple pages, but the `tabular` environment cannot.
- `tabbing` environments cannot be nested, while `tabular` environments can be nested to any number of levels.

7.2.5. MULTIPAGE TABLES — THE PACKAGE LONGTABLE

As we have noted, we cannot create table spanning more than one page using the `tabular` environment. But the package longtable by David Carlisle can do this and it has quite a few other tricks also. To use this package, load it as usual with the command `\usepackage{longtable}` in the preamble and then to produce a no-frills "longtable" just use the commands

```
\begin{longtable} ... \end{longtable}
```

instead of the `\begin{tabular} ... \end{tabular}` commands. We can use footnotes and the `\newpage` commands inside the `longtable` environment. If the package array is also loaded, its extra features can be used.

Apart from this, this package has provisions to specify *at the start of the input* the following items

- the rows that should appear at the *top of the table*; the input for these to be terminated by `\endfirsthead`
- the rows that should appear in *every page after the first*, such input terminated by `\endhead`
- those at the *bottom of every page*, the input terminated by `\endfoot`
- those rows at the *very end* of the table, terminated by `\endlastfoot`

These are illustrated in the (long!) table below.

Science and Technology in the Twentieth Century

Year	Event
1900	Max Planck proposes quantum theory
	Publication of Sigmund Freud's *The Interpretation of Dreams*
1901	Discovery of principal blood groups

continued on the next page

Science and Technology in the Twentieth Century
(*contd.*)

Year	Event
	Guglielmo Marconi transmits wireless signals across the atlantic
1903	Wright brothers make their first flight
1905	Albert Einstein presents Special Theory of Relativity
1911	Ernest Rutherford proposes theory of atomic structure
1912	Victor Hess discovers cosmic rays
1916	Albert Einstein presents general Theory of Relativity
1920	Radio broadcasting begins
1926	John Logie Baird demonstrates television
1928	Alexander Fleming discovers penicillin
1933	Discovery of polythene
1934	Discovery of nuclear fission
1938	Discovery of nylon
1940	Plutonium obtained by bombardment of uranium
1942	Construction of first nuclear reactor
1946	Construction of first electronic digital computer
1947	First supersonic flight
	Invention of the transistor
1951	Nuclear power stations introduced
1953	James Watson and Francis Crick show DNA molecule structure
1956	Contraceptive pill introduced
1957	Launch of the first space satellite (*Sputnik 1*)
1959	First photograph of the dark side of the moon (*Luna 3*)
...	.....
...	.....
...	.....
...	.....
...	.....
...	.....
...	.....
...	.....

continued on the next page

Science and Technology in the Twentieth Century
(*contd.*)

Year	Event
...	.....
...	.....
...	.....
...	.....
1961	Yuri Gagarin becomes first man in space (*Vostok 1*)
1966	First lunar soft landing (*Luna 9*)
1967	Discovery of pulsars
1968	First manned lunar orbit (*Apollo 8*)
1969	First man on moon (Neil Armstrong)
1972	Pocket calculator introduced
1974	First 'test-tube babies'
1977	Launch of *Voyager* missions to outer spce
1983	IBM personal computer launched
1986	Hailey's comet intercepted
1997	Cloning of "Dolly" the sheep
2000	Decoding of 90% of human genome completed

Source : *The Cambrige Factfinder*

Part of the code to produce this is given below.

```
\renewcommand{\arraystretch}{1.2}
\begin{longtable}{|c|l|}
  \multicolumn{2}{c}%
    {\textbf{Science and Technology in the
      Twentieth Century}}\\[5pt]
  \hline
  \multicolumn{1}{|c|}{\sffamily Year}
  &\multicolumn{1}{|c|}{\sffamily Event}\\
  \hline
  \endfirsthead
   \multicolumn{2}{c}%
      {\textbf{Science and Technology in the
       Twentieth Century}
       (\textit{continued})}\\[5pt]
  \hline
  \multicolumn{1}{|c|}{\sffamily Year}
  &\multicolumn{1}{|c|}{\sffamily Event}\\
  \hline
```

```
  \endhead
  \hline
  \multicolumn{2}{r}{\small\itshape
     continued on the next page}\\
  \endfoot
  \hline
  \multicolumn{2}{r}{\small Source\,:\,\itshape
     The Cambrige Factfinder}
  \endlastfoot
  1900 & Max Planck proposes quantum theory\\
  ..........................................
  2000 & Decoding of 90\% of human genome completed\\
  \hline
\end{longtable}
```

7.2.6. AND THAT'S NOT ALL!

There are many more packages which help to produce tables of various requirements. Be sure to check out the pakages tabularx, delarray, dcolumn and hhline. The command `texdoc` followed by the name of the package without extension in your terminal will open up the PDF documentation of the package if it is available in your system.

```
texdoc tabularx
```

CHAPTER VIII

TYPESETTING MATHEMATICS

DONALD KNUTH created TEX primarily to typeset Mathematics beautifully. LATEX includes all the capabilities of TEX in Mathematics typesetting, sometimes with easier user interfaces. Then there are packages like amsmath which enhance and refine these interfaces.

8.1. THE BASICS

A mathematical expression occurring in running text (called *in-text* math) is produced by enclosing it between dollar signs. Thus to produce

> The equation representing a straight line in the Cartesian plane is of the form $ax + by + c = 0$, where a, b, c are constants.

we type

```
The equation representing  a straight line in the
Cartesian plane is of the form $ax+by+c=0$, where
$a$, $b$, $c$ are constants.
```

Some comments are in order. First note that the text within dollars is typeset in *italic* (actually math italic). Again, even though we did not leave any spaces within `ax+by+c=0`, TEX leaves spaces on either side of the addition signs and the equality sign. On the other hand, even if we type `$ax + by + c = 0$`, the output would be the same: $ax + by + c = 0$. The moral? TEX *has its own spacing rules in math mode*.

To see another instance of this, change the last part of the code above to read

```
... where $a, b, c$ are constants.
```

Saves some typing, does not it? But look at the output.

> The equation representing a straight line in the Cartesian plane is of the form $ax + by + c = 0$, where a, b, c are constants.

Do you see the difference? There are no spaces after the commas, though we had such spaces in the output. So TEX swallows spaces in math mode (you can not save dollars that way!).

Incidentally, dollar signs are TEX way of distinguishing Mathematical text. LATEX has other ways also of doing it, using `\( ... \)` or `\begin{math} ... \end{math}`. Thus either of the inputs shown below also produces the same output as above.

```
The equation representing a straight line in the Cartesian
plane is of the form \(ax+by+c=0\), where \(a\), \(b\),
\(c\) are constants.
```

```
The equation representing a straight line in the Cartesian
plane is of the form \begin{math}ax+by+c=0\end{math}, where
\begin{math} a \end{math}, \begin{math} b \end{math},
\begin{math} c \end{math} are constants.
```

Now suppose we want to *display* the equation in the above output as in

> The equation representing a straight line in the Cartesian plane is of the form
>
> $$ax + by + c = 0$$
>
> where a, b, c are constants.

This can be done by changing the input as follows:

```
The equation representing a straight line in the Cartesian
plane is of the form
$$
  ax+by+c=0
$$
where $a$, $b$, $c$ are constants.
```

Again `$$ ... $$` is the TEX way of producing displayed math. LATEX has the constructs `\[ ... \]` or `\begin{displaymath} ... \end{displaymath}` also to do this.

8.1.1. SUPERSCRIPTS AND SUBSCRIPTS

Look at the text below

> In the seventeenth century, Fermat conjectured that if $n > 2$, then there are no integers x, y, z for which
>
> $$x^n + y^n = z^n.$$
>
> This was proved in 1994 by Andrew Wiles.

This is produced by the input

```
In the seventeenth century, Fermat conjectured that if $n>2$,
then there are no integers $x$, $y$, $z$ for which
$$
x^n+y^n=z^n.
$$
This was proved in 1994 by Andrew Wiles.
```

This shows that superscripts (mathematicians call them exponents) are produced by the ^ symbol. If the superscript is more than one character long, we must be careful to *group* these characters properly. Thus to produce

> It is easily seen that $(x^m)^n = x^{mn}$.

we must type

```
It is easily seen that $(x^m)^n=x^{mn}$.
```

Instead of `$x^{mn}$`, if we type `$x^mn$` we end up with $x^m n$ instead of the intended x^{mn} in the output.

We can have superscripts of superscripts (and mathematicians do need them). For example,

> Numbers of the form $2^{2^n} + 1$, where n is a natural number, are called Fermat numbers.

is produced by

```
Numbers of the form $2^{2^n}+1$, where $n$ is a natural
number, are called Fermat numbers.
```

Note the grouping of superscripts. (What happens if you type `$2^2^n+1$` or `${2^2}^n$`?)

Now let us see how subscripts (mathematicians call them subscripts) are produced. To get

The sequence (x_n) defined by

$$x_1 = 1, \quad x_2 = 1, \quad x_n = x_{n-1} + x_{n-2} \ \ (n > 2)$$

is called the Fibonacci sequence.

we must type

```
The sequence $(x_n)$ defined by
$$
  x_1=1,\quad x_2=1,\quad x_n=x_{n-1}+x_{n-2}\;\;(n>2)
$$
is called the Fibonacci sequence.
```

Thus subscripts are produced by the _ character. Note how we insert spaces by the `\quad` command. (The command `\;` in *math mode* produces what is known as a "thickspace".) Subscripts of subscripts can be produced as in the case of superscripts (with appropriate grouping).

We can also have superscripts and subscripts together. Thus

If the sequence (x_n) converges to a, then the sequence (x_n^2) converges to a^2

is produced by

```
If the sequence $(x_n)$ converges to $a$, then the sequence
$(x_n^2)$ converges to $a^2$
```

Again, we must be careful about the grouping (or the lack of it) when typesetting superscripts and subscripts together. The following inputs and the corresponding outputs make the point.

```
$$
 x_m^n\qquad x^n_m\qquad {x_m}^n\qquad {x^n}_m
$$
```

$$x_m^n \qquad x_m^n \qquad {x_m}^n \qquad {x^n}_m$$

(This has to do with the way TEX works, producing "boxes" to fit the output characters. The box for x_m^n is like $\boxed{x_m^n}$ while the box for ${x_m}^n$ is $\boxed{{x_m}^n}$.

8.1.2. ROOTS

Square roots are produced by the `\sqrt` argument. Thus `$\sqrt{2}$` produces $\sqrt{2}$. This command has an optional argument to produce other roots. Thus

Which is greater $\sqrt[4]{5}$ or $\sqrt[5]{4}$?

is produced by

```
Which is greater $\sqrt[4]{5}$ or $\sqrt[5]{4}$?
```

The horizontal line above the root (called *vinculum* by mathematicians of yore) elongates to accommodate the enclosed text. For example, `$\sqrt{x+y}$` produces $\sqrt{x+y}$. Also, you can produce nested roots as in

The sequence

$$2\sqrt{2}\,,\quad 2^2\sqrt{2-\sqrt{2}}\,,\quad 2^3\sqrt{2-\sqrt{2+\sqrt{2}}}\,,\quad 2^4\sqrt{2-\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2}}}}}\,,\;\ldots$$

converge to π.

by typing

```
The sequence
 $$
 2\sqrt{2}\,,\quad 2^2\sqrt{2-\sqrt{2}}\,,\quad 2^3
    \sqrt{2-\sqrt{2+\sqrt{2}}}}\,,\quad 2^4\sqrt{2-
     \sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2}}}}}}\,,\;\ldots
 $$
converge to $\pi$.
```

The `\ldots` command above produces …, the three dots indicating indefinite continuation, called *ellipsis* (more about them later). The command `\,` produces a "thinspace" (as opposed to a thickspace produced by `\;` , seen earlier). Why all this thin and thick spaces in the above input? Remove them and see the difference. (A tastefully applied thinspace is what makes a mathematical expression typeset in TEX really beautiful.)

The symbol π in the output produced by `$\pi$` maybe familiar from high school mathematics. It is a Greek letter named "pi". Mathematicians often use letters of the Greek alphabet (which even otherwise is Greek to many) and a multitude of other symbols in their work. A list of available symbols in LATEX is given at the end of this chapter.

8.1.3. MATHEMATICAL SYMBOLS

In the list at the end of this chapter, note that certain symbols are marked to be not available in native LaTeX, but only in certain packages. We will discuss some such packages later. Another thing about the list is that they are categorized into classes such as "Binary Relations", "Operators", "Functions" and so on. This is not merely a matter of convenience.

We have noted that TeX leaves some additional spaces around "binary operators" such as + and −. The same is true for any symbol classified as a binary operator. For example, consider the following

For real numbers x and y, define an operation $\circ$ by

$$x \circ y = x + y - xy$$

This operation is associative.

From the list of symbols (Table 8.2), we see that $\circ$ is produced by `\circ` and this is classified as a binary operator, so that we can produce this by

```
For real numbers $x$ and $y$, define an operation $\circ$ by
$$
x\circ y = x+y-xy
$$
This operation is associative.
```

Note the spaces surrounding the $\circ$ symbol in the output. On the other hand suppose you want

For real numbers x and y, define an operation $\Box$ by

$$x \Box y = x^2 + y^2$$

The list of symbols (Table 8.6) show that the symbol $\Box$ is produced by `\Box` but that it is available only in the package latexsym or amssymb. So if we load one of these using the `\usepackage` command and then type

```
For real numbers $x$ and $y$, define an operation $\Box$ by
$$
x\Box y = x^2+y^2
$$
```

you will only get

For real numbers x and y, define an operation $\Box$ by

$$x \Box y = x^2 + y^2$$

Notice the difference? There are no spaces around $\Box$; this is because, this symbol is not by default defined as a binary operator. (Note that it is classified under "Miscellaneous".) But *we* can ask TEX to consider this symbol as a binary operator by the command `\mathbin` before `\Box` as in

```
For real numbers $x$ and $y$, define an operation $\Box$ by
$$
x\mathbin\Box y=x^2+y^2
$$
```

and this will produce the output shown first.

This holds for "Relations" also. TEX leaves some space around "Relation" symbols and we can instruct TEX to consider any symbol as a relation by the command `\mathrel`. Thus we can produce

Define the relation ρ on the set of real numbers by $x \rho y$ iff $x - y$ is a rational number.

by typing

```
Define the relation $\rho$ on the set of real numbers by
$x\mathrel\rho y$ iff $x-y$ is a rational number.
```

(See what happens if you remove the `\mathrel` command.)

8.2. CUSTOM COMMANDS

We have seen that LATEX produces mathematics (and many other things as well) by means of "commands". The interesting thing is that we can build our own commands using the ones available. For example, suppose that the expression $(x_1, x_2, \ldots, x_n)$ occurs frequently in a document. If we now write

```
\newcommand{\vect}{(x_1,x_2,\dots,x_n)}
```

Then we can type `$\vect$` anywhere after wards to produce $(x_1, x_2, \ldots, x_n)$ as in

```
We often write $x$ to denote the vector $\vect$.
```

to get

We often write x to denote the vector $(x_1, x_2, \ldots, x_n)$.

(By the way, the best place to keep such "newcommands" is the preamble, so that you can use them anywhere in the document. Also, it will be easier to change the commands, if the need arises).

OK, we can now produce $(x_1, x_2, \ldots, x_n)$ with `$\vect$`, but how about $(y_1, y_2, \ldots, y_n)$ or $(z_1, z_2, \ldots, z_n)$? Do we have to define newcommands for each of these? Not at all. We can also define commands with *variable arguments* also. Thus if we change our definition of `\vect` to

```
\newcommand{\vect}[1]{(#1_1,#1_2,\dots,#1_n)}
```

Then we can use `$\vect{x}$` to produce $(x_1, x_2, \ldots, x_n)$ and `$\vect{a}$` to produce $(a_1, a_2, \ldots, a_n)$ and so on.

The form of this definition calls for some comments. The `[1]` in the `\newcommand` above indicates that the command is to have *one* (variable) argument. What about the `#1`? Before producing the output, each occurrence of `#1` will be replaced by the (single) argument we supply to `\vect` in the input. For example, the input `$\vect{a}$` will be changed to `$(a_1,a_2,\dots,a_n)$` at some stage of the compilation.

We can also define commands with more than one argument (the maximum number is 9). Thus for example, if the document contains not only $(x_1, x_2, \ldots, x_n)$, $(y_1, y_2, \ldots, y_n)$ and so on, but $(x_1, x_2, \ldots, x_m)$, $(y_1, y_2, \ldots, y_p)$ also, then we can change our definition of `\vect` to

```
\newcommand{\vect}[2]{(#1_1,#1_2,\dotsc,#1_#2)}
```

so that we can use `$\vect{x}{n}$` to produce $(x_1, x_2, \ldots, x_n)$ and `$\vect{a}{p}$` to produce $(a_1, a_2, \ldots, a_p)$.

8.3. MORE ON MATHEMATICS

There are some many other features of typesetting math in LaTeX, but these have better implementations in the package amsmath which has some additional features as well. So, for the rest of the chapter the discussion will be with reference to this

package and some allied ones. Thus all discussion below is under the assumption that the package amsmath has been loaded with the command `\usepackage{amsmath}`.

8.3.1. SINGLE EQUATIONS

In addition to the LaTeX commands for displaying math as discussed earlier, the amsmath also provides the

```
\begin{equation*} ... \end{equation*}
```

construct. Thus with this package loaded, the output

> The equation representing a straight line in the Cartesian plane is of the form
>
> $$ax + by + c = 0$$
>
> where a, b, c are constants.

can also be produced by

```
The equation representing a straight line in the
Cartesian plane is of the form
\begin{equation*}
  ax+by+c=0
\end{equation*}
where $a$, $b$, $c$ are constants.
```

Why the * after `equation`? Suppose we try it without the * as

```
The equation representing a straight line in the Cartesian
plane is of the form
\begin{equation}
  ax+by+c=0
\end{equation}
where $a$, $b$, $c$ are constants.
```

we get

> The equation representing a straight line in the Cartesian plane is of the form
>
> $$ax + by + c = 0 \tag{8.1}$$
>
> where a, b, c are constants.

This provides the equation with a *number*. We will discuss equation numbering in some more detail later on. For the time

being, we just note that for any environment name with a star we discuss here, the unstarred version provides the output with numbers.

Ordinary text can be inserted inside an equation using the `\text` command. Thus we can get

Thus for all real numbers x we have

$$x \le |x| \quad \text{and} \quad x \ge |x|$$

and so

$$x \le |x| \quad \text{for all } x \text{ in } R.$$

from

```
Thus for all real numbers $x$ we have
\begin{equation*}
  x\le|x|\quad\text{and}\quad x\ge|x|
\end{equation*}
and so
\begin{equation*}
  x\le|x|\quad\text{for all $x$ in $R$}.
\end{equation*}
```

Note the use of dollar signs in the second `\text` above to produce mathematical symbols within `\text`.

Sometimes a single equation maybe too long to fit into one line (or sometimes even *two* lines). Look at the one below:

$$(a+b+c+d+e)^2 = a^2+b^2+c^2+d^2+e^2 + 2ab+2ac+2ad+2ae+2bc+2bd+2be+2cd+2ce+2de$$

This is produced by the environment `multline*` (note the spelling carefully—it is *not* `mult`i`line`), as shown below.

```
\begin{multline*}
  (a+b+c+d+e)^2=a^2+b^2+c^2+d^2+e^2\\
       +2ab+2ac+2ad+2ae+2bc+2bd+2be+2cd+2ce+2de
\end{multline*}
```

`multline` can be used for equations requiring more than two lines, but without tweaking, the results are not very satisfactory. For example, the input

```
\begin{multline*}
  (a+b+c+d+e+f)^2=a^2+b^2+c^2+d^2+e^2+f^2\\
```

```
        +2ab+2ac+2ad+2ae+2af\\
        +2bc+2bd+2be+2bf\\
        +2cd+2ce+2cf\\
        +2de+2df\\
        +2ef
\end{multline*}
```

produces

$$\begin{aligned}(a+b+c+d+e+f)^2 = a^2+b^2+c^2+d^2+e^2+f^2 \\ +2ab+2ac+2ad+2ae+2af \\ +2bc+2bd+2be+2bf \\ +2cd+2ce+2cf \\ +2de+2df \\ +2ef\end{aligned}$$

By default, the `multline` environment places the first line flush left, the last line flush right (except for some indentation) and the lines in between, centered within the display.

