Essential LaTeX

U05.04–202
November 1992

Jon Warbrick

1 Introduction

This document is an attempt to give you all the essential information that you will need in order to use the LaTeX Document Preparation System. Only very basic features are covered, and a vast amount of detail has been omitted. In a document of this size it is not possible to include everything that you might need to know, and if you intend to make extensive use of the program you should refer to a more complete reference. Attempting to produce complex documents using only the information found below will require much more work than it should, and will probably produce a less than satisfactory result.

The main reference for LaTeX is *The* LaTeX *User's guide and Reference Manual* by Leslie Lamport (M05.04–200). This contains all the information that you will ever need to know about the program, and you will need access to a copy if you are to use LaTeX seriously.

Both the Manual and this document avoid mentioning anything that depends on the particular computer system that you are using. This is because LaTeX is available on a number of systems and they all differ in one way or another. Instead, they both refer to a *local guide* for their particular system. For LaTeX on the Prime, the *local guide* is the handout *The Local Guide to* LaTeX *on the Plymouth Prime System* (U05.04–201).

2 How does LaTeX work?

In order to use LaTeX you generate a file containing both the text that you wish to print and instructions to tell LaTeX how you want it to appear. You will normally create this file using your system's text editor. You can give the file any name you like, but it should end "`.TEX`" to identify the file's contents. You then get LaTeX to process the file, and it creates a new file of typesetting commands; this has the same name as your file but the "`.TEX`" ending is replaced by "`.DVI`". This stands for '*Dev*ice *I*ndependent' and, as the name implies, this file can be used to create output on a range of printing devices. Your *local guide* will go into more detail.

Rather than encourage you to dictate exactly how your document should be laid out, LaTeX instructions allow you describe its *logical structure*. For example, you can think of a quotation embedded within your text as an element of this logical structure: you would normally expect a quotation to be displayed in a recognisable style to set it off from the rest of the text. A human typesetter would recognise the quotation and handle it accordingly, but since LaTeX is only a computer program it requires your help. There are therefore LaTeX commands that allow you to identify quotations and as a result allow LaTeX to typeset them correctly.

Fundamental to LaTeX is the idea of a *document style* that determines exactly how a document will be formatted. LaTeX provides standard document styles that describe how standard logical structures (such as quotations) should be formatted. You may have to supplement these styles by specifying the formatting of logical structures peculiar to your document, such as mathematical formulae. You can also modify the standard document styles or even create an entirely new one, though you should know the basic principles of typographical design before creating a radically new style.

There are a number of good reasons for concentrating on the logical structure rather than on the appearance of a document. It prevents you from making elementary typographical errors in the mistaken idea that they improve the aesthetics of a document—you should remember that the primary function of document design is to make documents easier to read, not prettier. It is more flexible, since you only need to alter the definition of the quotation style to change the appearance of all the quotations in a document. Most important of all, logical design encourages better writing. A visual system makes it easier to create visual effects rather than a coherent structure; logical design encourages you to concentrate on your writing and makes it harder to use formatting as a substitute for good writing.

3 A Sample LaTeX file

Have a look at the example LaTeX file in Figure 1. It is a slightly modified copy of the standard LaTeX example file `SMALL.TEX`. Your local guide will tell you how you can make your own copy of this file. The line numbers down the left-hand side are not part of the file, but have been added to make it easier to identify various portions. Also have a look at Figure 2 which shows, more or less, the result of processing this file.

3.1 Running Text

Most documents consist almost entirely of running text—words formed into sentences, which are in turn formed into paragraphs—and the example file is no exception. Describing running text poses

```
% SMALL.TEX -- Released 5 July 1985
% USE THIS FILE AS A MODEL FOR MAKING YOUR OWN LaTeX INPUT FILE.
% EVERYTHING TO THE RIGHT OF A  %  IS A REMARK TO YOU AND IS IGNORED
% BY LaTeX.
%
% WARNING!  DO NOT TYPE ANY OF THE FOLLOWING 10 CHARACTERS EXCEPT AS
% DIRECTED:       &   $   #   %   _   {   }   ^   ~   \

\documentstyle[11pt,a4]{article}  % YOUR INPUT FILE MUST CONTAIN THESE
\begin{document}                  % TWO LINES PLUS THE \end COMMAND AT
                                  % THE END

\section{Simple Text}          % THIS COMMAND MAKES A SECTION TITLE.

Words are separated by one or    more      spaces.  Paragraphs are
    separated by one or more blank lines.  The output is not affected
by adding extra spaces or extra blank lines to the input file.


Double quotes are typed like this: ``quoted text''.
Single quotes are typed like this: `single-quoted text'.

Long dashes are typed as three dash characters---like this.

Italic text is typed like this: {\em this is italic text}.
Bold   text is typed like this: {\bf this is  bold  text}.

\subsection{A Warning or Two}        % THIS MAKES A SUBSECTION TITLE.

If you get too much space after a mid-sentence period---abbreviations
like etc.\ are the common culprits)---then type a backslash followed by
a space after the period, as in this sentence.

Remember, don't type the 10 special characters (such as dollar sign and
backslash) except as directed!  The following seven are printed by
typing a backslash in front of them:  \$  \&  \#  \%  \_  \{  and  \}.
The manual tells how to make other symbols.

\end{document}                    % THE INPUT FILE ENDS LIKE THIS
```

Figure 1: A Sample LaTeX File

1 Simple Text

Words are separated by one or more spaces. Paragraphs are separated by one or more blank lines. The output is not affected by adding extra spaces or extra blank lines to the input file.

Double quotes are typed like this: “quoted text”. Single quotes are typed like this: ‘single-quoted text’.

Long dashes are typed as three dash characters—like this.

Italic text is typed like this: *this is italic text*. Bold text is typed like this: **this is bold text**.

1.1 A Warning or Two

If you get too much space after a mid-sentence period—abbreviations like etc. are the common culprits)—then type a backslash followed by a space after the period, as in this sentence.

Remember, don't type the 10 special characters (such as dollar sign and backslash) except as directed! The following seven are printed by typing a backslash in front of them: $ & # % _ { and }. The manual tells how to make other symbols.

Figure 2: The result of processing the sample file

no problems, you just type it in naturally. In the output that it produces, LaTeX will fill lines and adjust the spacing between words to give tidy left and right margins. The spacing and distribution of the words in your input file will have no effect at all on the eventual output. Any number of spaces in your input file are treated as a single space by LaTeX, it also regards the end of each line as a space between words (see lines 15–17). A new paragraph is indicated by a blank line in your input file, so don't leave any blank lines unless you really wish to start a paragraph.

LaTeX reserves a number of the less common keyboard characters for its own use. The ten characters

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

should not appear as part of your text, because if they do LaTeX will get confused.

3.2