A better way to typeset the above multiline (not multline) equation is as follows.

$$\begin{aligned}(a+b+c+d+e+f)^2 &= a^2+b^2+c^2+d^2+e^2+f^2 \\ &\quad +2ab+2ac+2ad+2ae+2af \\ &\quad +2bc+2bd+2be+2bf \\ &\quad +2cd+2ce+2cf \\ &\quad +2de+2df \\ &\quad +2ef\end{aligned}$$

This is done using the `split` environment as shown below.

```
\begin{equation*}
  \begin{split}
    (a+b+c+d+e+f)^2 &= a^2+b^2+c^2+d^2+e^2+f^2\\
                    &\quad +2ab+2ac+2ad+2ae+2af\\
                    &\quad +2bc+2bd+2be+2bf\\
                    &\quad +2cd+2ce+2cf\\
                    &\quad +2de+2df\\
                    &\quad +2ef
  \end{split}
\end{equation*}
```

Some comments seems to be in order. First note that the `split` environment cannot be used independently, but only inside some equation structure such as `equation` (and others we will soon see). Unlike `multline`, the `split` environment provides for alignment among the "split" lines (using the & character, as in `tabular`). Thus in the above example, all the + signs are aligned and these in turn are aligned with a point a `\quad` to the right of the = sign. It is also useful when the equation contains multiple equalities as in

$$\begin{aligned}(a+b)^2 &= (a+b)(a+b)\\ &= a^2+ab+ba+b^2\\ &= a^2+2ab+b^2\end{aligned}$$

which is produced by

```
\begin{equation*}
  \begin{split}
    (a+b)^2 & = (a+b)(a+b)\\
            & = a^2+ab+ba+b^2\\
            & = a^2+2ab+b^2
  \end{split}
\end{equation*}
```

8.3.2. GROUPS OF EQUATIONS

A group of displayed equations can be typeset in a single go using the `gather` environment. For example,

$$\begin{gathered}(a,b)+(c,d)=(a+c,b+d)\\ (a,b)(c,d)=(ac-bd,ad+bc)\end{gathered}$$

can be produced by

```
\begin{gather*}
  (a,b)+(c,d)=(a+c,b+d)\\
  (a,b)(c,d)=(ac-bd,ad+bc)
\end{gather*}
```

Now when several equations are to be considered one unit, the logically correct way of typesetting them is with some alignment (and it is perhaps easier on the eye too). For example,

Thus x, y and z satisfy the equations

$$x + y - z = 1$$
$$x - y + z = 1$$

This is obtained by using the `align*` environment as shown below

```
Thus $x$, $y$ and $z$ satisfy the equations
\begin{align*}
  x+y-z & = 1\\
  x-y+z & = 1
\end{align*}
```

We can add a short piece of text between the equations, without disturbing the alignment, using the `\intertext` command. For example, the output

Thus x, y and z satisfy the equations

$$x + y - z = 1$$
$$x - y + z = 1$$

and by hypothesis

$$x + y + z = 1$$

is produced by

```
Thus $x$, $y$ and $z$ satisfy the equations
\begin{align*}
  x+y-z & = 1\\
  x-y+z & = 1\\
  \intertext{and by hypothesis}
  x+y+z & =1
\end{align*}
```

We can also set multiple 'columns' of aligned equations side by side as in

Compare the following sets of equations

$$\cos^2 x + \sin^2 x = 1 \qquad \cosh^2 x - \sinh^2 x = 1$$
$$\cos^2 x - \sin^2 x = \cos 2x \qquad \cosh^2 x + \sinh^2 x = \cosh 2x$$

All that it needs are extra `&`'s to separate the columns as can be sen from the input

```
Compare the following sets of equations
\begin{align*}
  \cos^2x+\sin^2x & = 1
     & \cosh^2x-\sinh^2x & = 1\\
  \cos^2x-\sin^2x & = \cos 2x
     & \cosh^2x+\sinh^2x & = \cosh 2x
\end{align*}
```

We can also adjust the horizontal space between the equation columns. For example,

```
Compare the sets of equations
\begin{align*}
 \cos^2x+\sin^2x & = 1
                 &\qquad \cosh^2x-\sinh^2x
                 & = 1\\
 \cos^2x-\sin^2x & = \cos 2x
                 &\qquad \cosh^2x+\sinh^2x
                 & = \cosh 2x
\end{align*}
```

gives

Compare the sets of equations

$$\cos^2 x + \sin^2 x = 1 \qquad \cosh^2 x - \sinh^2 x = 1$$
$$\cos^2 x - \sin^2 x = \cos 2x \qquad \cosh^2 x + \sinh^2 x = \cosh 2x$$

Perhaps a nicer way of typesetting the above is

Compare the following sets of equations

$$\cos^2 x + \sin^2 x = 1 \qquad \cosh^2 x - \sinh^2 x = 1$$
$$\cos^2 x - \sin^2 x = \cos 2x \quad \text{and} \quad \cosh^2 x + \sinh^2 x = \cosh 2x$$

This cannot be produced by the equation structures discussed so far, because any of these environments takes up the entire width of the text for its display, so that we cannot put anything else on the same line. So amsmath provides variants `gathered`, `aligned` and `alignedat` which take up only the *actual width of the contents* for their display. Thus the above example is produced by the input

```
Compare the following sets of equations
\begin{equation*}
```

```
  \begin{aligned}
    \cos^2x+sin^2x  & = 1\\
    \cos^2x-\sin^2x & = \cos 2x
  \end{aligned}
  \qquad\text{and}\qquad
  \begin{aligned}
    \cosh^2x-\sinh^2x & = 1\\
    \cosh^2x+\sinh^2x & = \cosh 2x
  \end{aligned}
\end{equation*}
```

Another often recurring structure in mathematics is a display like this

$$|x| = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x \leq 0 \end{cases}$$

There is a special environment `cases` in amsmath to take care of these. The above example is in fact produced by

```
\begin{equation*}
  |x| =
  \begin{cases}
    x & \text{if $x\ge 0$}\\
    -x & \text{if $x\le 0$}
  \end{cases}
\end{equation*}
```

8.3.3. NUMBERED EQUATIONS

We have mentioned that each of the the 'starred' equation environments has a corresponding unstarred version, which also produces numbers for their displays. Thus our very first example of displayed equations with `equation` instead of `equation*` as in

```
The equation representing a straight line in the Cartesian
plane is of the form
\begin{equation}
  ax+by+c=0
\end{equation}
where $a$, $b$, $c$ are constants.
```

produces

The equation representing a straight line in the Cartesian plane is of the form

$$ax + by + c = 0 \tag{8.2}$$

where a, b, c are constants.

Why 8.2 for the equation number? Well, this is Equation number 2 of Chapter 8, isn't it? If you want the section number also in the equation number, just give the command

```
\numberwithin{equation}{section}
```

We can also override the number LaTeX produces with one of our own design with the `\tag` command as in

```
The equation representing a straight line in the Cartesian
plane is of the form
 \begin{equation}
  ax+by+c=0\tag{L}
\end{equation}
where $a$, $b$, $c$ are constants.
```

which gives

The equation representing a straight line in the Cartesian plane is of the form

$$ax + by + c = 0 \tag{L}$$

where a, b, c are constants.

There is also a `\tag*` command which typesets the *equation label* without parentheses.

What about numbering alignment structures? Except for `split` and `aligned`, all other alignment structures have unstarred forms which attach numbers to *each* aligned equation. For example,

```
\begin{align}
  x+y-z &= 1\\
  x-y+z &= 1
\end{align}
```

gives

$$x + y - z = 1 \tag{8.3}$$
$$x - y + z = 1 \tag{8.4}$$

Here is also, you can give a label of your own to *any* of the equations with the `\tag` command. Be careful to give the `\tag` *before* the end of line character `\\` though. (See what happens if you give a `\tag` command *after* a `\\`.) You can also suppress the label for any equation with the `\notag` command. These are illustrated in the sample input below:

```
Thus $x$, $y$ and $z$ satisfy the equations
\begin{align*}
  x+y-z &= 1\notag\\
  x-y+z &= 1\notag\\
  \intertext{and by hypothesis}
  x+y+z &=1\tag{H}
\end{align*}
```

which gives the following output

Thus x, y and z satisfy the equations

$$\begin{aligned} x+y-z &= 1\\ x-y+z &= 1 \end{aligned}$$

and by hypothesis

$$x+y+z = 1 \tag{H}$$

What about `split` and `aligned`? As we have seen, these can be used only within some other equation structure. The numbering or the lack of it is determined by this parent structure. Thus

```
\begin{equation}
  \begin{split}
    (a+b)^2 &= (a+b)(a+b)\\
            &= a^2+ab+ba+b^2\\
            &= a^2+2ab+b^2
  \end{split}
\end{equation}
```

gives

$$\begin{aligned} (a+b)^2 &= (a+b)(a+b)\\ &= a^2+ab+ba+b^2\\ &= a^2+2ab+b^2 \end{aligned} \tag{8.5}$$

8.4. MATHEMATICS MISCELLANY

There are more things Mathematics than just equations. Let us look at how LaTeX and in particular, the amsmath package deals with them.

8.4.1. MATRICES

Matrices are by definition numbers or mathematical expressions arranged in rows and columns. The amsmath has several environments for producing such arrays. For example

The system of equations

$$\begin{aligned} x+y-z &= 1\\ x-y+z &= 1\\ x+y+z &= 1 \end{aligned}$$

can be written in matrix terms as

$$\begin{pmatrix} 1 & 1 & -1\\ 1 & -1 & 1\\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x\\ y\\ z \end{pmatrix} = \begin{pmatrix} 1\\ 1\\ 1 \end{pmatrix}.$$

Here, the matrix $\begin{pmatrix} 1 & 1 & -1\\ 1 & -1 & 1\\ 1 & 1 & 1 \end{pmatrix}$ is invertible.

is produced by

```
The system of equations
\begin{align*}
  x+y-z &= 1\\
  x-y+z &= 1\\
  x+y+z &= 1
\end{align*}
can be written in matrix terms as
\begin{equation*}
  \begin{pmatrix}
    1 &  1 & -1\\
    1 & -1 &  1\\
    1 &  1 &  1
  \end{pmatrix}
  \begin{pmatrix}
    x\\
```

```
    y\\
    z
  \end{pmatrix} =
  \begin{pmatrix}
    1\\
    1\\
    1
  \end{pmatrix}.
\end{equation*}
Here, the matrix
$\begin{pmatrix}
  1 &  1 & -1\\
  1 & -1 &  1\\
  1 &  1 &  1
\end{pmatrix}$ is invertible.
```

Note that the environment `pmatrix` can be used within in-text mathematics or in displayed math. Why the `p`? There is indeed an environment `matrix` (without a `p`) but it produces an array *without* the enclosing parentheses (try it). If you want the array to be enclosed within *square brackets*, use `bmatrix` instead of `pmatrix`. Thus

Some mathematicians write matrices within parentheses as in $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ while others prefer square brackets as in $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$

is produced by

```
Some mathematicians write matrices within parentheses as in
$\begin{pmatrix}
  a & b\\
  c & d
\end{pmatrix}$  while others prefer square brackets as in
$\begin{bmatrix}
  a & b\\
  c & d
\end{bmatrix}$
```

There is also a `vmatrix` environment, which is usually used for determinants as in

The determinant $\begin{vmatrix} a & b \\ c & d \end{vmatrix}$ is defined by

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

which is obtained from the input

```
The determinant
$\begin{vmatrix}
  a & b\\
  c & d
\end{vmatrix}$ is defined by
\begin{equation*}
  \begin{vmatrix}
    a & b\\
    c & d
  \end{vmatrix} = ad -bc
\end{equation*}
```

There is a variant `Vmatrix` which encloses the array in double lines. Finally, we have a `Bmatrix` environment which produces an array enclosed within braces { }.

A row of dots in a matrix can be produced by the command `\hdotsfour`. it should be used with an argument specifying the number of columns to be spanned. For example, to get

A general $m \times n$ matrix is of the form

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ & \dots\dots\dots & & \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

we type

```
A general $m\times n$ matrix is of the form
\begin{equation*}
  \begin{pmatrix}
    a_{11} & a_{12} & \dots & a_{1n}\\
    a_{21} & a_{22} & \dots & a_{2n}\\
    \hdotsfor{4}\\
    a_{m1} & a_{m2} & \dots & a_{mn}
  \end{pmatrix}
\end{equation*}
```

The command `\hdotsfor` has also an optional argument to specify the spacing of dots. Thus in the above example, if we use `\hdotsfor[2]{4}`, then the space between the dots is doubled as in

A general $m \times n$ matrix is of the form

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ & \dots\dots\dots & & \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

8.4.2. DOTS

In the above example, we used the command `\dots` to produce a row of three dots. This can be used in other contexts also. For example,

```
Consider a finite sequence $X_1,X_2,\dots$, its sum
$X_1+X_2+\dots$ and product $X_1X_2\dots$.
```

gives

Consider a finite sequence $X_1, X_2, \dots$, its sum $X_1 + X_2 + \dots$ and product $X_1 X_2 \dots$.

Here the dots in all the three contexts are along the "baseline" of the text. Isn't it better to typeset this as

Consider a finite sequence $X_1, X_2, \dots$, its sum $X_1 + X_2 + \cdots$ and product $X_1 X_2 \cdots$.

with *raised* dots for addition and multiplication? The above text is typeset by the input

```
Consider a finite sequence $X_1,X_2,\dotsc$, its sum
$X_1+X_2+\dotsb$ and product $X_1X_2\dotsm$.
```

Here `\dotsc` stands for dots to be used with commas, `\dotsb` for dots with binary operations (or relations) and `\dotsm` for multiplication dots. There is also a `\dotsi` for dots with integrals as in

$$\int_{A_1}\int_{A_2}\cdots\int_{A_n} f$$

8.4.3. DELIMITERS

How do we produce something like

Since $\left|\begin{smallmatrix} a & h & g \\ h & b & f \\ g & f & c \end{smallmatrix}\right| = 0$, the matrix $\left(\begin{smallmatrix} a & h & g \\ h & b & f \\ g & f & c \end{smallmatrix}\right)$ is not invertible.

Here the 'small' in-text matrices are produced by the environment `smallmatrix`. This environment *does not* provide the enclosing *delimiters* ( ) or — — which we must supply as in

```
$\left|\begin{smallmatrix}
        a & h & g\\
        h & b & f\\
        g & f & c
      \end{smallmatrix}\right| = 0$, the matrix
$\left(\begin{smallmatrix}
        a & h & g\\
        h & b & f\\
        g & f & c
      \end{smallmatrix}\right)$ is not invertible.
```

Why the `\left|...\right|` and `\left{...\right}`? These commands `\left` and `\right` enlarge the *delimiter* following them to the size of the enclosed material. To see their effect, try typesetting the above example without these commands. The list of symbols at the end of this chapter gives a list of delimiters that are available off the shelf.

One interesting point about the `\left` and `\right` pair is that, though every `\left` should be matched to a `\right`, the *delimiters* to which they apply need not match. In particular we can produce a single large delimiter produced by `\left` or `\right` by matching it with a matching command followed by a period. For example,

$$\left.\begin{aligned} u_x &= v_y \\ u_y &= -v_x \end{aligned}\right\} \quad \text{Cauchy-Riemann Equations}$$

is produced by

```
\begin{equation*}
  \left.
    \begin{aligned}
      u_x &= v_y\\
      u_y &= -v_x
    \end{aligned}
  \right\}
  \quad\text{Cauchy-Riemann Equations}
\end{equation*}
```

There are instances where the delimiters produced by `\left` and `\right` are too small or too large. For example,

```
\begin{equation*}
  (x+y)^2-(x-y)^2=\left((x+y)+(x-y)\right)
  \left((x+y)-(x-y)\right)=4xy
\end{equation*}
```

gives

$$(x+y)^2-(x-y)^2=\left((x+y)+(x-y)\right)\left((x+y)-(x-y)\right)=4xy$$

where the parentheses are all of the same size. But it may be better to make the outer ones a little larger to make the nesting visually apparent, as in

$$(x+y)^2-(x-y)^2=\bigl((x+y)+(x-y)\bigr)\bigl((x+y)-(x-y)\bigr)=4xy$$

This is produced using the commands `\bigl` and `\bigr` before the outer parentheses as shown below:

```
\begin{equation*}
  (x+y)^2-(x-y)^2=\bigl((x+y)+(x-y)\bigr)
  \bigl((x+y)-(x-y)\bigr)=4xy
\end{equation*}
```

Apart from `\bigl` and `\bigr` there are `\Bigl`, `\biggl` and `\Biggl` commands (and their `r` counterparts) which (in order) produce delimiters of increasing size. (Experiment with them to get a feel for their sizes.)

As another example, look at

For n-tuples of complex numbers $(x_1,x_2,\dots,x_n)$ and $(y_1,y_2,\dots,y_n)$ of complex numbers

$$\left(\sum_{k=1}^{n}|x_k y_k|\right)^2\le\left(\sum_{k=1}^{n}|x_k|\right)\left(\sum_{k=1}^{n}|y_k|\right)$$

which is produced by

```
For $n$-tuples of complex numbers $(x_1,x_2,\dotsc,x_n)$
and $(y_1,y_2,\dotsc,y_n)$ of complex numbers
\begin{equation*}
  \left(\sum_{k=1}^n|x_ky_k|\right)^2\le
        \left(\sum_{k=1}^{n}|x_k|\right)
        \left(\sum_{k=1}^{n}|y_k|\right)
\end{equation*}
```

Does not the output below look better?

For n-tuples of complex numbers $(x_1, x_2, \dots, x_n)$ and $(y_1, y_2, \dots, y_n)$ of complex numbers

$$\left(\sum_{k=1}^{n} |x_k y_k|\right)^2 \le \left(\sum_{k=1}^{n} |x_k|\right)\left(\sum_{k=1}^{n} |y_k|\right)$$

This one is produced by

```
For $n$-tuples of complex numbers $(x_1,x_2,\dotsc,x_n)$
and $(y_1,y_2,\dotsc,y_n)$ of complex numbers
\begin{equation*}
  \biggl(\sum_{k=1}^n|x_ky_k|\biggr)^2\le
    \biggl(\sum_{k=1}^{n}|x_k|\biggr)
    \biggl(\sum_{k=1}^{n}|y_k|\biggr)
\end{equation*}
```

Here the trouble is that the delimiters produced by `\left` and `\right` are a bit too large.

8.4.4. PUTTING ONE OVER ANOTHER

Look at the following text

From the binomial theorem, it easily follows that if n is an even number, then

$$1 - \binom{n}{1}\frac{1}{2} + \binom{n}{2}\frac{1}{2^2} - \cdots - \binom{n}{n-1}\frac{1}{2^{n-1}} = 0$$

We have fractions like $\frac{1}{2^{n-1}}$ and *binomial coefficients* like $\binom{n}{2}$ here and the common feature of both is that they have one mathematical expression over another.

Fractions are produced by the `\frac` command which takes two arguments, the numerator followed by the denominator and the binomial coefficients are produced by the `\binom` command which also takes two arguments, the 'top' expression followed by the 'bottom' one. Thus the the input for the above example is

```
From the binomial theorem, it easily follows that if $n$ is
an even number, then
\begin{equation*}
  1-\binom{n}{1}\frac{1}{2}+\binom{n}{2}\frac{1}{2^2}
  -\dotsb  -\binom{n}{n-1}\frac{1}{2^{n-1}}=0
\end{equation*}
```

You can see from the first paragraph above that the *size* of the outputs of `\frac` and `\binom` are smaller in text than in display. This default behavior has to be modified sometimes for nicer looking output. For example, consider the following output

Since (x_n) converges to 0, there exists a positive integer p such that

$$|x_n| < \frac{1}{2} \quad \text{for all } n \ge p$$

Would not it be nicer to make the fraction smaller and typeset this as

Since (x_n) converges to 0, there exists a positive integer p such that

$$|x_n| < \tfrac{1}{2} \quad \text{for all } n \ge p$$

The second output is produced by the input

```
Since $(x_n)$ converges to $0$, there exists a positive
integer $p$ such that
\begin{equation*}
  |x_n|<\tfrac{1}{2}\quad\text{for all $n\ge p$}
\end{equation*}
```

Note the use of the command `\tfrac` to produce a smaller fraction. (The first output is produced by the usual `\frac` command.)

There is also command `\dfrac` to produce a display style (larger size) fraction in text. Thus the sentence after the first example in this (sub)section can be typeset as

We have fractions like $\dfrac{1}{2^{n-1}}$ and ...

by the input

```
We have fractions like $\dfrac{1}{2^{n-1}}$ and ...
```

As can be guessed, the original output was produced by `\frac`. Similarly, there are commands `\dbinom` (to produce display style binomial coefficients) and `\tbinom` (to produce text style binomial coefficients).

There is also a `\genfrac` command which can be used to produce custom fractions. To use it, we will have to specify six things

1. The left delimiter to be used—note that { must be specified as `\{`.
2. The right delimiter—again, } to be specified as `\}`.
3. The thickness of the horizontal line between the top expression and the bottom expression. If it is not specified, then it defaults to the 'normal' thickness. If it is set as `0pt` then there will be no such line at all in the output.
4. The size of the output—this is specified as an integer 0, 1, 2 or 3, greater values corresponding to *smaller* sizes. (Technically these values correspond to `\displaystyle`, `\textstyle`, `\scriptstyle` and `\scriptscriptstyle`.)
5. The top expression.
6. The bottom expression.

Thus instead of `\tfrac{1}{2}` we can also use

`\genfrac{}{}{}{1}{1}{2}` and instead of `\dbinom{n}{r}`,

we can also use `\genfrac{(}{)}{0pt}{0}{1}{2}` (but there is hardly any reason for doing so). More seriously, suppose we want to produce $\left\{{ij \atop k}\right\}$ and $\left[{ij \atop k}\right]$ as in

> The Christoffel symbol $\left\{{ij \atop k}\right\}$ of the second kind is related to the Christoffel symbol $\left[{ij \atop k}\right]$ of the first kind by the equation
>
> $$\left\{{ij \atop k}\right\} = g^{k1}\left[{ij \atop 1}\right] + g^{k2}\left[{ij \atop 2}\right]$$

This can be done by the input

```
 The Christoffel symbol $\genfrac{\{}{\}}{0pt}{}{ij}{k}$ of the
 second kind is related to the Christoffel symbol
 $\genfrac{[}{]}{0pt}{}{ij}{k}$ of the first kind by the equation
\begin{equation*}
 \genfrac{\{}{\}}{0pt}{}{ij}{k}=g^{k1}\genfrac{[}{]}{0pt}{}{ij}{1}
                               +g^{k2}\genfrac{[}{]}{0pt}{}{ij}{2}
\end{equation*}
```

If such expressions are frequent in the document, it would be better to define 'newcommands' for them and use them instead of `\genfrac` every time as in the following input (which produces the same output as above).

```
 \newcommand{\chsfk}[2]{\genfrac{[}{]}{0pt}{}{#1}{#2}}
```

```
\newcommand{\chssk}[2]{\genfrac{\{}{\}}{0pt}{}{#1}{#2}}
The Christoffel symbol $\chssk{ij}{k}$ of
the second kind is related to the Christoffel symbol
$\chsfk{ij}{k}$ of the first kind by the
equation
\begin{equation*}
  \chssk{ij}{k}=g^{k1}\chsfk{ij}{1}+g^{k2}\chsfk{ij}{2}
\end{equation*}
```

While on the topic of fractions, we should also mention the `\cfrac` command used to typeset continued fractions. For example, to get

$$\frac{4}{\pi} = 1 + \cfrac{1^2}{2 + \cfrac{3^2}{2 + \cfrac{5^2}{2 + \cdots}}}$$

simply type

```
\begin{equation*}
  \frac{4}{\pi}=1+\cfrac{1^2}{2+
                  \cfrac{3^2}{2+
                   \cfrac{5^2}{2+\dotsb}}}
\end{equation*}
```

Some mathematicians would like to write the above equation as

$$\frac{4}{\pi} = 1 + \frac{1^2}{2} + \frac{3^2}{2} + \frac{5^2}{2} + \cdots$$

There is no ready-to-use command to produce this, but we can define one as follows

```
\newcommand{\cfplus}{\mathbin{\genfrac{}{}{0pt}{}{}{+}}}
\begin{equation*}
  \frac{4}{\pi}
    =1+\frac{1^2}{2}\cfplus\frac{3^2}{2}\cfplus
    \frac{5^2}{2}\cfplus\dotsb
\end{equation*}
```

8.4.5. AFFIXING SYMBOLS—OVER OR UNDER

The table 8.11 at the end of this chapter gives various math mode *accents* such as `$\hat{a}$` to produce $\hat{a}$ and `$\dot{a}$`

to produce $\dot{a}$. But what if one needs $\overset{\circ}{a}$ or $\underset{\circ}{a}$? The commands `\overset` and `\underset` come to the rescue. Thus `$\overset{\circ}{a}$` produces $\overset{\circ}{a}$ and `$\underset{\circ}{a}$` produces $\underset{\circ}{a}$.

Basic LaTeX provides the commands `\overrightarrow` and `\overleftarrow` also to put (extensible) arrows over symbols, as can be seen from the table. The amsmath package also provides the commands `\underrightarrow` and `\underleftarrow` to put (extensible) arrows *below* mathematical expressions.

Speaking of arrows, amsmath provides the commands

`\xrightarrow` and `\xleftarrow`

which produces arrows which can accommodate long texts as superscripts or subscripts. Thus we can produce

Thus we see that
$$0 \longrightarrow A \xrightarrow{f} B \xrightarrow{g} C \longrightarrow 0$$
is a short exact sequence

from the input

```
Thus we see that
\begin{equation*}
  0\xrightarrow{} A\xrightarrow{f}
                  B\xrightarrow{g}
                  C\xrightarrow{} 0
\end{equation*}
is a short exact sequence
```

Note how the *mandatory* arguments of the first and last arrows are left empty to produce arrows with no superscripts. These commands also allow an *optional* argument (to be typed inside *square brackets*), which can be used to produce subscripts. For example

```
Thus we get
\begin{equation*}
  0\xrightarrow{} A\xrightarrow[\text{monic}]{f}
                  B\xrightarrow[\text{epi}]{g}
                  C\xrightarrow{} 0
\end{equation*}
```

gives

Thus we get
$$0 \to A \xrightarrow[\text{monic}]{f} B \xrightarrow[\text{epi}]{g} C \to 0$$

By the way, would not it be nicer to make the two middle arrows the same width? This can be done by changing the command for the third arrow (the one from B) as shown below

```
Thus we get
\begin{equation*}
  0\xrightarrow{} A\xrightarrow[\text{monic}]{f}
     B\xrightarrow[\hspace{7pt}\text{epi}\hspace{7pt}]{g}
     C\xrightarrow{}0
\end{equation*}
```

This gives

Thus we get
$$0 \to A \xrightarrow[\text{monic}]{f} B \xrightarrow[\text{epi}]{g} C \to 0$$

where the lengths of the two arrows are *almost* the same. There are indeed ways to make the lengths *exactly* the same, but we will talk about it in another chapter.

Mathematical symbols are also attached as *limits* to such *large operators* as sum ($\sum$), product ($\prod$) set union ($\bigcup$), set intersection ($\bigcap$) and so on. The limits are input as subscripts or superscripts, but their *positioning* in the output is different in text and display. For example, the input

```
Euler not only proved that the series
$\sum_{n=1}^\infty\frac{1}{n^2}$ converges, but also that
\begin{equation*}
  \sum_{n=1}^\infty\frac{1}{n^2}=\frac{\pi^2}{6}
\end{equation*}
```

gives the output

Euler not only proved that the series $\sum_{n=1}^{\infty} \frac{1}{n^2}$ converges, but also that
$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

Note that in display, the sum symbol is larger and the limits are put at the bottom and top (instead of at the sides,which

is usually the case for subscripts and superscripts). If you want the *same* type of symbol (size, limits and all) in text also, simply change the line

```
$\sum_{n=1}^\infty\frac{1}{n^2}$
```

to

```
$\displaystyle\sum_{n=1}^\infty\frac{1}{n^2}$
```

and you will get

Euler not only proved that the series $\displaystyle\sum_{n=1}^{\infty} \frac{1}{n^2}$ converges, but also that

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

(Note that this also changes the size of the fraction. What would you do to keep it small?) On the other hand, to make the displayed operator the same as in the text, add the command `\textstyle` before the `\sum` within the equation.

What if you only want to change the *position of the limits* but not the size of the operator in text? Then change the command `$\sum_{n=1}^\infty \frac{1}{n^2}$` to `$\sum_\limits {n=1}^\infty\frac{1}{n^2}$` and this will produce the output given below.

Euler not only proved that the series $\sum\limits_{n=1}^{\infty} \frac{1}{n^2}$ converges, but also that

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

On the other hand, if you want side-set limits in display type `\nolimits` after the `\sum` within the equation as in

```
Euler not only proved that the series
$\sum_{n=1}^\infty\frac{1}{n^2}$ converges, but also that
\begin{equation*}
  \sum\nolimits_{n=1}^\infty\frac{1}{n^2}=\frac{\pi^2}{6}
\end{equation*}
```

which gives

Euler not only proved that the series $\sum_{n=1}^{\infty} \frac{1}{n^2}$ converges, but also that

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

All these are true for other operators classified as "Variable-sized symbols" (Table 8.7), except integrals. Though the integral symbol in display is larger, the position of the limits in both text and display is on the side as can be seen from the output below

Thus $\lim_{x\to\infty} \int_0^x \frac{\sin x}{x}\,\mathrm{d}x = \frac{\pi}{2}$ and so by definition,

$$\int_0^{\infty} \frac{\sin x}{x}\,\mathrm{d}x = \frac{\pi}{2}$$

which is produced by

```
Thus
$\lim\limits_{x\to\infty}\int_0^x\frac{\sin x}{x}\,
  \mathrm{d}x  =\frac{\pi}{2}$
and so by definition,
\begin{equation*}
  \int_0^\infty\frac{\sin x}{x}\,\mathrm{d}x=\frac{\pi}{2}
\end{equation*}
```

If you want the limits to be above and below the integral sign, just add the command `\limits` immediately after the `\int` command. Thus

```
Thus
$\lim\limits_{x\to\infty}\int_0^x\frac{\sin x}{x}\,
   \mathrm{d}x  =\frac{\pi}{2}$
and so by definition,
\begin{equation*}
  \int\limits_0^\infty\frac{\sin x}{x}\,
  \mathrm{d}x=\frac{\pi}{2}
\end{equation*}
```

gives

Thus $\lim_{x\to\infty} \int_0^x \frac{\sin x}{x}\,\mathrm{d}x = \frac{\pi}{2}$ and so by definition,

$$\int\limits_0^{\infty} \frac{\sin x}{x}\,\mathrm{d}x = \frac{\pi}{2}$$

Now how do we typeset something like

$$p_k(x) = \prod_{\substack{i=1\\i\ne k}}^{n} \left(\frac{x-t_i}{t_k-t_i}\right)$$

where we have two lines of subscripts for $\prod$? There is a command `\substack` which will do the trick. The above output is obtained from

```
\begin{equation*}
  p_k(x)=\prod_{\substack{i=1\\i\ne k}}^n
              \left(\frac{x-t_i}{t_k-t_i}\right)
\end{equation*}
```

The amsmath package has also a `\sideset` command which can be used to put symbols at any of the four corners of a *large operator*. Thus

1. `$\sideset{_{ll}^{ul}}{_{lr}^{ur}}\bigcup$` produces $\sideset{_{ll}^{ul}}{_{lr}^{ur}}\bigcup$
2. `$\sideset{}{'}\sum$` produces $\sideset{}{'}\sum$.

8.5. NEW OPERATORS

Mathematical text is usually typeset in italics, and TEX follows this tradition. But certain functions in mathematics such as log, sin, lim and so on are traditionally typeset in roman. This is implemented in TEX by the use of commands like `$\log$`, `$\sin$`, `$\lim$` and so on. The symbols classified as "Log-like symbols" in Table 8.8 at the end of this chapter shows such functions which are predefined in LATEX.

Having read thus far, it may be no surprise to learn that we can define our own "operator names" which receive this special typographic treatment. This is done by the `\DeclareMathOperator` command. Thus if the operator cl occurs frequently in the document, you can make the declaration

```
\DeclareMathOperator{\cl}{cl}
```

in the preamble and then type `$\cl(A)$` to produce $\cl(A)$, for example.

Note that an operator defined like this accommodates subscripts and superscripts in the usual way, that is, at its sides. Thus

```
We denote the closure of $A$ in the subspace $Y$ of $X$ by
$\cl_Y(A)$
```

produces

We denote the closure of A in the subspace Y of X by $\mathrm{cl}_Y(A)$

If we want to define a new operator with subscripts and superscripts placed in the "limits" position below and above, then we should use the starred form of the `\DeclareMathOperator` as shown below

```
\DeclareMathOperator*{\esup}{ess\,sup}

For $f\in L^\infty(R)$, we define
\begin{equation*}
  ||f||_\infty=\esup_{x\in R}|f(x)|
\end{equation*}
```

(Note that the declaration *must* be done in the preamble.) This produces the output

For $f \in L^\infty(R)$, we define

$$||f||_\infty = \operatorname*{ess\,sup}_{x\in R} |f(x)|$$

(Why the `\,` command in the definition?)

8.6. THE MANY FACES OF MATHEMATICS

We have noted that most mathematics is typeset in italics typeface and some mathematical operators are typeset in an upright fashion. There may be need for additional typefaces as in typesetting vectors in boldface.

LaTeX includes several styles to typeset mathematics as shown in the table below

TYPE STYLE	COMD	EXAMPLE	
		INPUT	OUTPUT
italic (default)	`\mathit`	`$x+y=z$`	$x+y=z$
roman	`\mathrm`	`$\mathrm{x+y=z}$`	$\mathrm{x+y=z}$
bold	`\mathbf`	`$\mathbf{x+y=z}$`	$\mathbf{x+y=z}$
sans serif	`\mathsf`	`$\mathsf{x+y=z}$`	$\mathsf{x+y=z}$
typewriter	`\mathtt`	`$\mathtt{x+y=z}$`	$\mathtt{x+y=z}$
calligraphic (upper case only)	`\mathcal`	`$\mathcal{X+Y=Z}$`	$\mathcal{X+Y=Z}$

In addition to these, several other math alphabets are available in various packages (some of which are shown in the list of symbols at the end of this chapter).

Note that the command `\mathbf` produces only *roman* boldface and *not math italic* boldface. Sometimes you may need boldface math italic, for example to typeset vectors. For this, amsmath provides the `\boldsymbol` command. Thus we can get

In this case, we define
$$\boldsymbol{a}+\boldsymbol{b}=\boldsymbol{c}$$

from the input

```
In this case, we define
\begin{equation*}
  \boldsymbol{a}+\boldsymbol{b}=\boldsymbol{c}
\end{equation*}
```

If the document contains several occurrences of such symbols, it is better to make a new definition such as

```
\newcommand{\vect}[1]{\boldsymbol{#1}}
```

and then use `$\vect{a}$` to produce $\boldsymbol{a}$ and `$\vect{b}$` to produce $\boldsymbol{b}$ and so on. the additional advantage of this approach is that if you change your mind later and want vectors to be typeset with arrows above them as $\overrightarrow{a}$, then all you need is to change the `\boldsymol` part of the definition of `\vect` to `\overrightarrow` and the change will be effected throughout the document.

Now if we change the input of the above example as

```
In this case, we define
\begin{equation*}
  \boldsymbol{a+b=c}
\end{equation*}
```

then we get the output

In this case, we define
$$\boldsymbol{a+b=c}$$

Note that now the symbols + and = are also in boldface. Thus `\boldsymbol` makes bold every math symbol in its scope (provided the bold version of that symbol is available in the current math font).

There is another reason for tweaking the math fonts. Recently, the International Standards Organization (ISO) has established the recognized typesetting standards in mathematics. Some of the points in it are,

1. Simple variables are represented by italic letters as a, x.
2. Vectors are written in boldface italic as $\boldsymbol{a}$, $\boldsymbol{x}$.
3. Matrices may appear in sans serif as in A, X.
4. The special numbers e, i and the differential operator d are written in *upright roman*.

Point 1 is the default in LaTeX and we have seen how point 2 can be implemented. To fulfill Point 4, it is enough if we define something like

```
\newcommand{\me}{\mathrm{e}}
\newcommand{\mi}{\mathrm{i}}
\newcommand{\diff}{\mathrm{d}}
```

and then use `$\me$` for e and `$\mi$` for i and `$\diff x$` for $\mathrm{d}x$.

Point 3 can be implemented using `\mathsf` but it is a bit difficult (but not impossible) if we need them to be in *italic* also. The solution is to create a new math alphabet, say, `\mathsfsl` by the command

```
\DeclareMathAlphabet{\mathsfsl}{OT1}{cmss}{m}{sl}
```

(in the preamble) and use it to define a command `\matr` to typeset matrices in this font by

```
\newcommand{\matr}[1]{\ensuremath{\mathsfsl{#1}}}
```

so that `$\matr A$` produces $\matr{A}$.

8.7. AND THAT IS NOT ALL!

We have only briefly discussed the basic techniques of typesetting mathematics using LaTeX and some of the features of the amsmath package which helps us in this task. For more details on this package see the document `amsldoc.pdf` which should be available with your TeX distribution. If you want to produce *really* beautiful mathematical documents, read the Master—"The TeX Book" by Donald Knuth, especially Chapter 18, "Fine Points of Mathematics Typing".

8.8. SYMBOLS

Table 8.1: Greek Letters

α	\alpha	θ	\theta	o	o	τ	\tau
β	\beta	ϑ	\vartheta	π	\pi	υ	\upsilon
γ	\gamma	ι	\iota	ϖ	\varpi	ϕ	\phi
δ	\delta	κ	\kappa	ρ	\rho	φ	\varphi
ϵ	\epsilon	λ	\lambda	ϱ	\varrho	χ	\chi
ε	\varepsilon	μ	\mu	σ	\sigma	ψ	\psi
ζ	\zeta	ν	\nu	ς	\varsigma	ω	\omega
η	\eta	ξ	\xi				
Γ	\Gamma	Λ	\Lambda	Σ	\Sigma	Ψ	\Psi
Δ	\Delta	Ξ	\Xi	Υ	\Upsilon	Ω	\Omega
Θ	\Theta	Π	\Pi	Φ	\Phi		

Table 8.2: Binary Operation Symbols

$\pm$	\pm	$\cap$	\cap	$\diamond$	\diamond	$\oplus$	\oplus
$\mp$	\mp	$\cup$	\cup	$\triangle$	\bigtriangleup	$\ominus$	\ominus
$\times$	\times	$\uplus$	\uplus	$\triangledown$	\bigtriangledown	$\otimes$	\otimes
$\div$	\div	$\sqcap$	\sqcap	$\triangleleft$	\triangleleft	$\oslash$	\oslash
$*$	\ast	$\sqcup$	\sqcup	$\triangleright$	\triangleright	$\odot$	\odot
$\star$	\star	$\vee$	\vee	$\lhd$	\lhd*	$\bigcirc$	\bigcirc
$\circ$	\circ	$\wedge$	\wedge	$\rhd$	\rhd*	$\dagger$	\dagger
$\bullet$	\bullet	$\setminus$	\setminus	$\unlhd$	\unlhd*	$\ddagger$	\ddagger
$\cdot$	\cdot	$\wr$	\wr	$\unrhd$	\unrhd*	$\amalg$	\amalg
$+$	+	$-$	-				

* Not predefined in LaTeX 2ε. Use one of the packages latexsym, amsfonts or amssymb.

Table 8.3: Relation Symbols

$\leq$	\leq	$\geq$	\geq	$\equiv$	\equiv	$\models$	\models
$\prec$	\prec	$\succ$	\succ	$\sim$	\sim	$\perp$	\perp
$\preceq$	\preceq	$\succeq$	\succeq	$\simeq$	\simeq	$\mid$	\mid
$\ll$	\ll	$\gg$	\gg	$\asymp$	\asymp	$\parallel$	\parallel
$\subset$	\subset	$\supset$	\supset	$\approx$	\approx	$\bowtie$	\bowtie
$\subseteq$	\subseteq	$\supseteq$	\supseteq	$\cong$	\cong	$\bowtie$	\Join*

$\sqsubset$	\sqsubset*	$\sqsupset$	\sqsupset*	$\neq$	\neq	$\smile$	\smile
$\sqsubseteq$	\sqsubseteq	$\sqsupseteq$	\sqsupseteq	$\doteq$	\doteq	$\frown$	\frown
$\in$	\in	$\ni$	\ni	$\propto$	\propto	$=$	=
$\vdash$	\vdash	$\dashv$	\dashv	$<$	<	$>$	>
:	:						

* Not predefined in LaTeX 2ε. Use one of the packages latexsym, amsfonts or amssymb.

Table 8.4: Punctuation Symbols

,	,	;	;	:	\colon	.	\ldotp	·	\cdotp

Table 8.5: Arrow Symbols

$\leftarrow$	\leftarrow	$\longleftarrow$	\longleftarrow	$\uparrow$	\uparrow
$\Leftarrow$	\Leftarrow	$\Longleftarrow$	\Longleftarrow	$\Uparrow$	\Uparrow
$\rightarrow$	\rightarrow	$\longrightarrow$	\longrightarrow	$\downarrow$	\downarrow
$\Rightarrow$	\Rightarrow	$\Longrightarrow$	\Longrightarrow	$\Downarrow$	\Downarrow
$\leftrightarrow$	\leftrightarrow	$\longleftrightarrow$	\longleftrightarrow	$\updownarrow$	\updownarrow
$\Leftrightarrow$	\Leftrightarrow	$\Longleftrightarrow$	\Longleftrightarrow	$\Updownarrow$	\Updownarrow
$\mapsto$	\mapsto	$\longmapsto$	\longmapsto	$\nearrow$	\nearrow
$\hookleftarrow$	\hookleftarrow	$\hookrightarrow$	\hookrightarrow	$\searrow$	\searrow
$\leftharpoonup$	\leftharpoonup	$\rightharpoonup$	\rightharpoonup	$\swarrow$	\swarrow
$\leftharpoondown$	\leftharpoondown	$\rightharpoondown$	\rightharpoondown	$\nwarrow$	\nwarrow
$\rightleftharpoons$	\rightleftharpoons	$\rightsquigarrow$	\leadsto*		

* Not predefined in LaTeX 2ε. Use one of the packages latexsym, amsfonts or amssymb.

Table 8.6: Miscellaneous Symbols

…	\ldots	⋯	\cdots	⋮	\vdots	⋱	\ddots
$\aleph$	\aleph	′	\prime	$\forall$	\forall	∞	\infty
$\hbar$	\hbar	$\emptyset$	\emptyset	$\exists$	\exists	□	\Box*
$\imath$	\imath	∇	\nabla	¬	\neg	◇	\Diamond*
$\jmath$	\jmath	$\surd$	\surd	♭	\flat	△	\triangle
ℓ	\ell	⊤	\top	♮	\natural	♣	\clubsuit
$\wp$	\wp	⊥	\bot	♯	\sharp	◇	\diamondsuit
$\Re$	\Re	∥	\\|	\	\backslash	♡	\heartsuit
$\Im$	\Im	∠	\angle	∂	\partial	♠	\spadesuit
$\mho$	\mho*	.	.	|	|		

* Not predefined in LaTeX 2ε. Use one of the packages latexsym, amsfonts or amssymb.

Table 8.7: Variable-sized Symbols

$\sum$	\sum	$\bigcap$	\bigcap	$\bigodot$	\bigodot
$\prod$	\prod	$\bigcup$	\bigcup	$\bigotimes$	\bigotimes
$\coprod$	\coprod	$\bigsqcup$	\bigsqcup	$\bigoplus$	\bigoplus
$\int$	\int	$\bigvee$	\bigvee	$\biguplus$	\biguplus
$\oint$	\oint	$\bigwedge$	\bigwedge		

Table 8.8: Log-like Symbols

\arccos	\cos	\csc	\exp	\ker	\limsup	\min	\sinh
\arcsin	\cosh	\deg	\gcd	\lg	\ln	\Pr	\sup
\arctan	\cot	\det	\hom	\lim	\log	\sec	\tan
\arg	\coth	\dim	\inf	\liminf	\max	\sin	\tanh

Table 8.9: Delimiters

(	(	)	)	$\uparrow$	\uparrow	$\Uparrow$	\Uparrow
[	[	]	]	$\downarrow$	\downarrow	$\Downarrow$	\Downarrow
{	\{	}	\}	$\updownarrow$	\updownarrow	$\Updownarrow$	\Updownarrow
$\lfloor$	\lfloor	$\rfloor$	\rfloor	$\lceil$	\lceil	$\rceil$	\rceil
$\langle$	\langle	$\rangle$	\rangle	/	/	\	\backslash
|	|	$\|$	\\|			

Table 8.10: Large Delimiters

$\backslash$	\rmoustache	$\int$	\lmoustache	)	\rgroup	(	\lgroup
$\|$	\arrowvert	$\|\|$	\Arrowvert	$\|$	\bracevert		

Table 8.11: Math Mode Accents

$\hat{a}$	\hat{a}	$\acute{a}$	\acute{a}	$\bar{a}$	\bar{a}	$\dot{a}$	\dot{a}
$\breve{a}$	\breve{a}	$\check{a}$	\check{a}	$\grave{a}$	\grave{a}	$\vec{a}$	\vec{a}
$\ddot{a}$	\ddot{a}	$\tilde{a}$	\tilde{a}				

Table 8.12: Some other Constructions

$\widetilde{abc}$	\widetilde{abc}	$\widehat{abc}$	\widehat{abc}
$\overleftarrow{abc}$	\overleftarrow{abc}	$\overrightarrow{abc}$	\overrightarrow{abc}
$\overline{abc}$	\overline{abc}	$\underline{abc}$	\underline{abc}
$\overbrace{abc}$	\overbrace{abc}	$\underbrace{abc}$	\underbrace{abc}
$\sqrt{abc}$	\sqrt{abc}	$\sqrt[n]{abc}$	\sqrt[n]{abc}
f'	f'	$\frac{abc}{xyz}$	\frac{abc}{xyz}

Table 8.13: AMS Delimiters

⌜	\ulcorner	⌝	\urcorner
⌞	\llcorner	⌟	\lrcorner

Table 8.14: AMS Arrows

⇢	\dashrightarrow	⇠	\dashleftarrow
⇇	\leftleftarrows	⇆	\leftrightarrows
⇚	\Lleftarrow	↞	\twoheadleftarrow
↢	\leftarrowtail	↫	\looparrowleft
⇋	\leftrightharpoons	↶	\curvearrowleft
↺	\circlearrowleft	↰	\Lsh
⇈	\upuparrows	↿	\upharpoonleft
⇃	\downharpoonleft	⊸	\multimap
↭	\leftrightsquigarrow	⇉	\rightrightarrows
⇄	\rightleftarrows	⇉	\rightrightarrows
⇄	\rightleftarrows	↠	\twoheadrightarrow
↣	\rightarrowtail	↬	\looparrowright
⇌	\rightleftharpoons	↷	\curvearrowright
↻	\circlearrowright	↱	\Rsh
⇊	\downdownarrows	↾	\upharpoonright
⇂	\downharpoonright	⇝	\rightsquigarrow

Table 8.15: AMS Negated Arrows

↚	\nleftarrow	↛	\nrightarrow	⇍	\nLeftarrow
⇏	\nRightarrow	↮	\nleftrightarrow	⇎	\nLeftrightarrow

Table 8.16: AMS Greek

F	\digamma	$\varkappa$	\varkappa

Table 8.17: AMS Hebrew

ב	\beth	ד	\daleth	ג	\gimel

Table 8.18: AMS Miscellaneous

$\hbar$	\hbar	$\hslash$	\hslash
□	\square	◊	\lozenge
∡	\measuredangle	∄	\nexists
⅁	\Game	⅂	\Bbbk
▲	\blacktriangle	▼	\blacktriangledown
★	\bigstar	∢	\sphericalangle
／	\diagup	＼	\diagdown
△	\vartriangle	▽	\triangledown
Ⓢ	\circledS	∠	\angle
℧	\mho	ⅎ	\Finv
‵	\backprime	∅	\varnothing
■	\blacksquare	♦	\blacklozenge
∁	\complement	ð	\eth

Table 8.19: AMS Binary Operators

∔	\dotplus	∖	\smallsetminus	⋒	\Cap
⊼	\barwedge	⊻	\veebar	⩓	\doublebarwedge
⊠	\boxtimes	⊡	\boxdot	⊞	\boxplus
⋉	\ltimes	⋊	\rtimes	⋋	\leftthreetimes
⋏	\curlywedge	⋎	\curlyvee	⊖	\circleddash
◎	\circledcirc	⋅	\centerdot	⊺	\intercal
⋓	\Cup	⊟	\boxminus	⋇	\divideontimes
⋌	\rightthreetimes	⊛	\circledast		

Table 8.20: AMS Binary Relations

≦	\leqq	⩽	\leqslant	⪕	\eqslantless
≲	\lessapprox	≊	\approxeq	⋖	\lessdot
≶	\lessgtr	≦	\lesseqgtr	⪋	\lesseqqgtr

≒ \risingdotseq	≓ \fallingdotseq	∽ \backsim
⫅ \subseteqq	⋐ \Subset	⊏ \sqsubset
⋞ \curlyeqprec	≾ \precsim	⪷ \precapprox
⊴ \trianglelefteq	⊨ \vDash	⊪ \Vvdash
⌢ \smallfrown	≏ \bumpeq	≎ \Bumpeq
⩾ \geqslant	⪖ \eqslantgtr	≳ \gtrsim
⩾ \gtrdot	⋙ \ggg	≷ \gtrless
⪌ \gtreqqless	≗ \eqcirc	≗ \circeq
∼ \thicksim	≈ \thickapprox	⫆ \supseteqq
⊐ \sqsupset	≽ \succcurlyeq	⋟ \curlyeqsucc
⪸ \succapprox	▷ \vartriangleright	⊵ \trianglerighteq
∣ \shortmid	∥ \shortparallel	≬ \between
∝ \varpropto	◀ \blacktriangleleft	∴ \therefore
▶ \blacktriangleright	∵ \because	≲ \lesssim
⋘ \lll	≑ \doteqdot	⋍ \backsimeq
≼ \preccurlyeq	◁ \vartriangleleft	⌣ \smallsmile
⫆ \geqq	≳ \gtrapprox	⪌ \gtreqless
≜ \triangleq	⋑ \Supset	≿ \succsim
⊩ \Vdash	⫛ \pitchfork	϶ \backepsilon

Table 8.21: AMS Negated Binary Relations

≮ \nless	≰ \nleq	⪵ \nleqslant
⪇ \lneq	⪅ \lneqq	≨ \lvertneqq
⪉ \lnapprox	⊀ \nprec	⪯ \npreceq
⪹ \precnapprox	≁ \nsim	∤ \nshortmid
⊬ \nvdash	⊭ \nvDash	⋪ \ntriangleleft
⊈ \nsubseteq	⊊ \subsetneq	⫋ \varsubsetneq
⫋ \varsubsetneqq	≯ \ngtr	≱ \ngeq
≱ \ngeqq	⪈ \gneq	⪈ \gneqq
≩ \gnsim	⪊ \gnapprox	⊁ \nsucc
⪰ \nsucceq	≿ \succnsim	⪺ \succnapprox
∦ \nshortparallel	∦ \nparallel	⊯ \nvDash
⋫ \ntriangleright	⋭ \ntrianglerighteq	⫍ \nsupseteq
⊋ \supsetneq	⫌ \varsupsetneq	⫌ \supsetneqq
⪵ \nleqq	≴ \lnsim	⊀ \precnsim
∤ \nmid	⋬ \ntrianglelefteq	⫋ \subsetneqq
≱ \ngeqslant	≨ \gvertneqq	⪰ \nsucceq
≇ \ncong	⊮ \nVDash	⫌ \nsupseteqq
⫌ \varsupsetneqq		

Table 8.22: Math Alphabets

		Required package
ABCdef	`\mathrm{ABCdef}`	
ABCdef	`\mathitABCdef`	
ABCdef	`\mathnormal{ABCdef}`	
$\mathcal{ABC}$	`\mathcal{ABC}`	
$\mathcal{ABC}$	`\mathcal{ABC}`	euscript with option: mathcal
	`\mathscr{ABC}`	euscript with option: mathcr
$\mathfrak{ABCdef}$	`\mathfrak{ABCdef}`	eufrak
$\mathbb{ABC}$	`\mathbb{ABC}`	amsfonts or amssymb
$\mathscr{ABC}$	`\mathscr{ABC}`	mathrsfs

CHAPTER IX

TYPESETTING THEOREMS

IN MATHEMATICAL documents we often have special statements such as *axioms* (which are nothing but the assumptions made) and *theorems* (which are the conclusions obtained, sometimes known by other names like *propositions* or *lemmas*). These are often typeset in different font to distinguish them from surrounding text and given a name and a number for subsequent reference. Such distinguished statements are now increasingly seen in other subjects also. We use the term *theorem-like statements* for all such statements.

9.1. THEOREMS IN LaTeX

LaTeX provides the declaration `\newtheorem` to define the theorem-like statements needed in a document. This command has two arguments, the first for *the name we assign to the environment* and the second, *the name to be printed with the statement*. Thus if you want

> **Theorem 1.** *The sum of the angles of a triangle is* 180°.

you first specify

```
\newtheorem{thm}{Theorem}
```

and then type

```
\begin{thm}
  The sum of the angles of a triangle is $180^\circ$.
\end{thm}
```

Note that in the command `\newtheorem` the first argument can be any name you fancy, instead of the `thm` given here. Also, it is

a good idea to keep all your `\newtheorem` commands together in the preamble.

The `\newtheorem` command has a couple of optional arguments which control the way the corresponding statement is numbered. For example if you want the above theorem to be numbered 1.1 (the first theorem of the first section) rather than a plain 1, then you must specify

```
\newtheorem{thm}{Theorem}[section]
```

in the `\newtheorem` command. Then the same input as above for the theorem produces

Theorem 9.1.1. *The sum of the angles of a triangle is* 180°.

The next **Theorem** will be numbered 1.2, the third **Theorem** in the fourth section will be numbered 1.3 and so on.

The other optional argument of the `\newtheorem` command is useful when you have several different types of theorem-like statements (such as lemmas and corollaries) and you want some of them to share the same numbering sequence. For example if you want

Theorem 9.1.2. *The sum of the angles of a triangle is* 180°.

An immediate consequence of the result is the following

Corollary 9.1.3. *The sum of the angles of a quadrilateral is* 360°.

Then you must specify

```
\newtheorem{cor}[thm]{Corollary}
```

after the specification `\newtheorem{thm}[section]` and then type

```
\begin{thm}
  The sum of the angles of a triangle is $180^\circ$.
\end{thm}
```

An immediate consequence of the result is the following

Corollary 9.1.4. *The sum of the angles of a quadrilateral is* 360°.

The optional argument `thm` in the definition of the `cor` environment specifies that "Corollaries" and "Theorems" are to be numbered in the same sequence.

A theorem-like environment defined using the `\newtheorem` command has also an optional argument which is used to give a *note* about the theorem such as the name of its discoverer or its own common name. For example, to get

Theorem 9.1.5 (Euclid). *The sum of the angles of a triangle is* 180°.

you must type

```
\begin{thm}[Euclid]
  The sum of the angles of a triangle is $180^\circ$.
\end{thm}
```

Note the optional argument `Euclid` after the `\begin{thm}`. This use of [...] for optional notes sometimes lead to unintended results. For example, to get

Theorem 9.1.6. $[0,1]$ *is a compact subset of* $\mathbb{R}$.

if you type

```
\begin{thm}
  [0,1] is a compact subset of $\mathbb{R}$.
\end{thm}
```

then you get

Theorem 9.1.7 (0,1). *is a compact subset of* $\mathbb{R}$.

Do you see what happened? The string 0,1 within [ ] at the beginning of the theorem is considered an optional note by LaTeX! The correct way is to type

```
\begin{thm}
  $[0,1]$ is a compact subset of $\mathbb{R}$.
\end{thm}
```

Now all the theorem-like statements produced above have the *same typographical form*— name and number in **boldface** and the body of the statement in *italics*. What if you need something like

THEOREM 9.1.1 (Euclid). *The sum of the angles of a triangle is* 180°.

Such customization is necessitated not only by the aesthetics of the author but often by the whims of the designers in publishing houses also.

9.2. DESIGNER THEOREMS—THE AMSTHM PACKAGE

The package amsthm affords a high level of customization in formatting theorem-like statements. Let us first look at the predefined *styles* available in this package.

9.2.1. READY MADE STYLES

The default style (this is what you get if you do not say anything about the style) is termed `plain` and it is what we have seen so far—name and number in boldface and body in italic. Then there is the `definition` style which gives name and number in boldface and body in roman. And finally there is the `remark` style which gives number and name in italics and body in roman.

For example if you put in the preamble

```
\usepackage{amsthm}
\newtheorem{thm}{Theorem}[section]
\theoremstyle{definition}
\newtheorem{dfn}{Definition}[section]
\theoremstyle{remark}
\newtheorem{note}{Note}[section]
\theoremstyle{plain}
\newtheorem{lem}[thm]{Lemma}
```

and then type somewhere in your document

```
\begin{dfn}
A triangle is the figure formed by joining each pair
of three non collinear points by line segments.
\end{dfn}

\begin{note}
A triangle has three angles.
\end{note}

\begin{thm}
The sum of the angles of a triangle is $180^\circ$.
```

```
\end{thm}

\begin{lem}
The sum of any two sides of a triangle is greater than
or equal to the third.
\end{lem}
```

then you get

Definition 9.2.1. A triangle is the figure formed by joining each pair of three non collinear points by line segments.

Note 9.2.1. A triangle has three angles. [1]note

Theorem 9.2.1. *The sum of the angles of a triangle is* 180°.

Lemma 9.2.2. *The sum of any two sides of a triangle is greater than or equal to the third.*

Note how the `\theoremstyle` command is used to switch between various styles, especially the last `\theoremstyle{plain}` command. Without it, the previous `\theoremstyle{remark}` will still be in force when `lem` is defined and so "Lemma" will be typeset in the `remark` style.

9.2.2. CUSTOM MADE THEOREMS

Now we are ready to roll our own "theorem styles". This is done via the `\newtheoremstyle` command, which allows us to control almost all aspects of typesetting theorem like statements. this command has nine parameters and the general syntax is

```
\newtheoremstyle%
  {name}%
  {abovespace}%
  {belowspace}%
  {bodyfont}%
  {indent}%
  {headfont}%
  {headpunct}%
  {headspace}%
  {custom-head-spec}%
```

The first parameter *name* is the name of the new *style*. Note that it is *not* the name of the *environment* which is to be used later. Thus in the example above `remark` is the name of a new style for typesetting theorem like statements and `note` is the name of the environment subsequently defined to have this style (and `Note` is the name of the statement itself).

The next two parameters determine the vertical space between the theorem and the surrounding text—the *abovespace* is the space from the preceding text and the *belowspace* the space from the following text. You can specify either a rigid length (such as 12pt) or a rubber length (such as `\baselineskip`) as a value for either of these. Leaving either of these empty sets them to the "usual values" (Technically the `\topsep`).

The fourth parameter *bodyfont* specifies the font to be used for the body of the theorem-like statement. This is to be given as a *declaration* such as `\scshape` or `\bfseries` and *not* as a *command* such as `\textsc` or `\textbf`. If this is left empty, then the main text font of the document is used.

The next four parameters refer to the *theoremhead*—the part of the theorem like statement consisting of the name, number and the optional note. The fifth parameter *indent* specifies the indentation of *theoremhead* from the left margin. If this is empty, then there is no indentation of the *theoremhead* from the left margin. The next parameter specifies the font to be used for the *theoremhead*. The comments about the parameter *bodyfont*, made in the previous paragraph holds for this also. The parameter *headpunct* (the seventh in our list) is for specifying the *punctuation* after the theoremhead. If you do not want any, you can leave this empty. The last parameter in this category (the last but one in the entire list), namely *headspace*, determines the (horizontal) space to be left between the *theoremhead* and the *theorembody*. If you want only a normal interword space here put a *single blank space* as `{ }` in this place. (Note that it is not the same as leaving this *empty* as in `{}`.) Another option here is to put the command `\newline` here. Then instead of a space, you get a linebreak in the output; that is, the *theoremhead* will be printed in a line by itself and the *theorembody* starts from the next line.

The last parameter *custom-head-spec* is for customizing *theoremheads*. Since it needs some explanation (and since we

are definitely in need of some breathing space), let us now look at a few examples using the eight parameters we've already discussed.

It is almost obvious now how the last theorem in Section 1 (see Page 139) was designed. It was generated by

```
\newtheoremstyle{mystyle}{}{}{\slshape}{}{\scshape}{.}{ }{}
\theoremstyle{mystyle}
\newtheorem{mythm}{Theorem}[section]
\begin{mythm}
 The sum of the angles of a triangle is $180^\circ$.
\end{mythm}
```

As another example, consider the following

```
\newtheoremstyle{mynewstyle}{12pt}{12pt}{\itshape}%
  {}{\sffamily}{:}{\newline}{}
\theoremstyle{mynewstyle}
\newtheorem{mynewthm}{Theorem}[section]
\begin{mynewthm}[Euclid]
 The sum of the angles of a triangle is $180^\circ$.
\end{mynewthm}
```

This produces

Theorem 9.2.1 (Euclid):
The sum of the angles of a triangle is 180°.

Do you need anything more? Perhaps yes. Note that *theoremhead* includes the optional note to the theorem also, so that the font of the number and name of the theorem-like statement and that of the optional note are always the same. What if you need something like

Cauchy's Theorem (Third Version). *If G is a simply connected open subset of $\mathbb{C}$, then for every closed rectifiable curve γ in G, we have*

$$\int_\gamma f = 0.$$

It is in such cases, that the last parameter of `\newtheoremstyle` is needed. Using it we can separately customize the name and number of the theorem-like statement and also the optional note. The basic syntax for setting this parameter is

```
{commands#1commands#2commands#3}
```

where `#1` corresponds to the name of the theorem-like statement, `#2` corresponds to its number and `#3` corresponds to the optional note. We are here actually supplying the replacement text for a command `\thmhead` which has three arguments. It is as if we are defining

```
\renewcommand{\thmhead}[3]{...#1...#2...#3}
```

but without actually typing the `\renewcommand{\thmhead}[3]`. For example the theorem above (Cauchy's Theorem) was produced by

```
\newtheoremstyle{nonum}{}{}{\itshape}{}{\bfseries}{.}{ }
 {#1 (\mdseries #3)}
\theoremstyle{nonum}
\newtheorem{Cauchy}{Cauchy's Theorem}

\begin{Cauchy}[Third Version]
If $G$ is a simply connected open subset of $\mathbb{C}$,
then for every closed rectifiable curve $\gamma$ in $G$,
we have
 \begin{equation*}
   \int_\gamma f=0.
 \end{equation*}
\end{Cauchy}
```

Note that the absence of `#2` in the *custom-head-spec*, suppresses the theorem number and that the *space* after `#1` and the command `(\mdseries#3)` sets the optional note in medium size within parentheses and with a preceding space.

Now if you try to produce

Riemann Mapping Theorem. *Every open simply connected proper subset of $\mathbb{C}$ is analytically homeomorphic to the open unit disk in $\mathbb{C}$.*

by typing

```
\theoremstyle{nonum}
\newtheorem{Riemann}{Riemann Mapping Theorem}

\begin{Riemann}Every open simply connected proper subset of
  $\mathbb{C}$ is analytically homeomorphic to the open unit
  disk in $\mathbb{C}$.
\end{Riemann}
```

you will get

Riemann Mapping Theorem ( ). *Every open simply connected proper subset of* $\mathbb{C}$ *is analytically homeomorphic to the open unit disk in* $\mathbb{C}$.

Do you see what has happened? In the

```
\theoremstyle{nonum},
```

the parameter controlling the *note* part of the *theoremhead* was defined as `(\mdseries #3)` and in the `\newtheorem{Riemann}`, there is no optional note, so that in the output, you get an empty "note", *enclosed in parantheses* (and also with a preceding space).

To get around these difficulties, you can use the commands

`\thmname`, `\thmnumber` and `\thmnote`

within the {*custom-head-spec*} as

```
{\thmname{commands#1}%
   \thmnumber{commands#2}%
   \thmnote{commands#3}}
```

Each of these three commands will typeset its argument *if and only if the corresponding argument in the* `\thmhead` *is non empty*. Thus the correct way to get the **Riemann Mapping theorem** in Page 144 is to input

```
\newtheoremstyle{newnonum}{}{}
    {\itshape}{}{\bfseries}{.}{ }%
    {\thmname{#1}\thmnote{ (\mdseries #3)}}
\theoremstyle{newnonum}
\newtheorem{newRiemann}{Riemann Mapping Theorem}

\begin{newRiemann} Every open simply connected proper
  subset of $\mathbb{C}$ is analytically homeomorphic to
  the open unit disk in $\mathbb{C}$.
\end{newRiemann}
```

Then you can also produce **Cauchy's Theorem** in Page 143 by typing

```
\theoremstyle{newnonum}
\newtheorem{newCauchy}{Cauchy's Theorem}
```

```
\begin{newCauchy}[Third Version]If $G$ is a simply connected
  open subset of $\mathbb{C}$, then for every closed
  rectifiable curve $\gamma$ in $G$, we have
\begin{equation*}
 \int_\gamma f=0
\end{equation*}
\end{newCauchy}
```

The output will be exactly the same as that seen in Page 143. Now suppose you want to highlight certain theorems from other sources in your document, such as

> **Axiom 1 in [1].** *Things that are equal to the same thing are equal to one another.*

This can be done as follows:

```
\newtheoremstyle{citing}{}{}{\itshape}{}{\bfseries}
 {.}{ }{\thmnote{#3}}

\theoremstyle{citing}
\newtheorem{cit}{}

\begin{cit}[Axiom 1 in \cite{eu}]
   Things that are equal to the same thing are equal to
   one another.
\end{cit}
```

Of course, your *bibliography* should include the citation with *label* `eu`.

9.2.3. THERE IS MORE!

There are some more predefined features in amsthm package. In all the different examples we have seen so far, the *theorem number* comes after the *theorem name*. Some prefer to have it the other way round as in

> **9.2.1 Theorem** (Euclid). *The sum of the angles in a triangle is* 180°.

This effect is produced by the command `\swapnumbers` as shown below:

```
\swapnumbers
\theoremstyle{plain}
\newtheorem{numfirstthm}{Theorem}[section]
```

```
\begin{numfirstthm}[Euclid]
 The sum of the angles in a triangle is $180^\circ$
\end{numfirstthm}
```

Note that the `\swapnumbers` command is a sort of toggle-switch, so that once it is given, *all subsequent theorem-like statements* will have their numbers first. If you want it the other way for some other theorem, then give `\swapnumbers` again before its definition.

A quick way to suppress *theoremnumbers* is to use the `\newtheorem*` command as in

```
\newtheorem*{numlessthm}{Theorem}[section]

\begin{numlessthm}[Euclid]
 The sum of the angles in a triangle is $180^\circ$.
\end{numlessthm}
```

to produce

Euclid. *The sum of the angles in a triangle is* 180°.

Note that this could also be done by leaving out `#2` in the *custom-head-spec* parameter of `\newtheoremstyle`, as seen earlier.

We have been talking only about *theorems* so far, but Mathematicians do not live by theorems alone; they need *proofs*. The amsthm package contains a predefined `proof` environment so that the proof of a theorem-like statement can be enclosed within `\begin{proof} ... \end{proof}` commands as shown below:

```
\begin{thmsec}
  The number of primes is infinite.
\end{thmsec}

\begin{proof}
 Let $\{p_1,p_2,\dotsc p_k\}$ be a finite set of primes.
 Define $n=p_1p_2\dotsm p_k+1$. Then either $n$
 itself is a prime or has a prime factor. Now  $n$ is
 neither equal to nor is divisible by any of the primes
 $p_1,p_2,\dotsc p_k$ so that in either case, we get
 a prime different from $p_1,p_2,\dotsc p_k$. Thus no
 finite set of primes can include all the primes.
\end{proof}
```

to produce the following output

Theorem 9.2.3. *The number of primes is infinite.*

Proof. Let $\{p_1, p_2, \dots p_k\}$ be a finite set of primes. Define $n = p_1 p_2 \cdots p_k + 1$. Then either n itself is a prime or has a prime factor. Now n is neither equal to nor is divisible by any of the primes $p_1, p_2, \dots p_k$ so that in either case, we get a prime different from $p_1, p_2, \dots p_k$. Thus no finite set of primes can include all the primes. □

There is an optional argument to the `proof` environment which can be used to change the *proofhead*. For example,

```
\begin{proof}[\textsc{Proof\,(Euclid)}:]
 Let $\{p_1,p_2,\dotsc p_k\}$ be a finite set of primes.
 Define $n=p_1p_2\dotsm p_k+1$. Then either $n$
 itself is a prime or has a prime factor. Now  $n$
 is neither equal to nor is divisible by any of the
 primes $p_1,p_2,\dotsc p_k$ so that in either case, we
 get a prime different from $p_1,p_2,\dotsc p_k$.
 Thus no finite set of primes can include all the primes.
\end{proof}
```

produces the following

PROOF (EUCLID): Let $\{p_1, p_2, \dots p_k\}$ be a finite set of primes. Define $n = p_1 p_2 \cdots p_k + 1$. Then either n itself is a prime or has a prime factor. Now n is neither equal to nor is divisible by any of the primes $p_1, p_2, \dots p_k$ so that in either case, we get a prime different from $p_1, p_2 \dots p_k$. Thus no finite set of primes can include all the primes. □

Note that the end of a proof is *automatically* marked with a □ which is defined in the package by the command `\qedsymbol`. If you wish to change it, use `\renewcommand` to redefine the `\qedsymbol`. Thus if you like the original "Halmos symbol" ∎ to mark the ends of your proofs, include

```
\newcommand{\halmos}{\rule{1mm}{2.5mm}}
\renewcommand{\qedsymbol}{\halmos}
```

in the preamble to your document.

Again, the placement of the `\qedsymbol` at the *end* of the last line of the proof is done via the command `\qed`. The default placement may not be very pleasing in some cases as in

Theorem 9.2.4. *The square of the sum of two numbers is equal to the sum of their squares and twice their product.*

Proof. This follows easily from the equation

$$(x+y)^2 = x^2 + y^2 + 2xy$$

□

It would be better if this is typeset as

Theorem 9.2.5. *The square of the sum of two numbers is equal to the sum of their squares and twice their product.*

Proof. This follows easily from the equation

□ $$(x+y)^2 = x^2 + y^2 + 2xy$$

which is achieved by the input shown below:

```
\begin{proof}
 This follows easily from the equation
 \begin{equation}
  (x+y)^2=x^2+y^2+2xy\tag*{\qed}
 \end{equation}
 \renewcommand{\qed}{}
\end{proof}
```

For this trick to work, you must have loaded the package `amsmath` *without* the `leqno` option. Or, if you prefer

Proof. This follows easily from the equation

$$(x+y)^2 = x^2 + y^2 + 2xy \quad \square$$

Then you can use

```
\begin{proof}
 This follows easily from the equation
  \begin{equation*}
   (x+y)^2=x^2+y^2+2xy\qed
  \end{equation*}
 \renewcommand{\qed}{}
\end{proof}
```

9.3. HOUSEKEEPING

It is better to keep all `\newtheoremstyle` commands in the preamble than scattering them all over the document. Better still, you can keep them together with other customization in a personal `.sty` file and load it using the `\usepackage` command in the preamble. Also, within this `.sty` file, you can divide your `\newtheorem` commands into groups and preface each group with the appropriate `\theoremstyle`.

BIBLIOGRAPHY

[1] Euclid, *The Elements*, Greece 300 BC

CHAPTER X

SEVERAL KINDS OF BOXES

SIMPLY SPEAKING, TEX does everything with boxes. For example, when you type `{\Large boy}` in your document, you get boy after TEX has complied it *and* the printer driver has processed it, but what TEX produces internally is something like  or more precisely,  with instructions on what to put in each box. In other words, individual characters are put in *character boxes* which are stringed together horizontally to form *line boxes* which again are packed vertically into *paragraph boxes* and so on.

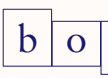
In this chapter we'll see how we can produce our own boxes in TEX.

10.1. BOXES — WITH FRAMES

How was the boxed "boy" of the first paragraph produced? The command is `\fbox` ('f' for *frame*). When we give a command of the form `\fbox{`*stuff*`}`, in our document, we get a frame around *stuff* in the output. Thus

```
but what \TeX\ produces is
{\Large\fbox{b}\fbox{o}\fbox{y}}
or more precisely,
```

produces

but what TEX produces is  or more precisely,

But this is not quite what we saw in the first paragraph, is it? Well, when you say `\fbox{`*stuff*`}`, TEX also puts some padding around *stuff*. The amount of space between *stuff* and the box

boundaries is controlled by the *length* `\fboxsep`. Thus to produce actual output of the first paragraph, we must type

```
but what \TeX\ produces is
{\setlength{\fboxsep}{0pt}\Large\fbox{b}\fbox{o}\fbox{y}}
or more precisely,
```

which gives

but what TEX produces is boy or more precisely,

The default value of `\fboxsep` is 3 pt and you can set it to any value using the `\setlength` command. Another length associated with an fbox is the thickness of the lines forming the frame. It is set by the length `\fboxrule` with default value 0.4 pt. Thus you can produce

an ordinary box and a custom box with extra-thick walls and extra padding inside

with

```
an \fbox{ordinary box} and a
\setlength{\fboxrule}{5pt}
\setlength{\fboxsep}{15pt}
\fbox{custom box} with extra-thick walls and extra padding
inside
```

But then how were the *empty* boxes in the first paragraph produced? There we use the TEX command `\phantom`: when you type `\phantom{`*stuff*`}`, TEX produces an empty box of the same size as *stuff*. Thus

```
or more precisely,
{\setlength{\fboxsep}{0pt}
 \Large
 \fbox{\phantom{b}}\fbox{\phantom{o}}\fbox{\phantom{y}}}
 with instructions on what to put in each
 box.
```

gives

or more precisely,  with instructions on what to put in each box.

Not only individual characters, but words and sentences can also be put in an fbox. Thus we can produce

On the table was a strip of paper with a single sentence:

This sentence is false

Here was a paradox: if the sentence is true, it's false and if it's false, it's true.

by simply enclosing the sentence to be framed in an `\fbox` as:

```
On the table was a strip of paper with a single
sentence:
\begin{center}
  \fbox{This sentence is false}
\end{center}
Here was a paradox: if the sentence is true, it's
false and if it's false, it's true.
```

But two things must be noted here:

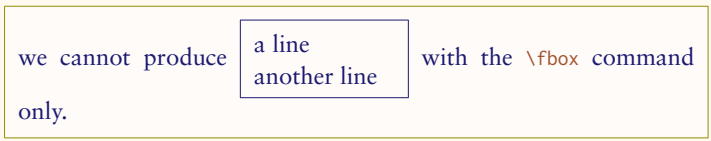
- TEX treats an fbox (or any other box for that matter) as a single *indivisible unit*.
- The text inside an fbox (or certain other boxes) is typeset in a *single line*

In practical terms, the first fact means that you cannot break an fbox across lines. So,

you cannot produce a broken fbox in TEX

Well, at least not with the `\fbox` command. This also means that even if the text inside the fbox is longer than the `\textwidth` TEX will try to put in a single line, with a resulting `Overfull \hbox`.

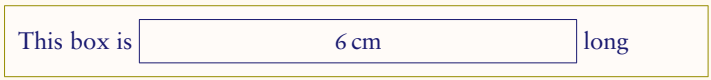
The practical implication of the second fact is that we cannot put two (or more) lines of text in an fbox so that



Because of these limitations, fboxes are called LR boxes ("LR" for Left-Right) in TEX parlance. There are other LR boxes which we'll now consider.

10.2. FRAMES OF ANY LENGTH

Another point to note about fboxes is that the command `\fbox` produces only a box just wide enough to contain the enclosed text (with `\fboxsep` on left and right, of course). Thus you cannot directly produce something like



with `\fbox`.

What we need is a mechanism to specify the width of an fbox. This is provided in the command `\framebox`. Thus the above example is typeset by the input

```
This box is
\framebox[6cm]{6\,cm}
long
```

Well, not quite. The above input will produce a box of length 6 cm plus 6pt. (Remember `\fboxsep`?). So, if we want a box *exactly* 6 cm long (why ever do we need that?) , we'll have to `\usepackage{calc}` and go

```
This box is
\framebox[\dimexpr(6cm - 6pt)]{6\,cm}
long
```

Note that the length specification is *optional*—if we don't specify any length, then we get an fbox, that is a box wide enough to contain the enclosed text. Also, if the length specified is shorter than the text, then the contents will spill out of the box (and over adjoining text as well) as in

if you try

```
No use crying over \framebox[0.6cm]{spilled} text
```

In a framebox, we can also specify the *position* of the enclosed text (relative to the box) in addition to the width of the enclosing box. Thus the full syntax of the command is

\fbox[*width*][*position*]{*text*}

The possible values of the *position* parameter are `l` for text flush with (no, not joy) the *left* edge of the box, `c` for text *centered* inside the box (this is the default placement, if you don't specify any position), `r` for text right up against the *right* edge of the box and finally (you didn't think there'd be one more, eh?) `s` for text *stretched* between the edges of the box (it's not really the text that's stretched, but any stretchable *space* it contains). For example

```
\framebox[3cm][l]{flushleft text}\\
\framebox[3cm][c]{centered text}\\
\framebox[3cm][r]{flushright text}\\
\framebox[3cm][s]{stretched text}
```

flushleft text

centered text

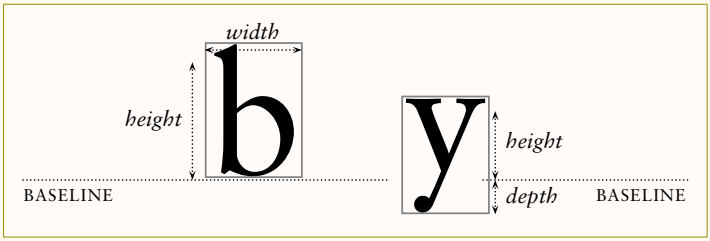
flushright text

stretched text

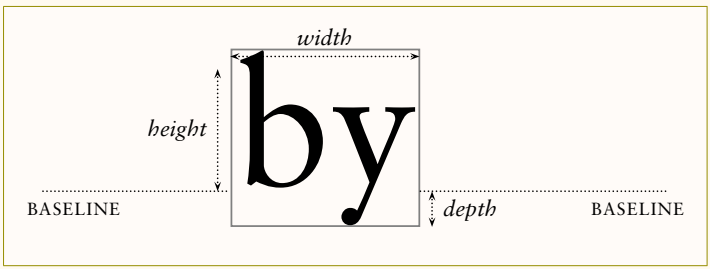
10.3. SPECIAL LENGTHS

Instead of specifying The width of a framebox (and other boxes allowing a width specification. which we'll soon see) can also be specified relatively in terms of the *natural dimensions* of the text inside, instead of in absolute units. This requires some explanation.

As we mentioned at the beginning, TEX reduces each character into a box. Every box has a *width* (no explanation needed), *height* (which is its height *above the baseline*) and *depth* (which is the distance it extends below the baseline) as shown below for two character boxes:



Thus the "b" box has 0 depth, while the "y" box a positive depth. Note that when these are joined to form the word "by", the resulting box has positive depth, equal to the depth of "y"



Let's now return to our discussion on setting the width of frameboxes in terms of the dimensions of the enclosed text. We can use the special commands `\width`, `\height` and `\depth` to refer to the width, height and depth of the enclosed text, in specifying the width of a framebox; we also have `\totalheight` which is the sum of `height` and `\depth`. Thus we can say `\fbox[2\width]{`*text*`}` to produce a framebox of width twice that of *text* as in the example below:

```
\framebox{Text}\\
\framebox[2\width][l]{Text}
```

Text

Text

Note that `\width` refers to the width of *the enclosed text* only, so that the width of the lower box in the above example is not equal to double the width of the upper box. The width of a box includes two `\fboxsep` and two `\fboxrule` also. So, if we want a box of double the width, we must add these lengths also to the width. Using the package calc we can do this as in the example below. (Here, the numbers indicate he width of each box, as provided by TEX itself.)

```
Text\\[5pt]
\framebox{Text}\\[5pt]
\framebox[2\width][l]{Text}\\[5pt]
\framebox[\dimexpr(2\width
          + 4\fboxsep
          + 4\fboxrule)][l]{Text}
```

Text 19.12973pt

Text 25.92972pt

Text 38.25946pt

Text 52.15923pt

It's worth reiterating that these commands such as `\width` have meaning only within the width specification of a `\framebox` (or in certain dimension specifications of some of the other boxes we are yet to see) and cause error messages outside these contexts.

10.4. BOXES WITHOUT FRAMES

LaTeX also has commands to generate boxes without frames. The command

`\mbox{`*text*`}`

puts *text* in a LR box of width equal to that of *text* and the command

`\makebox[`*width*`][`*positon*`]{`*text*`}`

puts *text* at the location specified by *position* (`l` for left, `c` for center and `r` for right) within a box of width specified by *width*, just like fbox and framebox, but *without* frames.

Now what good are boxes without frames, you may ask. Well, there are several uses. For one thing, TeX's treatment of boxes as individual units and consequent refusal to break them across lines can be turned to advantage. Consider the following output

Saltpeter is made by mixing carbon, sulphur and potassium nitrate in the ratio 3 : 2 : 1. To make 1.2 kg of saltpeter, how many grams of each chemical is required?

(We'll see later in this chapter, how we can create such boxed paragraphs.) Here the narrow line width specification has forced TeX to break the ratio 3 : 2 : 1 (which it usually doesn't). Now the same input with 3 : 2 : 1 put in an `\mbox` as `\mbox{$3:2:1$}` produces the output below:

Saltpeter is made by mixing carbon, sulphur and potassium nitrate in the ratio 3 : 2 : 1. To make 1.2 kg of saltpeter, how many grams of each chemical is required?

Also, zero width mboxes are useful to put text sticking out of the margin as in

	Class	Cumulative Frequency
	10-20	15
	20-30	27
	30-40	45
	40-50	70
median→	50-60	100
	60-70	116
	70-80	126

which is produced by

```
\begin{center}
  \renewcommand{\arraystretch}{1.5}
  \begin{tabular}{|c|@{}p{0.6cm}@{}p{0.6cm}@{}p{0.6cm}@{}|}
    \hline
    Class & \multicolumn{3}{m{1.8cm}|}%
            {\centering Cumulative \\
                        Frequency}
                      \tabularnewline
    \hline
    10-20 & & \raggedleft  15 & \\
    20-30 & & \raggedleft  27 & \\
    30-40 & & \raggedleft  45 & \\
    40-50 & & \raggedleft  70 & \\
    \makebox[0cm][r]%
    {\textit{median}$\rightarrow$\hspace{\tabcolsep}}%
    50-60 & & \raggedleft 100 & \\
```

```
    60-70 & & \raggedleft 116 & \\
    70-80 & & \raggedleft 126 & \\
    \hline
  \end{tabular}
\end{center}
```

(By the way, why the *empty* columns in the above `tabular`?)

10.5. MULTILINE BOXES

We have noted that the boxes considered so far can contain only one line of text We next consider various methods of boxing texts of more than one line.

One way to put multiline text inside a box is to use the `\parbox` command. We must also specify the width of the box. (Note that the width specification is *optional* for the boxes considered so far, but for `\parbox`, it is *mandatory*.) Thus you can produce

We can put a paragraph (that is, multiple lines) of text inside a box using the `\parbox` command.

by

```
We can put
\quad\parbox{3cm}{a paragraph (that is,  multiple lines)
of text inside a box}\quad
using the \parbox command.
```

Note that here, the *middle* of the parbox is aligned with line it's on. There's an optional argument to the `\parbox` command which changes this behavior. We type

`\parbox[`*position*`]{`*width*`}{`*text*`}`

to put *text* in a parbox of length *width* and position the box with its top or bottom (instead of the middle) aligned with the line of text where it occurs, using `t` or `b` as the value of *position*. Thus

```
We can put
\quad\parbox[t]{3cm}{a paragraph of text inside a box
with its \emph{top} aligned with the external line
of text}\quad using the \parbox[t] command.
```

gives

We can put a paragraph of text inside a box with its *top* aligned with the external line of text using the `\parbox[t]` command.

and

```
We can put
\quad\parbox[b]{3cm}{a paragraph of text inside a box with
its \emph{bottom} aligned with the external line of text}\quad
using  the \parbox[b] command
```

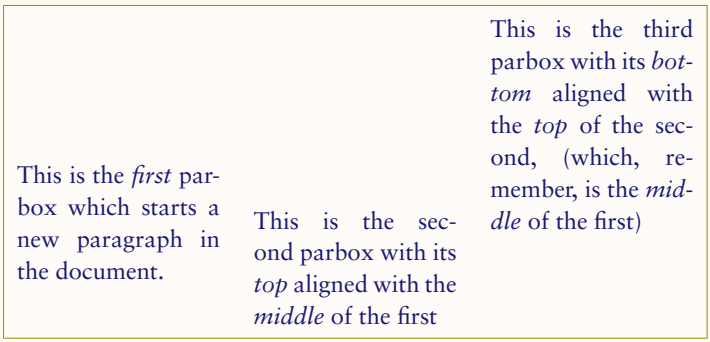
gives

We can put a paragraph of text inside a box with its *bottom* aligned with the external line of text using the `\parbox[b]` command.

Note that the positioning argument is meaningful only when there is an external line of reference for the parbox; in other words, only when the box is part of a paragraph of text. If the parbox is at the beginning of such a paragraph, the vertical positioning is determined by subsequent elements of that paragraph (which may be a single line of text or another parbox). If the parbox is the sole entry in a paragraph, then the positioning argument is irrelevant. The following string of parboxes may make this clear:

```
\parbox{0.3\textwidth}{This is the \emph{first} parbox
  which starts a new paragraph in the document.}\hfill
\parbox[t]{0.3\textwidth}{This is the second parbox
  with its \emph{top} aligned with the \emph{middle}
  of the first}\hfill
\parbox[b]{0.3\textwidth}{This is the third parbox
  with its \emph{bottom} aligned with the \emph{top}
  of the second (which, remember, is the \emph{middle}
  of the first)}
```

which produces



Confused? Juggle with these positioning arguments till you get the hang of it.

The `\parbox` command also allows us to specify the *height* of the box and the *position* of the text inside. Thus the form of this command in full is

`\parbox[`*position*`][`*height*`][`*inner position*`]{`*width*`}{`*text*`}`

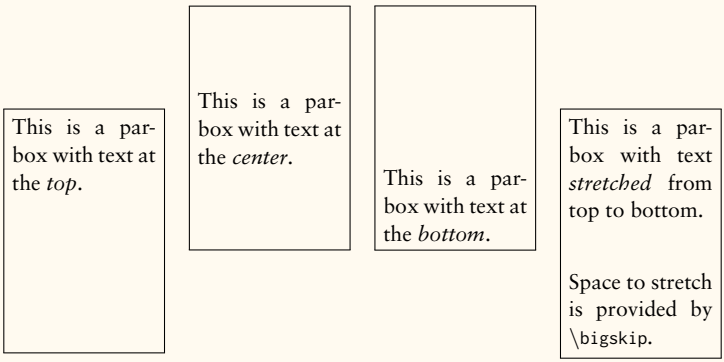
where *height* is the height of the box and *inner position* is where the *text* is to be placed *within* the box. It can be `t` for *top* of the box, `c` for *center* of the box or `b` for *bottom* of the box. There's also the `s` option which *stretches* the text from the to the bottom of the box, provided the text contains some stretchable space (for example, in the form of some `\vspace` command). For example, the input

```
\fbox{\parbox[t][3cm][t]{0.2\textwidth}{This is a parbox
    with text at the \emph{top}.}}
    \hfill
\fbox{\parbox[c][3cm][c]{0.2\textwidth}{This is a parbox  with
    text at the \emph{center}.}}
    \hfill
\fbox{\parbox[c][3cm][b]{0.2\textwidth}{This is a parbox with
    text at the \emph{bottom}.}}
    \hfill
\fbox{\parbox[t][3cm][s]{0.2\textwidth}{This is a parbox with
    text \emph{stretched} from  top to bottom.

    \bigskip
```

```
    Space to stretch is provided by
    \textbackslash\texttt{bigskip}.}}
```

produces



Here, we've used `\fbox` to show the position of the text within the box clearly. (It also shows we can nest the box commands.)

Another way to create multiline boxes is to use the *environment* minipage. The syntax is

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

Every example we've given in this section can also be achieved with the `minipage` environment instead of the `\parbox` command. In fact, the `minipage` environment, as the name suggests, complete micro version of a page and can even contain its own footnotes.

10.6. BLACK BOXES AND INVISIBLE LINES

LaTeX has also a mechanism to produce solid black boxes. the command to use is `\rule` whose general syntax is

```
\rule[lift]{width}{height}
```

This produces a solid black rectangle of width *width* and height *height* and places it at a distance of *lift* above the baseline. For example,

```
This is a black rectangle 1\,cm wide  and 3\,mm
high \rule{1cm}{3mm}. This is such a rectangle 5\,pt
```

```
above the baseline \rule[5pt]{1cm}{3mm} and this is
such a rectangle 5\,pt below the baseline
\rule[-5pt]{1cm}{3mm}
```

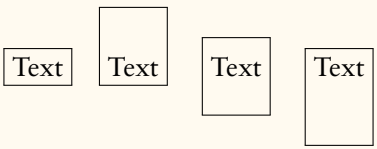
gives

This is a black rectangle 1 cm wide and 3 mm high  . This is such a rectangle 5 pt above the baseline  and this is such a rectangle 5 pt below the baseline 

An interesting possibility is a `\rule` of *zero* width. This creates an invisible line of height *height* which is called a *strut* in TEX parlance. Such a construct can be used to produce a LR box of desired height or depth. Consider the following example:

```
\fbox{Text}\quad\fbox{\rule{0mm}{8mm}Text}
\quad\fbox{\rule[-4mm]{0mm}{8mm}Text}
\quad\fbox{\rule[-8mm]{0mm}{8mm}Text}
```

which gives



What if we want the text to be at the center of the larger box? We can use the fact that a parbox is by default centered with respect to an external line of text. So,

```
\fbox{\parbox{0mm}{\rule[-8mm ]{0mm}{8mm}}Text}
```

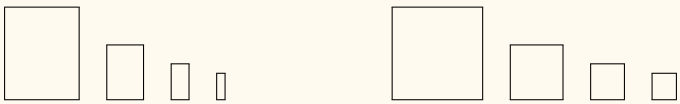
produces

Struts are also useful to draw blank rectangles and squares as in the example below:

```
\framebox[1cm]{\rule{0cm}{1cm}}\quad
\framebox[0.5cm]{\rule{0cm}{0.5cm}}\quad
\framebox[0.25cm]{\rule{0cm}{0.25cm}}\quad
```

```
\framebox[0.125cm]{\rule{0cm}{0.125cm}}
\hspace{2cm}
\framebox[\dimexpr(1cm + 2\fboxsep)]{\rule{0cm}{1cm}}\quad
\framebox[\dimexpr(0.5cm + 2\fboxsep)]{\rule{0cm}{0.5cm}}\quad
\framebox[\dimexpr(0.25cm + 2\fboxsep)]{\rule{0cm}{0.25cm}}\quad
\framebox[\dimexpr(0.125cm + 2\fboxsep)]{\rule{0cm}{0.125cm}}
```

gives



10.7. MOVING BOXES AROUND

We can move LR boxes ups or down using the `\raisebox` command. Thus

`\raisebox{`*lift*`}{`*text*`}`

creates an `\mbox` with *text* and places it above the current baseline at a height *lift*, if *lift* is positive. If *lift* is negative, the box is placed below the baseline at this distance. For example, we can produce

convex concave

by

```
\Large
c\raisebox{2pt}{o}\raisebox{4pt}{n}%
\raisebox{4pt}{v}\raisebox{2pt}{e}x
\hspace{2cm}
c\raisebox{-2pt}{o}\raisebox{-4pt}{n}%
\raisebox{-4pt}{c}\raisebox{-4pt}{a}\raisebox{-2pt}{v}e
```

Now TEX leaves a certain amount of space between lines of text and in the normal course of things, this space (named `\baselineskip`) is the same for any two adjacent lines. But when a line contains text that is *raised* or *lowered* using the `\raisebox` command, this interline space is automatically adjusted by TEX to accommodate the displaced boxes, as seen in this paragraph.

In fact, thew above paragraph is produced by

```
Now \TeX\ leaves a certain amount of space between
lines of text and in the normal course of things,
this space (named "\baselineskip") is the same for
any two adjacent lines.  But when a line contains
text that is
\raisebox{10pt}{\textit{raised}}
or
\raisebox{-10pt}{\textit{lowered}}
using the "\raisebox" command, this
interline space is automatically adjusted by
\TeX\ to accomodate the displaced boxes, as
seen in this paragraph.
```

Note that the amount of extra space provided to accommodate raised or lowered boxes depends on the (natural) height and depth of the displaced text. The command `\raisebox` also has a pair of optional arguments which control the height and depth of the box containing the raised text so that we can control the amount of extra interline space. More precisely,

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

creates an `\mbox` of height *height* and depth *depth* containing the text and places above or below the current baseline at a distance specified by *lift*. As an illustration, the input for the above example changed to read

```
\TeX\ leaves a certain amount of space between
lines of text and in the normal course of
things, this space (named "\baselineskip") is the
same for any two adjacent lines.  But when a
line contains text that is
 \raisebox{10pt}[20pt][0pt]{\textit{raised}}
or
 \raisebox{-10pt}[0pt][15pt]{\textit{lowered}}
using the "\raisebox" command, this
interline space is automatically adjusted by
\TeX\ to accomodate the displaced boxes,
as seen in this paragraph.
```

gives the output

TEX leaves a certain amount of space between lines of text and in the normal course of things, this space (named `\baselineskip`) is the same for any two adjacent lines. But when a line contains text that is *raised* or *lowered* using the `\raisebox` command, this interline space is automatically adjusted by TEX to accommodate the displaced boxes, as seen in this paragraph.

All these are available by default in LATEX. There are extra packages which allow further manipulation of boxes such as rotation (available in the rotating package) or scaling boxes (available in the graphicx package). These are matters for discussion elsewhere.

CHAPTER XI

FLOATS

11.1. THE figure ENVIRONMENT

FIGURES are really problematical to present in a document because they never split between pages. This leads to bad page breaks which in turn leave blank space at the bottom of pages. For fine-tuning that document, the typesetter has to adjust the page breaks manually.

But LaTeX provides floating figures which automatically move to suitable locations. So the positioning of figures is the duty of LaTeX.

11.1.1. CREATING FLOATING FIGURES

Floating figures are created by putting commands in a `figure` environment. The contents of the figure environment always remains in one chunk, floating to produce good page breaks. The following commands put the graphic from `figure.eps` inside a floating figure:

```
\begin{figure}
\centering
\includegraphics{figure.eps}
\caption{This is an inserted EPS graphic}
\label{fig1}
\end{figure}
```

Features

- The optional `\label` command can be used with the `\ref`, and `\pageref` commands to reference the caption. The `\label` command must be placed immediately *after* the `\caption`.

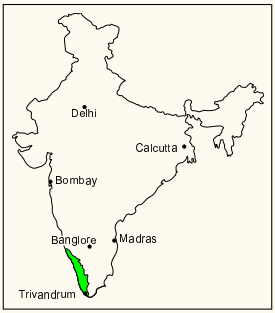


Figure 11.1: This is an inserted EPS graphic

- If the figure environment contains no `\caption` commands, it produces an unnumbered floating figure.
- If the figure environment contains multiple `\caption` commands, it produces multiple figures which float together. This is useful in constructing side-by-side graphics or complex arrangements.
- A list of figures is generated by the `\listoffigures` command.
- By default, the caption text is used as the caption and also in the list of figures. The caption has an optional argument which specifies the list-of-figure entry. For example,

  `\caption[`*List Text*`]{`*Caption Text*`}`

  causes "Caption Text" to appear in the caption, but "List Text" to appear in the list of figures. This is useful when using long, descriptive captions.
- The figure environment can only be used in *outer paragraph mode*, preventing it from being used inside any box (such as parbox or minipage).
- Figure environments inside the paragraphs are not processed until the end of the paragraph. For example:

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

11.1.2. FIGURE PLACEMENT

The `figure` environment has an optional argument which allows users to specify possible figure locations. The optional argument can contain any combination of the letters: `h`, `t`, `b`, `p`.

- `h` Place the figure in the text where the figure command is located. This option cannot be executed if there is not enough room remaining on the page.
- `t` Place the figure at the top of the page.
- `b` Place the figure at the bottom of a page.
- `p` Place the figure on a page containing only floats.

If no optional arguments are given, the placement options default to `[tbp]`.

When we input a float, LaTeX will read that float and hold it until it can be placed at a better location. Unprocessed floats are those which are read by LaTeX but have not yet been placed on the page. Though the float-placing is done by LaTeX, sometimes the user has to invoke commands to process unprocessed floats. Following commands will do that job:

`\clearpage` This command places unprocessed floats and starts a new page.

`\FloatBarrier` This command causes all unprocessed floats to be processed. This is provided by the `placeins` package. It does not start a new page, unlike `\clearpage`.

Since it is often desirable to keep floats in the section in which they were issued, the `section` option

```
\usepackage[section]{placeins}
```

redefines the `\section` command, inserting a `\FloatBarrier` command before each section. Note that this option is very strict. This option does not allow a float from the previous section to appear at the bottom of the page, since that is after the start of a new section.

The `below` option

```
\usepackage[below]{placeins}
```

is a less-restrictive version of the `section` option. It allows floats to be placed after the beginning of a new section, provided that some of the previous section appears on the page.

\afterpage/\clearpage The afterpage package provides the \afterpage command that executes a command at the next naturally-occurring page break.

Therefore, using \afterpage{\clearpage} causes all unprocessed floats to be cleared at the next page break. \afterpage{\clearpage} is especially useful when producing small float-page figures.

11.1.3. CUSTOMIZING FLOAT PLACEMENT

The following style parameters are used by LaTeX to prevent awkward-looking pages which contain too many floats or badly-placed floats.

Float placement counters

\topnumber The maximum number of floats allowed at the top of a text page (the default is 2).

\bottomnumber The maximum number of floats allowed at the bottom of a text page (the default is 1).

\totalnumber The maximum number of floats allowed on any one text page (the default is 3).

These counters prevent LaTeX from placing too many floats on a text page. These counters do not affect float pages. Specifying a ! in the float placement options causes LaTeX to ignore these parameters. The values of these counters are set with the \setcounter command. For example,

```
\setcounter{totalnumber}{2}
```

prevents more than two floats from being placed on any text page.

Figure fractions

The commands given below control what fraction of a page can be covered by floats (where "fraction" refers to the height of the floats divided by \textheight). The first three commands pertain only to text pages, while the last command pertains only to float pages. Specifying a ! in the float placement options causes LaTeX to ignore the first three parameters, but \floatpagefraction is always used. The value of these fractions are set by \renewcommand. For example,

```
\renewcommand{\textfraction}{0.3}
```

\textfraction The minimum fraction of a text page which must be occupied by text. The default is 0.2, which prevents floats from covering more than 80% of a text page.

\topfraction The maximum fraction of a text page which can be occupied by floats at the top of the page. The default is 0.7, which prevents any float whose height is greater than 70% of `\textheight` from being placed at the top of a page.

\bottomfraction The maximum fraction of a text page which can be occupied by floats at the bottom of the page. The default is 0.3, which prevents any float whose height is greater than 40% of `\textheight` from being placed at the bottom of a text page.

\floatpagefraction The minimum fraction of a float page that must be occupied by floats. Thus the fraction of blank space on a float page cannot be more than `1-\floatpagefraction`. The default is 0.5.

11.2. THE table ENVIRONMENT

With the *box* elements already explained in the previous chapter, it would be possible to produce all sorts of framed and unframed tables. However, LaTeX offers the user far more convenient ways to build such complicated structures.

11.2.1. CONSTRUCTING TABLES

The environments `tabular` and `tabular*` are the basic tools with which tables can be constructed. The syntax for these environments is:

```
\begin{tabular}[pos]{cols} rows \end{tabular}
\begin{tabular*}{width}[pos]{cols} rows \end{tabular*}
```

Both the above environments actually create a minipage. The meaning of the above arguments is as follows:

pos Vertical positioning arguments (see also the explanation of this argument for parboxes). It can take on the values:

- **t** The top line of the table is aligned with the baseline of the current external line of text.
- **b** The bottom line of the table is aligned with the external baseline.

With no positioning argument given, the table is centered on the external baseline.

width This argument applies only to the `tabular*` environment and determines its overall width. In this case, the *cols* argument must contain the `@`-expression (see below) `@{\extracolsep{\fill}}` somewhere after the first entry. For the other two environments, the total width is fixed by the textual content.

cols The column formatting argument. There must be an entry for every column, as well as possible extra entries for the left and right borders of the table or for the inter-column spacing. The possible *column formatting symbols* are:

- `l` The column contents are left justified.
- `c` The column contents are centered.
- `r` The column contents are right justified.
- `{wd}` The text in this column is set into lines of width *wd* and the top line is aligned with the other columns. In fact, the text is set in a parbox with the command `\parbox[t]{wd}{column text}`.
- `*{num}{cols}` The column format contained in *cols* is reproduced *num* times, so that `*{3}{|c|}` is the same as `|c|c|c|`.

The available formatting symbols for right and left borders and for the inter-column spacing are:

- `|` Draws a vertical line.

`||` Draws two vertical lines next to each other.

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

`@`-expression removes the inter-column spacing that is automatically put between each pair of columns. If white space is needed between the inserted text and the next column, this must be explicitly included with `\hspace{ }` within the text of the `@`-expression. If the inter-column spacing between two particular columns is to be something other than the standard, this may be easily achieved by placing `@{\hspace{`*wd*`}}` between the appropriate columns in the formatting argument. This replaces the standard inter-column spacing with the width *wd*.

An `\extracolsep{`*wd*`}` within an `@`-expression will put extra spacing of amount *wd* between all the following columns, until countermanded by another `\extracolsep` command. In contrast to the standard spacing, this additional spacing is not removed by later `@`-expression. In the `tabular*` environment, there must be a command `@{\extracolsep\fill}` somewhere in the column format so that all the subsequent inter-column spacing can stretch out to fill the predefined table width.

If the left or right borders of the table do not consist of a vertical line, a spacing equal to half the normal inter-column spacing is added there. If this spacing is not required, it may be suppressed by including an empty `@`-expression `@{}` at the beginning or end of the column format.

rows Contain the actual entries in the table, each horizontal row being terminated with `\\`. These rows consist of a sequence of column entries separated from each other by the `&` symbol. Thus each row in the table contains the same number of column entries as in the column definition *cols*. Some entries may be empty. The individual column entries are treated by LaTeX as though they were enclosed in braces `{ }`, so that any change in type style or size are restricted to that one column.

`\hline` This command may only appear before the first row or immediately after a row termination `\\`. It draws

a horizontal line the full width of the table below the row that was just ended, or at the top of the table if it comes at the beginning. Two `\hline` commands together draw two horizontal lines with a little space between them.

`\cline{`*n* − *m*`}`
: This command draws a horizontal line from the left side of column *n* to the right side of column *m*. Like `\hline`, it may only be given just after a row termination `\\`, and there may be more than one after another. The command `\cline{`*1-3*`} \cline{`*5-7*`}` draws two horizontal lines from column 1 to 3 and from column 5 to 7, below the row that was just ended. In each case, the full column widths are underlined.

`\vline`
: This command draws a vertical line with the height of the row at the location where it appears. In this way, vertical lines that do not extend the whole height of the table may be inserted with a column.

`\multicolumn{num}{col}{text}`
: This command combines the following *num* columns into a single column with their total width including inter-column spacing. The argument *col* contains exactly one of the positioning symbols `l`, `r`, `c`, with possible `@`-expressions and vertical lines. A value of 1 may be given for *num* when the positioning argument is to be changed for that column in one particular row.

  In this context, a 'column' starts with a positioning symbol `l`, `r`, or `c` and includes everything up to but excluding the next one. The first column also includes everything before the first positioning symbol. Thus `c@{}rl` contains three columns: the first is `c@{}`, the second `r`, and the third `l`.

11.2.2. TABLE STYLE PARAMETERS

There are a number of style parameters used in generating tables which LaTeX sets to standard values. These may be altered by the user, either globally within the preamble or locally inside an environment. They should not be changed within the

tabular environment.

- \tabcolsep is half the width of the spacing that is inserted between columns in the tabular and tabular* environments.
- \arrayrulewidth is the thickness of the vertical and horizontal lines within a table.
- \doublerulesep is the separation between the lines of a double rule.
- \arraystretch can be used to change the distance between the rows of a table. This is a multiplying factor, with a standard value of 1. A value of 1.5 means that the inter-row spacing is increased by 50%. A new value is set by redefining the parameter with the command:

  \renewcommand{\arraystretch}{*factor*}

  Following are the commands for changing the table style parameters that relate to dimensions:

  \setlength\tabcolsep{*dimen*}
  \setlength\arrayrulewidth{*dimen*}
  \setlength\doublerulesep{*dimen*}

11.2.3. EXAMPLE

Creating tables is much easier in practice than it would seem from the above list of formatting possibilities. This is best illustrated with an example.

The simplest table consists of rows and columns in which the text entries are either centered or justified to one side. The column widths, the spacing between the columns, and thus the entire width of the table are automatically calculated.

Sample Tabular		
col head	col head	col head
Left	centered	right
aligned	items	aligned
items	items	items
Left items	centered	right aligned

See the code that generated the table above.

```
\begin{tabular}{l|c|r|}
\hline
\multicolumn{3}{|c|}{Sample Tabular}\\  \hline
col head   & col head & col head\\  \hline
Left       & centered & right   \\\cline{1-2}
aligned    & items    & aligned \\\cline{2-3}
items      & items    & items   \\\cline{1-2}
Left items & centered & right aligned\\  \hline
\end{tabular}
```

The discussion on tables doesn't conclude with this chapter, instead more bells and whistles are to be discussed, such as long tables (tables that span multiple pages), how to repeat the column headings and special footlines in all multi-paged tables, color tables and also a few other embellishments, which the scientific community at large might require in their document preparation.

11.2.4. EXERCISE

Here is an exercise you can try.

Plan for TeX Users Group 2001–2003						
Project	No.		Name			
Year	2001		2002		2003	
	Rs.	US$	Rs.	US$	Rs.	US$
Internet costs						
Journal costs						
TeXLive production costs						
Signature				Authorization		

CHAPTER XII

GRAPHICS INCLUSION

The LaTeX typesetting system has limited capability of drawing geometric pictures and there are packages which extend this considerably. However, very often we use other tools to produce graphics such as graphs of experimental data or photographs. In this chapter, we discuss how such files can be included in LaTeX files. There are two different packages graphics and graphicx which can be used for this purpose. They have the same functionality, but different user interfaces. We discuss only the graphicx package since (we feel) it has a friendlier syntax.

12.1. THE GRAPHICX PACKAGE

12.1.1. LOADING THE PACKAGE

As mentioned above, the graphicx package provides commands to include external graphic files and also to transform them by scaling or rotating. To use the commands available in the package, we must *load* it using the command

```
\usepackage{graphicx}
```

In all the examples given below, we assume that this has been done and do not explicitly mention it every time.

One important thing to bear in mind is that LaTeX only *reserves* space in its output for the typeset graphics. It's the task of the *device driver* (such as dvips, xdvi, pdflatex etc. to perform the *actual inclusion* of the graphic and its *display* (either in print or in a previewer). So, the driver to be used must be specified as an option to the `\usepackage{graphicx}` command such as for example

```
\usepackage[dvips]{graphicx}
```

if you use the dvips driver or

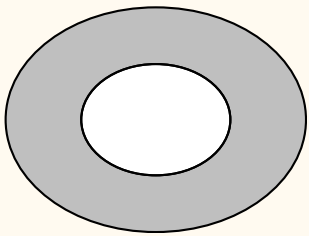
```
\usepackage[xetex]{graphicx}
```

if you use the xetex program that supports utf-8.

12.1.2. INCLUDING A GRAPHICS FILE

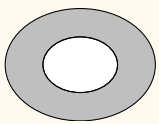
Suppose your working directory contains a graphic file ellann.eps containing the picture of an elliptic annulus which you want to include in a LaTeX file. (The package graphicx can handle graphic files of other formats other than EPS or "Encapsulated PostScript", but EPS files are the easiest to handle). We show below how you do it and the resulting output:

```
\includegraphics%
    {ellipse}
```



We can scale down the picture to half its original dimensions using the *key* scale as in the next example.

```
\includegraphics%
    [scale=0.5]%
     {ellann.eps}
```

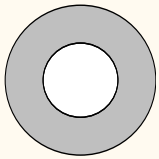


We can also make the picture fit into a rectangle of specified dimensions using the width and height keys as shown below:

```
\includegraphics%
   [width=4cm,%
     height=0.5cm]%
      {ellann.eps}
```

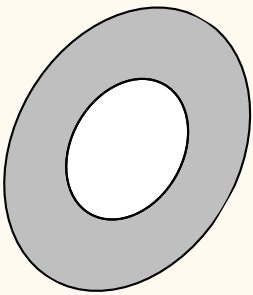
Note that by making the width and height equal, we can make the annulus circular.

```
\includegraphics%
   [width=2cm,%
     height=2cm]%
      {ellann.pdf}
```



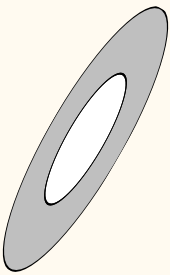
We can also rotate our picture by specifying the angle (in degrees), measured counter-clockwise. The key in this case is `angle` as in the example below:

```
\includegraphics%
    [angle=60]%
     {ellann.pdf}
```



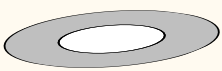
The fun really starts when you combine these commands. Look at this example:

```
\includegraphics%
  [width=4cm,%
    height=1cm,%
    angle=60]%
     {ellann.pdf}
```



Note that the keys are read from *left to right* so that in the above example, the picture is scaled (to fit into the specified rectangle) and *the scaled picture rotated*. Quite a different is obtained if we reverse the order of things, so that the picture is rotated and *the rotated image is scaled*:

```
\includegraphics%
  [angle=60,width=4cm,%
    height=1cm]%
     {ellann.pdf}
```

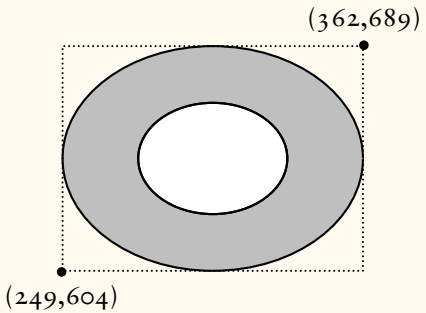


12.1.3. CLIPPING PICTURES

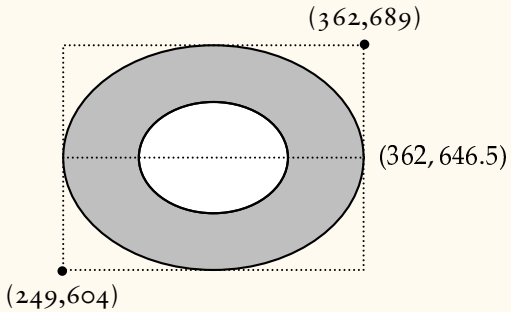
We can also "clip" a portion of the picture into our file. For this a little bit of computation is necessary. First note that the EPS file specifies a rectangle into which the whole graphic fits. This so called "BoundingBox" is specified by the co-ordinates of its lower left and upper right corners. Thus in the file `ellann.eps`, there is a line

```
%%BoundingBox: 249 604 362 689
```

which means that the ellipse is bounded by an invisible rectangle as shown below:

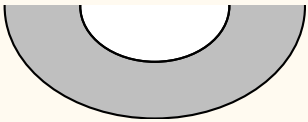


Suppose that we now want to use only the lower half of the ellipse in our document. First we must calculate the coordinates for bounding box of this portion. These can be easily found out to be (249,604) and (362,646.5) as shown below:



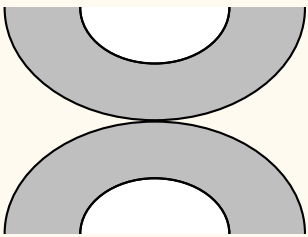
Now we can specify this bounding box as the value to the key `bb` in `\includegraphics` and use the key `clip` as shown below:

```
\includegraphics%
  [bb=249 604 362 646.5,%
    clip]%
    {ellann.eps}
```



Similarly (how?) you can clip the upper half of the ellipse; then if you want, you can stack up the two halves in "reverse" to get

```
\includegraphics%
  [bb=249 604 362 646.5,%
    clip]%
   {ellann.eps}\\[-1.253pt]
\includegraphics%
  [bb=249 646.5 362 689,%
     clip]%
     {ellann.eps}
```

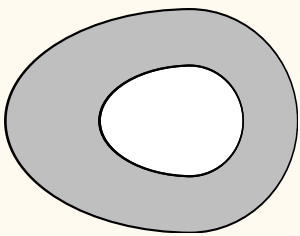


There are some more keys available, but we don't discuss them here. See the documentation `grfguide.dvi` available in most TEX installation for further details.

Again, you can create an "eggshell" from the original picture as follows

```
\includegraphics%
  [width=69.375pt,%
   height=85pt,%
   bb=249 604 304.5 689,%
   clip]%
  {ellann.eps}
  \hspace{-6pt}
  \includegraphics%
    [width=41.625pt,%
     height=85pt,%

    bb=304.5 604 362 689,%
     clip]%
    {ellann.eps}
```



PS (not the program, but an afterthought): the elliptic annulus shown in the examples above is actually drawn within TEX, using the indexpackage!=pstricks=pstricks package by Timothy

Van Zandt (more about this wonderful package in another *book*.) To generate it, use the code below:

```
\documentclass{article}
\usepackage{pstricks}

\begin{document}
\begin{center}
  \psset{unit=.5cm}
  \begin{pspicture}(-5,-5)(5,5)
   \psellipse[fillstyle=solid,fillcolor=lightgray](0,0)(4,3)
    \psellipse[fillstyle=solid](0,0)(2,1.5)
    \psellipse(0,0)(2,1.5)
  \end{pspicture}
\end{center}
\end{document}
```

This is LaTeX compiled and then the PostScript (the program—*not* an afterthought) file produced in the usual way. Then the utility `ps2eps` (available for GNU/Linux) is used to produce the EPS file to be included in the document.

12.1.4. GRAPHICS SEARCH PATH

By default, LaTeX looks for graphics files in any directory on the TeX search path. In addition to these directories, LaTeX also looks in any directories specified in the `\graphicspath` command. For example,

```
\graphicspath{{dir1/}{dir2/}}
```

tells LaTeX to look for graphics files also in `dir1/` and `dir2/`. For Macintosh, this becomes

```
\graphicspath{{dir1:}{dir2:}}
```

12.1.5. GRAPHICS EXTENSIONS

The `\DeclareGraphicsExtensions` command tells LaTeX which extensions to try if a file with no extension is specified in the `\includegraphics` command. For convenience, a default set of extensions is pre-defined depending on which graphics driver is selected. For example if `dvips` is used, the following graphics extensions (defined in `dvips.def`) are used by default

```
\DeclareGraphicsExtensions{.eps,.ps,.eps.gz,.ps.gz,.eps.Z}
```

With the above graphics extensions specified, `\includegraphics` *file* first looks for `file.eps`, then `file.ps`, then file `file.eps.gz`, etc. until a file is found. This allows the graphics to be specified with

```
\includegraphics{file}
```

instead of

```
\includegraphics{file.eps}
```

CHAPTER XIII

CROSS REFERENCES IN LaTeX

13.1. WHY CROSS REFERENCES?

CROSS REFERENCE is the technical term for quoting yourself. This is what you do when you say something like, "As I said earlier,...". More seriously, in a written article you may often have occasion to refer the reader to something mentioned earlier (or sometimes to something yet to be said) in the same document. Thus you may have explained a new term in the second section of your article and when you use this term again in the fourth section, it is a matter of courtesy to the reader to point to the explanation. Again, in a mathematics article, you may have to cite an earlier result in the proof of the current result.

Such cross referencing can be done by hand, but if you revise your document and insert some new sections (or theorems) then changing all cross references manually is no easy task. It is always better to automate such tedious tasks. (After all what's a computer for, if not to do such mundane jobs?)

13.2. LET LaTeX DO IT

The basic method of using cross references (see Section 13.1 for what we mean by cross reference) in LaTeX is quite simple. Suppose that somewhere in the second section of your article, you want to refer to the first section. You assign a *key* to the first section using the command

```
\section{sectionname}\label{key}
```

and at the point in the second section where the reference is to be made, you type the command

```
\ref{key}
```

Thus the reference "see Section 13.1..." in the first sentence of this section was produced by including the command `\label{intro}` in the command for the first section as

```
\section{Why cross references}\label{intro}
```

and the command `\ref{intro}` at the place of reference in the second section as

```
...(see Section \ref{intro} for...
```

Okay, the example is a bit silly, since the actual reference here is not *really* necessary, but you get the general idea, don't you? Incidentally, the `\label{`*key*`}` for a section need not be given immediately after the `\section{`*section name*`}`. It can be given anywhere within the section.

The first time you run LaTeX on a file named, say, `myfile.tex` containing cross references, the reference information is written in an auxiliary file named `myfile.aux` and at the end of the run LaTeX prints a warning

```
LaTeX Warning: There were undefined references.

LaTeX Warning: Label(s) may have changed.
            Rerun to get cross-references right.
```

A second run gets the references right. The same thing happens when you've changed the reference information in any way, say, by adding a new section.

Though the *key* in `\label{`*key*`}` can be any sequence of letters, digits or punctuation characters, it is convenient to use some mnemonic (such as `\label{`*limcon*`}` for a section entitled "Limits and Continuity" rather than `\label{`*sec@#*?!*`}`). Also, when you make a reference, it's better to type `~\ref{`*limcon*`}` (notice the *tie*?) than `\ref{`*limcon*`}` to prevent the possibility of the reference number falling off the edge as in "....see Section 13.1 for further details....".

In addition to sectioning commands such as `\chapter` or `\section`, reference can also be made to an `\item` entry in an `enumerate` environment, by attaching a `\label`. For example the input

```
In the classical \emph{syllogism}
```

```
  \begin{enumerate}[(1)]
  \item All men are mortal.\label{pre1}
  \item Socrates is a man.\label{pre2}
  \item So Socrates is a mortal.\label{con}
  \end{enumerate}
Statements (\ref{pre1}) and (\ref{pre2}) are the \emph{premises} and
 statement (\ref{con}) is the conclusion.
```

gives the following output

In the classical *syllogism*

(1) All men are mortal.
(2) Socrates is a man.
(3) So Socrates is a mortal.

Statements (1) and (2) are the *premises* and statement (3) is the conclusion.

You must be a bit careful about references to tables or figures (technically, "floats"). For them, the `\label` command should be given after the `\caption` command or in its argument, as in the example below:

```
\begin{table}[h]
\begin{center}
\setlength{\extrarowheight}{5pt}
\begin{tabular}{|c|c|c|c|}
\hline
Value of $x$ & 1 & 2 & 3\\
\hline
Value of $y$ & 1 & 8 & 27\\
\hline
\end{tabular}
\caption{Observed values of $x$ and $y$}\label{tabxy}
\end{center}
\end{table}
Two possible relations betweeen $x$ and $y$ satisfying
the data in Table~\ref{tabxy} are $y=x^3$ and
$y=6x^2-11x+6$
```

This produces the following output:

Two possible relations between x and y satisfying the data in Table 13.1 are $y = x^3$ and $y = 6x^2 - 11x + 6$

Value of x	1	2	3
Value of y	1	8	27

Table 13.1: Observed values of x and y

You can think of a `\caption` command within a `figure` or `table` environment as a sort of sectioning command within the environment. Thus you can have several `\caption` and `\label` pairs within a single `figure` or `table` environment.

You can also make *forward* references in exactly the same way by `\ref`-ing to the *key* of some succeeding `\label` such as "see Subsection 13.2.1 for a discussion of cross references in mathematics."

13.2.1. CROSS REFERENCES IN MATH

Mathematical documents abound in cross references. There are references to theorems and equations and figures and whatnot. The method of reference is exactly as before. Thus if you've defined `\newtheorem{theorem}{Theorem}[subsection]`, then after typing

```
\begin{theorem}\label{diffcon}
Every differentiable function is continuous
\end{theorem}
```

you get

13.2.1.1 Theorem. *Every differentiable function is continuous*

and you can type elsewhere in the document

```
The converse of Theorem~\ref{diffcon} is false.
```

to get

The converse of Theorem 13.2.1.1 is false.

References can be made to equations as in the following examples:

```
\begin{equation}\label{sumsq}
  (x+y)^2=x^2+2xy+y^2
\end{equation}
```

```
Changing $y$ to $-y$ in Equation~(\ref{sumsq}) gives the
following
```

(13.1) $(x+y)^2 = x^2 + 2xy + y^2$

Changing y to $-y$ in Equation (13.1) gives the following

If you load the package amsmath, you can use the command `\eqref` instead of `\ref` to make a reference to an equation. This automatically supplies the parantheses around the equation number and provides an italic correction before the closing parenthesis, if necessary. For example,

```
Equation \eqref{sumsq} gives the following ..........
```

produces

Equation 13.1 gives the following .....

References can be made to individual equations in multiline displays of equations produced by such environments as `align` or `gather` (defined in the amsmath package). The `\label` command can be used within such a structure for subnumbering as in the example below:

```
\begin{align}
 (x+y)^2&=x^2+2xy+y^2\label{sum}\\
 (x-y)^2&=x^2-2xy+y^2\tag{\ref{sum}a}
\end{align}
```

(13.2) $(x+y)^2 = x^2 + 2xy + y^2$

(13.2a) $(x-y)^2 = x^2 - 2xy + y^2$

13.3. POINTING TO A PAGE—THE PACKAGE VARIOREF

In making a reference to a table or an equation, it is more to convenient (for the reader, that is) to give the page number of the reference also. The command

```
\pageref{key}
```

typesets the number of the page where the command, namely, \label{*key*} was given. Thus for example

```
see Table~\ref{tabxy} in page~\pageref{tabxy}
```

in this document produces

see Table 13.1 in page 190

To avoid the tedium of repeated by typing

\ref{*key*} on page \pageref{*key*}

you can define the macro

```
\newcommand{\fullref}[1]{\ref{#1} on page~\pageref{#1}}
```

and use \fullref for such references. But the trouble is that at times the referred object and the reference to it fall on the same page (with TeX you never know this till the end) so that you get a reference to the page number of the very page you are reading, which looks funny. This can be avoided by using the package varioref. If you load this package by including \usepackage{varioref} in your preamble, then you can use the command

\vref{*key*}

to refer to an object you've marked with \label{*key*} elsewhere in the document. The action of \vref varies according to the page(s) where the referred object and the references are typeset by TeX in the final output.

(1) If the object and the reference are on the same page, \vref produces only a \ref suppressing \pageref so that only the number pointing to the object is typeset, without any reference to the page number.

(2) If the object and the reference are on different pages whose numbers differ by more than one, \vref produces both \ref and \pageref.

(3) If the object and the reference fall on pages whose numbers differ by one (that is, on successive pages), \vref produces \ref followed by the phrase "on the preceding page" or "on the following page" depending on whether the object or the reference occurs first. Moreover, in the next occurrence of \vref in a situation of the same type, the phrases

are changed to "on the next page" and the "page before" respectively.

This is the default behavior of \vref in the article documentclass. If the article class is used with the twoside option or if the documentclass book is used, then the behavior in Case (3) above is a bit different.

(1) If the object and the reference fall on the two sides of the same *leaf*, the behavior of \vref is as in (3) above.

(2) If the object and the reference fall on pages forming a double spread (that is, a page of even number followed by the next page), then \vref produces \ref followed by the phrase "on the facing page". Moreover, in the next occurence of \vref in a situation of the same type, the phrases are changed to "on the preceding page" and "on the next page" respectively.

The phrases used in the various cases considered above can be customized by redefining the commands used in generating them. For the article class without the twoside option, reference to the previous page uses the command \reftextbefore and reference to the next page uses \reftextafter. In the case of the article class with the twoside option or the book class, the commands \reftextfaceafter and \reftextfacebefore are used in the case of reference to a page in a double spread. The default definitions of these commands are given below. In all these, the two arguments of the command \reftextvario are phrases alternatively used in the repeated use of the reference as mentioned above.

```
\newcommand{\reftextbefore}
           {on the \reftextvario{preceding page}{page before}}
\newcommand{\reftextafter}
           {on the \reftextvario{following}{next} page}
\newcommand{\reftextfacebefore}
           {on the \reftextvario{facing}{preceding} page}
\newcommand{\reftextfaceafter}
           {on the \reftextvario{facing}{next}{page}}
```

You can customize the phrases generated in various situations by redefining these with phrases of your choice in the arguments of \reftextvario.

If you want to refer only to a page number using \varioref, you can use the command

```
\vpageref{key}
```

to produce the page number of the object marked with the command, \label{*key*}. The phrases used in the various special cases are the same as described above, except that when the referred object and the reference fall on the same page, either the phrase "on this page" or "on the current page" is produced. The command used to generate these is \reftextcurrent whose default definition is

```
\newcommand{\reftextcurrent}
           {on \reftextvario{this}{the current} page}
```

You can change the phrases "this" and "the current" *globally* by redefining this command. You can also make some *local* changes by using the two optional arguments that \vpageref allows. Thus you can use the command

```
\vpageref[same page phrase][other page phrase]{key}
```

to refer to the page number of the object marked with the command, \label{*key*}. The *same page phrase* will be used if the object and the reference fall on the same page and the phrase *other page phrase* will be used, if they fall on different pages. Thus for example, the command

```
see the \vpageref[above table][table~]{tabxy}
```

given in this document will produce

see the above table

if the reference occurs on the same page as Table 13.1 and

see the table on page 190

if they fall on different pages.

13.4. POINTING OUTSIDE—THE PACKAGE XR

Sometimes you may want to refer to something in a document other than the one you are working on. (This happens, for instance if you keep an article as separate files.) The package xr allows such external references.

If you want to refer to objects in a file named `other.tex` in your current document, load the package `xr` and set the external document as `other.tex` using the commands

```
\usepackage{xr}  \externaldocument{other}
```

in the preamble of the current document. Then you can use the `\ref` and `\pageref` to refer to anything that has been marked with the `\label` command in either the current document or `other.tex`. Any number of such external documents can be specified.

If the same *key* is used to mark different objects in two such documents, there'll be a conflict. To get over this, you can use the optional argument available in `\externaldocument` command. If you say

```
\externaldocument[a-]{other}
```

then a reference to `\label{key}` in `other.tex` could be made by `\ref{a-key}`. The prefix need not be `a-`; it can be any convenient string.

13.5. LOST THE KEYS? USE lablst.tex

One of the conveniences of using keys for cross references is that you need not keep track of the actual numbers, but then you'll have to remember the keys. You can produce the list of keys used in a document by running LaTeX on the file `lablst.tex`. In our system, we do this by first typing

```
latex lablst
```

LaTeX responds as follows:

```
*********************************
* Enter input file name
*   without the .tex extension:
*********************************

\lablstfile=
```

We type in the file name as `cref` which is the source of this document and is presented with another query.

```
**********************************************
* Enter document class used in file cref.tex
```

```
  *   with no options or extension:
  ********************************************

\lablstclass=
```

So we type `article`. And is asked

```
  ******************************************
  * Enter packages used in file cref.tex
  *   with no options or extensions:
  ******************************************

\lablstpackages=
```

Here only those packages used in the article which define commands used in section titles etc. need be given. So we type

```
  amsmath,array,enumerate
```

This produces a file `lablst.dvi` which can be viewed to see a list of keys used in the document.

Finally if your text editor is `GNU Emacs`, then you can use its `RefTeX` package to automate generation, insertion and location of keys at the editing stage.

CHAPTER XIV

FOOTNOTES, MARGINPARS, AND ENDNOTES

HE LaTeX typesetting system has facilities to typeset "inserted" text, such as footnotes, marginal notes, figures and tables. This chapter looks more closely at different kinds of notes.

14.1. FOOTNOTES

Footnotes are generated with the command

`\footnote{`*footnote_text*`}`

which comes immediately after the word requiring an explanation in a footnote. The text *footnote_text* appears as a footnote in a smaller typeface at the bottom of the page. The first line of the footnote is indented and is given the same footnote marker as that inserted in the main text. The first footnote on a page is separated from the rest of the page text by means of a short horizontal line.

The standard footnote marker is a small, raised number[1], which is sequentially numbered.

Footnotes produced with the `\footnote` command inside a **minipage** environment use the **mpfootnote** counter and are typeset at the bottom of the parbox produced by the **minipage**[2].

However, if you use the `\footnotemark` command in a `minipage` it will produce a footnote mark in the same style and sequence

[1] See how the footnote is produced: " ... raised number `\footnote{See how the footnote is produced: ... }`.

[2] With nested minipages, the footnote comes after the next `\endminipage` command, which could be at the wrong place.

as the main text footnotes—i.e., stepping the `mpfootnote` counter and using the `\thefootnote` command for the representation. This behavior allows you to produce a footnote inside your `minipage` that is typeset in sequence with the main text footnotes at the bottom of the page: you place a `\footnotemark` inside the `minipage` and the corresponding `\footnotetext` after it. See below:

Footnotes in a minipage are numbered using lowercase letters.[a]
This text references a footnote at the bottom of the page.[3]

[a] Inside minipage

```
\begin{minipage}{5cm}
  Footnotes in a minipage are
  numbered using lowercase
  letters.\footnote{Inside
  minipage}\par
  This text references a
  footnote at the bottom
  of the page.\footnotemark
\end{minipage}
\footnotetext{At bottom of page}
```

The footnote numbering is incremented throughout the document for the article class, where it is reset to 1 for each new chapter in the report and book classes.

14.1.1. FOOTNOTES IN TABULAR MATERIAL

Footnotes appearing inside tabular material are not typeset by standard LaTeX. Only `tabularx` and `longtable` environments will treat footnotes correctly. But footnotes used in these tables won't appear just following the tables, but would appear at the bottom of the page just like the footnotes used in the text. But in `longtable` you can place the footnotes as table notes by placing the longtable in a minipage. See below:

Table 14.1: PostScript type 1 fonts

Courier[a]	cour, courb, courbi, couri
Nimbus[b]	unmr, unmrs
URW Antiqua[b]	uaqrrc
URW Grotesk[b]	ugqp
Utopia[c]	putb, putbi, putr, putri

[a] Donated by IBM.
[b] Donated by URW GmbH.
[c] Donated by Adobe.

[3] At bottom of page.

```
\begin{minipage}{.47\textwidth}
 \renewcommand{\thefootnote}{\thempfootnote}
  \begin{longtable}{ll}
   \caption{PostScript type 1 fonts}\\
    Courier\footnote{Donated by IBM.}
      & cour,courb,courbi,couri \\
    Nimbus\footnote{Donated by URW GmbH.}
      & unmr, unmrs \\
    URW Antiqua\footnotemark[\value{mpfootnote}]
      & uaqrrc\\
    URW Grotesk\footnotemark[\value{mpfootnote}]
      & ugqp\\
    Utopia\footnote{Donated by Adobe.}
      & putb, putbi, putr, putri
  \end{longtable}
 \end{minipage}
```

You can also put your `tabular` or `array` environment inside a `minipage` environment, since in that case footnotes are typeset just following that environment. Note the redefinition of `\thefootnote` that allows us to make use of the `\footnotemark` command inside the `minipage` environment. Without this redefinition `\footnotemark` would have generated a footnote mark in the style of the footnotes for the main page.

```
\begin{minipage}{.5\linewidth}
 \renewcommand{\thefootnote}{\thempfootnote}
  \begin{tabular}{ll}
   \multicolumn{2}{c}{\bfseries PostScript type 1 fonts} \\
   Courier\footnote{Donated by IBM.}
      & cour,courb,courbi,couri \\
   Charter\footnote{Donated by Bitstream.}
      & bchb,bchbi,bchr,bchri\\
   Nimbus\footnote{Donated by URW GmbH.}
      & unmr, unmrs \\
   URW Antiqua\footnotemark[\value{mpfootnote}]
      & uaqrrc\\
   URW Grotesk\footnotemark[\value{mpfootnote}]
      & ugqp\\
   Utopia\footnote{Donated by Adobe.}
      & putb, putbi, putr, putri
 \end{tabular}
\end{minipage}
```

PostScript type 1 fonts

Courier[a]	cour, courb, courbi, couri
Charter[b]	bchb, bchbi, bchr, bchri
Nimbus[c]	unmr, unmrs
URW Antiqua[c]	uaqrrc
URW Grotesk[c]	ugqp
Utopia[d]	putb, putbi, putr, putri

[a] Donated by IBM.
[b] Donated by Bitstream.
[c] Donated by URW GmbH.
[d] Donated by Adobe.

Of course this approach does not automatically limit the width of the footnotes to the width of the table, so a little iteration with the `minipage` width argument might be necessary.

Another way to typeset table notes is with the package `threeparttable` by Donald Arseneau. This package has the advantage that it indicates unambiguously that you are dealing with notes inside tables and, moreover, it gives you full control of the actual reference marks and offers the possibility of having a caption for our tabular material. In this sense, the `threeparttable` environment is similar to the nonfloating `table` environment.

```
\begin{threeparttable}
 \caption{\textbf{PostScript type 1 fonts}}
  \begin{tabular}{ll}
   Courier\tnote{a} & cour, courb, courbi, couri\\
   Charter\tnote{b} & bchb, bchbi, bchr, bchri \\
   Nimbus\tnote{c} & unmr, unmrs \\
   URW Antiqua\tnote{c} & uaqrrc\\
   URW Grotesk\tnote{c} & ugqp\\
   Utopia\tnote{d} & putb, putbi, putr, putri
  \end{tabular}
 \begin{tablenotes}
  \item[a] Donated by IBM.
  \item[b] Donated by Bitstream.
  \item[c] Donated by URW GmbH.
  \item[d] Donated by Adobe.
 \end{tablenotes}
\end{threeparttable}
```

Table 14.2: **PostScript type 1 fonts**

Courier[a]	cour, courb, courbi, couri
Charter[b]	bchb, bchbi, bchr, bchri
Nimbus[c]	unmr, unmrs
URW Antiqua[c]	uaqrrc
URW Grotesk[c]	ugqp
Utopia[d]	putb, putbi, putr, putri

[a] Donated by IBM.
[b] Donated by Bitstream.
[c] Donated by URW GmbH.
[d] Donated by Adobe.

14.1.2. CUSTOMIZING FOOTNOTES

If the user wishes the footnote numbering to be reset to 1 for each `\section` command with the article class, this may be achieved by putting

```
\setcounter{footnote}{0}
```

before every section or using the following command at preamble[4]

```
\@addtoreset{footnote}{section}
```

The internal footnote counter has the name `footnote`. Each call to `\footnote` increments this counter by one and prints the new value in Arabic numbering as the footnote marker. A different style of marker can be implemented with the command

```
\renewcommand{\thefootnote}{number\_style{footnote}}
```

where *number_style* is one of the counter print commands; `\arabic`, `\roman`, `\Roman`, `\alph`, or `\Alph`. However, for the counter `footnote`, there is an additional counter print command available, `\fnsymbol`, which prints the counter values 1–9 as one of nine symbols:

⋆ † ‡ § ¶ ‖ ⋆⋆ †† ‡‡

[4] This command will only work within `\makeatletter` and `\makeatother`.

It is up to the user to see that the footnote counter is reset to zero sometime before the tenth `\footnote` call is made. If the user wants to add values above nine, then he has to edit the definition of `\fnsymbol`. See an example, which allows up to 12 footnotes without resetting the counter:

```
\makeatletter
\def\@fnsymbol#1{\ensuremath{\ifcase#1\or *\or \dagger
  \or \ddagger\or \mathsection\or \mathparagraph
  \or \|\or **\or \dagger\dagger\or \ddagger\ddagger
  \or \mathsection\mathsection\or \mathparagraph
  \mathparagraph \or \|\|\else\@ctrerr\fi}}
\renewcommand{\thefootnote}{\fnsymbol{footnote}}
\makeatother
```

An optional argument may be added to the `\footnote` command:

`\footnote[`*num*`]{`*footnote_text*`}`

where *num* is a positive integer that is used instead of the value of the footnote counter for the marker. In this case, the footnote counter is not incremented. For example**,

```
\renewcommand{\thefootnote}{\fnsymbol{footnote}}
 For example\footnote[7]{The 7$^{\rm th}$ symbol .... marker.},
\renewcommand{\thefootnote}{\arabic{footnote}}
```

where the last line is necessary to restore the footnote marker style to its standard form. Otherwise, all future footnotes would be marked with symbols and not with numbers.

14.1.3. FOOTNOTE STYLE PARAMETERS

The appearance of the standard footnote can be changed by customizing the parameters listed below:

`\footnotesize` The font size used inside footnotes.

`\footnotesep` The height of a strut placed at the beginning of every footnote. If it is greater than the `\baselineskip` used for `\footnotesize`, then additional vertical space will be inserted above each footnote.

**The 7^{th} symbol appears as the footnote marker.

`\skip\footins` A low-level TEX command that defines the space between the main text and the start of the footnotes. You can change its value with the command `\setlength` or `\addtolength` commands by putting `\skip\footins` into the first argument, e.g.,

```
\addtolength{\skip\footins}{3mm}
```

`\footnoterule` A macro to draw the rule separating footnotes from the main text. It is executed right after the vertical space of `\skip\footins`. It should take zero vertical space, i.e., it should use a negative skip to compensate for any positive space it occupies, for example:

```
\renewcommand{\footnoterule}{\vspace*{-3pt}%
\rule{.4\columnwidth}{0.4pt}\vspace*{2.6pt}}
```

You can also construct a fancier "rule" e.g., one consisting of a series of dots:

```
\renewcommand{\footnoterule}{\vspace*{-3pt}%
\qquad\dotfill\qquad\vspace*{2.6pt}}
```

14.2. MARGINAL NOTES

`\marginpar[`*left-text*`]{`*right-text*`}`

The `\marginpar` command generates a marginal note. This command typesets the text given as an argument in the margin, the first line at the same height as the line in the main text where the `\marginpar` command occurs. The marginal note appearing here was generated with

```
... command occurs\marginpar{This is a marginal note}. The ...
```

This is a marginal note

When only the mandatory argument *right-text* is specified, then the text goes to the right margin for one-sided printing; to the outside margin for two-sided printing; and to the nearest margin for two-column formatting. When you specify an optional argument, it is used for the left margin, while the second (mandatory) argument is used for the right.

There are a few important things to understand when using marginal notes. First, `\marginpar` command does not start a paragraph, that is, if it is used before the first word of a paragraph, the vertical alignment may not match the beginning of the paragraph. Secondly, if the margin is narrow, and the words are long (as in German), you may have to precede the first word by a `\hspace{0pt}` command to allow hyphenation of the first word. These two potential problems can be eased by defining a command `\marginlabel{`*text*`}`, which starts with an empty box `\mbox{}`, typesets a marginal note ragged left, and adds a `\hspace{0pt}` in front of the argument.

```
\newcommand{\marginlabel}[1]
{\mbox{}\marginpar{\raggedleft\hspace{0pt}#1}}
```

By default, in one-sided printing the marginal notes go on the outside margin. These defaults can be changed by the following declarations:

`\reversemarginpar`	Marginal notes go into the opposite margin with respect to the default one.
`\normalmarginpar`	Marginal notes go into the default margin.

14.2.1. USES OF MARGINAL NOTES

`\marginpar{}` can be used to draw attention to certain text passages by marking them with a vertical bar in the margin. The example marking this paragraph was made by including

```
\marginpar{\rule[-10.5mm]{1mm}{10mm}}
```

in the first line. By defining a macro `\query` as shown below

```
\def\query#1#2{\underline{#1}\marginpar{#2}}
```

Hey! Look

we can produce queries. For example LaTeX. This query is produced with the following command.

```
For example \query{\LaTeX}{Hey!\\ Look}{}. This ...
```

14.2.2. STYLE PARAMETERS FOR MARGINAL NOTES

The following style parameters may be changed to redefine how marginal notes appear:

`\marginparwidth` Determines the width of the margin box.

`\marginparsep` Sets the separation between the margin box and the edge of the main text.

`\marginparpush` Is the smallest vertical distance between two marginal notes.

These parameters are all lengths and are assigned new values as usual with the `\setlength` command.

14.3. ENDNOTES

Scholarly works usually group notes at the end of each chapter or at the end of the document. These are called endnotes. Endnotes are not supported in standard LaTeX, but they can be created in several ways.

The package `endnotes` (by John Lavagnino) typesets endnotes in a way similar to footnotes. It uses an extra external file, with extension `.ent`, to hold the text of the endnotes. This file can be deleted after the run since a new version is generated each time.

With this package you can output your footnotes as endnotes by simply giving the command:

```
\renewcommand{\footnote}{\endnote}
```

The user interface for endnotes is very similar to the one for footnotes after substituting the word "foot" for "end". The following example shows the principle of the use of endnotes, where you save text in memory with the `\endnote` command, and then typeset all accumulated text material at a point in the document controlled by the user.

This is simple text.[1] This is simple text.[2] This is simple text.[3]

Notes

[1] The first endnote.
[2] The second endnote.
[3] The third endnote.

This is some more simple text

```
This is simple text.\endnote{The
 first endnote.} This is simple
 text.\endnote{The second
 endnote.} This is simple
text.\endnote{The third
endnote.}

\theendnotes\bigskip
This is some more simple text
```

Index

^ (carot), 97
$, 95
\`, 74
\<, 74
\-, 74
\,, 99, 127
\;, 98
\', 74
\(...\), 96
\[...\], 96
\\, 21, 76
\+, 74
\++, 74
_ (underscore), 98
\>, 70
\=, 70
\{, 120
\}, 120
\\[5pt], 76
|, 80
\addcontentsline, 44
\addtocontents, 43, 44
\afterpage, 170
\and, 21
\arraystretch, 84
article, 22
\@, 4
@-expression, 173
\@dotsep, 46
\@dottedtocline, 45
\@pnumwidth, 46
\@tocmarg, 46
\author, 20
\baselineskip, 164
\begin{abstract}, 22
\begin{align}, 110
\begin{align*}, 106–108, 111, 112
\begin{aligned}, 108
\begin{bmatrix}, 113
\begin{cases}, 109
\begin{description}, 65
\begin{displaymath}, 96
\begin{document}, 2
\begin{enumerate}, 61
\begin{equation}, 103, 109, 111
\begin{equation*}, 103, 109, 112
\begin{figure}, 167
\begin{gather*}, 106
\begin{itemize}, 58, 59
\begin{longtable}, 90
\begin{math}, 96
\begin{multline*}, 104
\begin{pmatrix}, 112, 113, 115
\begin{quotation}, 56
\begin{quote}, 56
\begin{smallmatrix}, 115
\begin{split}, 105, 106, 111
\begin{tabular}, 77
\begin{thebibliography}, 28
\begin{verse}, 57
\begin{vmatrix}, 113
\bibitem, 27, 28
\bibliography, 35
\bibliographystyle, 36
\bibpunct, 33
\biggl, 117
\Biggl, 117
\biggr, 118
\bigl, 117
\Bigl, 117
\bigr, 117
\binom, 118, 119
\boldsymbol, 128
book, 21, 24, 46
\bottomfraction, 171
\bottomnumber, 170
BoundingBox, 182
\caption, 43, 44, 167, 168, 190
\centering, 83
\cfrac, 121

\chapter, 19, 22, 24, 46
\cite, 27, 28
\citealp, 32
\citealt, 32
\citeauthor, 31, 32
\citeauthor*, 32
\citep, 31, 32
\citep*, 31
\citepalias, 33
\citet, 31
\citet*, 31
\citetalias, 33
\citetext, 32
\citeyear, 32
\citeyearpar, 32
\cline, 83
\date, 20
\dbinom, 119
\DeclareGraphicsExtensions, 184
\DeclareMathAlphabet, 129
\DeclareMathOperator, 126
\depth, 156
\descriptionlabel, 65
\dfrac, 119
\documentclass, 2, 15
  article, 15
  article.cls, 2
  book, 15
  book.cls, 2
  letter, 15
  report, 15
  options, 15
    notitlepage, 17
    openany, 17
    openright, 17
    titlepage, 17
\dot, 121
\dots, 115
\dotsb, 115
\dotsc, 115
\dotsi, 115
\dotsm, 115
\emph, 2
\end{abstract}, 22
\end{align}, 110
\end{align*}, 106–108, 111, 112
\end{aligned}, 108
\end{bmatrix}, 113
\end{cases}, 109
\end{description}, 65
\end{displaymath}, 96
\end{document}, 2
\end{enumerate}, 61
\end{equation}, 103, 109, 111
\end{equation*}, 103, 109, 112
\end{figure}, 167
\endfirsthead, 90
\endfoot, 90
\end{gather*}, 106
\endhead, 90
\end{itemize}, 58, 59
\endlastfoot, 90
\end{longtable}, 90
\end{math}, 96
\end{multline*}, 104
\endnote, 205
\end{pmatrix}, 112, 113, 115
\end{quotation}, 56
\end{quote}, 56
\end{smallmatrix}, 115
\end{split}, 105, 106, 111
\end{tabular}, 77
\end{verse}, 57
\end{vmatrix}, 113
\eqef, 191
\externaldocument, 195
\fbox, 151, 154
\fboxrule, 152, 156
\fboxsep, 152, 154, 156
\FloatBarrier, 169
\floatpagefraction, 171
\footnote, 21, 197
\footnotemark, 198, 199
\footnotetext, 198
\frac, 118, 119
\framebox, 154, 157
  position, 155
\genfrac, 119–121
\glossaryentry, 53
\graphicspath, 184
harvard, 27
\hat, 121
\hdotsfor, 114
\height, 156
\hline, 80
\includegraphics, 167, 180
  angle, 181
  bb, 183
  clip, 182
  height, 180

width, 180
\index, 48
\int, 125
\intertext, 107
\item, 58, 65, 66
\label, 167, 188–190, 195
\labelitemi, 60
\labelitemii, 60
\labelitemiii, 60
\labelitemiv, 60
\ldots, 99
\left, 116
\limits, 125
\listoffigures, 43, 168
\listoftables, 43
\lq, 5
\makeglossary, 53
\maketitle, 20
\marboth, 19
\marginlabel, 204
\marginpar, 203, 204
\marginparpush, 205
\marginparsep, 205
\marginparwidth, 205
\markboth, 18
\markright, 19
\mathbf, 128
\mathbin, 100
\mathrel, 101
\mathrm, 129
\mathsf, 129
\mbox, 157, 164
mpfootnote, 198
\multicolumn, 81, 84
\multirowsetup, 88, 89
natbib, 27, 31
\newcommand, 102
\newtheorem, 137–139, 190
\nocite, 37
\nofiles, 44
\noindent, 4
\normalmarginpar, 204
notitlepage, 22
\numberline, 44
\numberwithin, 110
\overleftarrrow, 122
\overrightarrow, 122
\overset, 122
\pagenumbering, 19
Alph, 19
arabic, 19
roman, 19
Roman, 19
\pageref, 167, 191, 192, 194, 195
\pagestyle, 17
empty, 17
headings, 17
myheadings, 17
plain, 17
\paragraph, 22
\parbox
alignment, 161
bottom aligned, 160
height, 161
inner position, 161
middle aligned, 159
top aligned, 160
width, 161
\part, 25
\phantom, 152
\poptabs, 73
\prabox, 159
\prod, 126
\pushtabs, 73
quad, 70
\raisebox, 164
\ref, 167, 187–189, 192, 195
report, 21, 24, 46
\reversemarginpar, 204
\right, 116
\rq, 5
\ruje
height, 162
\rule, 162
lift, 162
width, 162
\section, 22, 24
\section*, 24
\setcounter, 19, 170
\setlength, 20, 152
\sideset, 126
\sqrt, 98
\subparagraph, 22
\subsection, 22
\subsubsection, 22
\sum, 124
\tabbingsep, 74
\tableofcontents, 43
\tabularnewline, 79

\tag, 110
\tbinom, 119
\text, 104
\textfraction, 171
>{*decl*}, 87
LR box, 154
\textstyle, 124
\textwidth, 20, 153
\tfrac, 119
\thanks, 21
\theendnote, 205
\thefootnote, 198, 199
\theoremstyle, 140, 141
\thispagestyle, 18
\title, 20
titlepage, 22
tocdepth, 46
\topfraction, 171
\topnumber, 170
\totalnumber, 170
\underleftarrow, 122
\underrightarrow, 122
\underset, 122
\usepackage, 47, 86, 90, 100, 103, 179, 192, 195
\vpageref, 194
\vref, 192
\width, 156, 157
\xleftarrow, 122
\xrightarrow, 122

abstract, 22
accents, 6
  \`, 6
  \^, 6
  \~, 6
  \., 6
  \', 6
  \", 6
  \b, 6
  \c, 6
  \d, 6
  \=, 6
  \H, 6
  \t, 6
  \u, 6
  \v, 6
AMS arrows, 133
AMS binary operators, 134
AMS binary relations, 134
AMS delimiters, 133
AMS Greek, 133
AMS Hebrew, 134
AMS miscellaneous, 134
AMS negated arrows, 133
AMS negated binary relations, 135
array features, 85
arrow symbols, 131
axiom, 137

backslash, 2
bibliographic style, 29
  apalike, 29
  astron, 29
  authordate, 29
  chicago, 29
  harvard, 29
  natbib, 29
  plain, 29
bibliography, 27, 35
  thebibliography, 37
  database, 35
    creation, 37
    entry type, 38
    field, 38
    field-optional, 38
    field-required, 38
    procedure, 41
  style, 35, 36
    .bbl file, 37
    \nocite, 37
    abbrv, 36
    acm, 36
    alpha, 36
    apalike, 36
    others, 36
    plain, 36
    steps, 37
    unsrt, 36
BibTeX, 27, 35
binary operators, 130
binary relation, 100
blank line, 3
\textbf, 11
bounding box, 182
box, 151
  \baselineskip, 164
  \depth, 156
  \fbox, 151, 154
    indivisible unit, 153

\fboxrule, 152, 156
\fboxsep, 152, 154, 156
\framebox, 154, 157
  position, 155
\height, 156
\mbox, 157, 164
\parbox, 159
  alignment, 161
  bottom aligned, 160
  height, 161
  inner position, 161
  middle aligned, 159
  top aligned, 160
  width, 161
\phantom, 152
\rainsebox, 164
\raisebox, 164
\rule, 162
  height, 162
  lift, 162
  width, 162
\setlength, 152
LR box, 154
\textwidth, 153
\width, 156, 157
character box, 151
depth, 155
frame, 151
height, 155
invisible line, 162
length, 152
line box, 151
natural dimensions, 155
padding, 152
paragraph box, 151
rule, 162
width, 155
with frame, 151

Carlisle, David, 62
character box, 151
citation, 31
  aliasing, 33
  author–year, 31
  multiple, 31
  numerical, 32
  parantheses, 32
  partial, 32
  punctuation, 33
  styling, 33
clipping picture, 182
column, 69
corollary, 138
counter, 19
cross-ref, 187
  \caption, 190
  \eqref, 191
  \externaldocument, 195
  \label, 188–190, 195
  \newtheorem, 190
  \pageref, 191, 192, 194, 195
  \ref, 187–189, 192, 195
  \vpageref, 194
  \vref, 192
  xr package, 194
cross reference, 187

dashes, 6
  emdash '—', 6
  endash '–', 6
  hyphen '-', 6
definition list, 64
delimiter, 115
delimiters, 132
depth, 155
description list, 64
designer theorem, 140
  ready made styles, 140
device driver, 179
displayed text, 55
  \begin{enumerate}, 61
  \begin{itemize}, 58, 59
  \begin{quotation}, 56
  \begin{quote}, 56
  \begin{verse}, 57
  \end{enumerate}, 61
  \end{itemize}, 58, 59
  \end{quotation}, 56
  \end{quote}, 56
  \end{verse}, 57
  list, 57
  poetry, 57
document, 15
  class, 15
document class, 15
dots, 115
dvips, 179

ellipsis, 99
Emacs, 1

encapsulated postscript, 180
endnote, 197, 205
  \endnote, 205
  \theendnote, 205
enhanced tabbing, 75
environment, 56
  {align}, 110
  {align*}, 106–108, 111, 112
  {aligned}, 108
  {alignedat}, 108
  {bmatrix}, 113
  {cases}, 109
  {displaymath}, 96
  {enumerate}, 27
  {equation}, 103, 109, 111
  {equation*}, 103, 109, 112
  {figure}, 167
  {gather*}, 106
  {gathered}, 108
  {longtable}, 90
  {math}, 96
  {multline*}, 104
  {pmatrix}, 112, 113, 115
  {smallmatrix}, 115
  {split}, 105, 106, 111
  {tabbing}, 69
  {table}, 171
  {tabular}, 77
  {thebibliography}, 27, 37
  {vmatrix}, 113
equation number, 103

figure, 167
  \begin{figure}, 167
  \caption, 167, 168
  \end{figure}, 167
  \FloatBarrier, 169
  \includegraphics, 167
    \DeclareGraphicsExtensions, 184
    \graphicspath, 184
  \listoffigures, 168
  features, 167
  graphics
    search path, 184
  outer paragraph mode, 168
  placement, 169
float, 167
  \afterpage, 170
  \bottomfraction, 171
  \bottomnumber, 170
  \floatpagefraction, 171
  \textfraction, 171
  \topfraction, 171
  \topnumber, 170
  \totalnumber, 170
  table, 171
float fraction, 170
floating figure, 167
  creation, 167
fonts, 10
  \bfseries, 12
  \mdseries, 12
  \rmfamily, 12
  \sffamily, 12
  \textbf, 12
  \textmd, 12
  \textrm, 12
  \textsf, 12
  \texttt, 12
  \ttfamily, 12
  roman, 11
  sans serif, 11
font size, 15
footnote, 197
  \footnote, 197
  \footnotemark, 197–199
  \footnotetext, 198
  mpfootnote, 198
  \thefootnote, 198, 199
  customize, 201
  in tabular, 198
  number style
    \alph, 201
    \Alph, 201
    \arabic, 201
    \fnsymbol, 201
    \roman, 201
    \Roman, 201
  numer style, 201
  style parameters, 202
    \footins, 202
    \footnoterule, 203
    \footnotesep, 202
    \footnotesize, 202
formatting length, 20
fraction, 118
function, 100

glossary, 52
  \glossaryentry, 53

\makeglossary, 53
graphic file, 180
graphics, 179
BoundingBox, 182
\includegraphics, 180
angle, 181
bb, 183
clip, 182
height, 180
width, 180
bounding box, 182
clipping, 182
device driver, 179
dvips, 179
encapsulated postscript, 180
extensions, 184
graphic file, 180
inclusion, 180
inclusion, 179
pdflatex, 179
search path, 184
xdvi, 179
Greek letters, 130
group of equations, 106

heading declaration, 18
height, 155

indentation, 4
\noindent, 4
index, 47
\index, 48
\makeindex, 48
makeindex, 47, 48
\printindex, 48
theindex, 53
cross-reference, 49
page range, 49
presentation, 51
process steps, 49
simple entry, 48
special characters, 52
subentry, 48
India, iii
Indian TEX Users Group, iii
\textit, 10

Knuth, Donald, 95, 129
Krishnan, E., iii
Kronecker, 4

large delimiters, 132
LATEX, 2
LATEX, 1
Lavignino, Johan, 205
lemma, 137
limits, 123, 127
line box, 151
Linux, 1
list, 57
\begin{description}, 65
\descriptionlabel, 65
\end{description}, 65
\item, 65, 66
\labelitemi...iv, 60
definition, 64
description, 64
label, 60
nested, 58, 59, 62
ordered, 61
list of figures, 43
list of tables, 43
log-like symbols, 132
longtable, 90
\endfirsthead, 90
\endfoot, 90
\endhead, 90
\endlastfoot, 90
\newpage, 90

makeindex, 47
marginal note, 203
marginpar, 197, 203
\marginlabel, 204
\marginpar, 203, 204
\marginparpush, 205
\marginparsep, 205
\marginparwidth, 205
\normalmarginpar, 204
\reversemarginpar, 204
math, 95
^ (carot), 97
$, 95
$$, 96
\,, 99
\(...\), 96
\[...\], 96
_ (underscore), 98
\{, 120
\}, 120
\begin{align}, 110

\begin{align*}, 106–108, 111, 112
\begin{aligned}, 108
\begin{bmatrix}, 113
\begin{cases}, 109
\begin{displaymath}, 96
\begin{equation}, 103, 109, 111
\begin{equation*}, 103, 109, 112
\begin{gather*}, 106
\begin{math}, 96
\begin{multline*}, 104
\begin{pmatrix}, 112, 113, 115
\begin{smallmatrix}, 115
\begin{split}, 105, 106, 111
\begin{vmatrix}, 113
\biggl, 117
\Biggl, 117
\biggr, 118
\bigl, 117
\Bigl, 117
\bigr, 117
\binom, 118, 119
\boldsymbol, 128
\cfrac, 121
\dbinom, 119
\DeclareMathAlphabet, 129
\DeclareMathOperator, 126
\dfrac, 119
\dot, 121
\dots, 115
\dotsb, 115
\dotsc, 115
\dotsi, 115
\dotsm, 115
\end{align}, 110
\end{align*}, 106–108, 111, 112
\end{aligned}, 108
\end{bmatrix}, 113
\end{cases}, 109
\end{displaymath}, 96
\end{equation}, 103, 109, 111
\end{equation*}, 103, 109, 112
\end{gather*}, 106
\end{math}, 96
\end{multline*}, 104
\end{pmatrix}, 112, 113, 115
\end{smallmatrix}, 115
\end{split}, 105, 106, 111
\end{vmatrix}, 113
\frac, 118, 119
\genfrac, 119–121
\hat, 121
\hdotsfor, 114
\int, 125
\intertext, 107
\ldots, 99
\left, 116
\limits, 125
\mathbf, 128
\mathbin, 100
\mathrel, 101
\mathrm, 129
\mathsf, 129
\numberwithin, 110
\overleftarrow, 122
\overrightarrow, 122
\overset, 122
\prod, 126
\right, 116
\sideset, 126
\sqrt, 98
\sum, 124
\tag, 110
\tbinom, 119
\text, 104
\textstyle, 124
\tfrac, 119
\underleftarrow, 122
\underrightarrow, 122
\underset, 122
\xleftarrow, 122
\xrightarrow, 122
AMS arrows, 133
AMS binary operators, 134
AMS binary relations, 134
AMS delimiters, 133
AMS Greek, 133
AMS Hebrew, 134
AMS miscellaneous, 134
AMS negated arrows, 133
AMS negated binary relations, 135
arrow symbols, 131
binary operators, 130
binary relation, 100
custom commands, 101
delimiter, 115
delimiters, 132
displayed equation, 96
dots, 115
ellipsis, 99
equation, 103

fraction, 118
function, 100
Greek letters, 130
grouping, 98
group of equations, 106
in-text, 95
large delimiters, 132
limits, 123, 127
log-like symbols, 132
math accents, 132
math alphabets, 136
matrix, 112
miscellaneous symbols, 131
nested roots, 99
new operator, 126
numbered equations, 109
operator, 100
operator space, 101
punctuation symbols, 131
relation symbols, 130
root, 98
nested roots, 99
vinculum, 99
spacing, 96
subscript, 97, 98
superscript, 97
theorem, 137
thinspace, 99
under–over, 121
variable arguments, 102
variable-sized symbols, 132
vinculum, 99
math accent, 121
math accents, 132
math alphabets, 136
math mode, 95
math symbols, 100
matrix, 112
minitoc, 46
\dominitoc, 46
\faketableofcontents, 47
\minitoc, 47
\minitocdepth, 47
\minitocoff, 47
\mtcfont, 47
\mtcindent, 47
miscellaneous symbols, 131
multirow, 88
\multirowsetup, 88, 89
nested roots, 99
new operator, 126
numbered equations, 109

operator, 100
operator space, 101

package
afterpage, 169
amsmath, 95, 102, 103, 108, 112, 122, 126, 128, 129, 191
amssymb, 100
amsthm, 140
array, 86, 88
calc, 154, 156
dcolumn, 93
delarray, 93
endnotes, 205
enumerate, 62
fancyhdr, 19
graphics, 179
graphicx, 166, 179, 180
hhline, 93
latexsym, 100
longtable, 90
minitoc, 46
multirow, 88
natbib, 29
options, 29
placeins, 169
rotating, 166
sectsty, 25
tabularx, 93
varioref, 191
xr, 194
loading, 179
page formats, 16
onecolumn, 16
oneside, 16
twocolumn, 16
twoside, 16
page numbering, 19
page style, 17
paper size, 16
a4paper, 16
a5paper, 16
b5paper, 16
executivepaper, 16
legalpaper, 16
letterpaper, 16

paragraph, 3
  blank line, 3
paragraph box, 151
pdflatex, 179
poetry, 57
previewer, 1
  xdvi, 2
printer driver, 1
proposition, 137
punctuation symbols, 131

quote, 5
  " (double), 5
  ` (left), 5
  \lq, 5
  ' (right), 5
  \rq, 5
  double quote, 5
  single quote, 5

relation symbols, 130
root, 98
row, 69

single equation, 103
space, 4
  \,, 99, 127
  \;, 98, 99
  \@, 4
  \quad, 70
  thickspace, 98
  thinspace, 99
  visible space, 5
special characters
  !` (¡), 7
  !' (!'), 7
  ?` (¿), 7
  \aa (å), 7
  \AA (Å), 7
  \ae (æ), 7
  \AE (Æ), 7
  \l (ł), 7
  \L (Ł), 7
  \oe (œ), 7
  \OE (Œ), 7
  \o (ø), 7
  \O (Ø), 7
  \ss (ß), 7
special symbols, 7
  _ (_), 8
  \$ ($), 8
  \# (#), 8
  \{ ({), 8
  \} (}), 8
  \% (%), 8
  \textasciitilde (~), 8
  \textbackslash (\), 8
subscript, 97
superscript, 97

tabbing, 69
  \`, 74
  \<, 74
  \>, 70
  \-, 74
  \', 74
  \+, 74
  \++, 74
  \=, 70
  \poptabs, 73
  \pushtabs, 73
  \tabbingsep, 74
  enhanced, 75
  popping, 73
  pushing, 73
table, 76, 171
  tabular, 171
  tabular*, 171
table of contents, 43
  multiple, 46
tabular, 76, 171
  \\, 76
  &, 76
  \\[5pt], 76
  |, 80
  \arraystretch, 84
  @-expression, 173
  \cline, 83
  \hline, 80
  {lcr}, 80
  \multicolumn, 81, 84
  \tabularnewline, 79, 82
  >{*decl*}, 87
  array features, 85
  cols, 172
  col. specifier, 77
  enhancement, 86
  multirow, 88
  pos, 172
tabular*, 171

@-expression, 173
cols, 172
width, 172
TEX, 1
text
alignment, 9
centering, 9
displayed, 55
flushleft, 9
flushright, 9
positioning, 9
text editor, 1
Emacs, 1
text style
\bfseries, 12
\itshape, 12
\scshape, 12
\slshape, 12
\textit, 12
\textsc, 12
\textsl, 12
\textup, 12
\upshape, 12
theorem, 137
\newtheorem, 137–139
\theoremstyle, 140, 141
axiom, 137
corollary, 138
designer theorem, 140
lemma, 137
proposition, 137
theorem-like, 137
theorem-like, 137
The TEXBook, 129
Fine Points of Math Typing, 129
title, 20
Trivandrum, iii
type size, 12
\footnotesize, 13
\huge, 13
\Huge, 13
\large, 13
\Large, 13
\LARGE, 13
\normalsize, 13
\scriptsize, 13
\small, 13
\tiny, 13
typestyle
\itshape, 11
\scshape, 11
\upshape, 11
type style, 10

under–over, 121

variable arguments, 102
variable-sized symbols, 132
verse, 57
vinculum, 99

Why LATEX?, 3
width, 155
word processors, 1

xdvi, 179

Zandt, Timothy Van, 183

Sayahna Foundation • JWRA 34 • Jagathy • Trivandrum 695014 •
INDIA • URL: ml.sayahna.org • EMAIL: <info@sayahna.org>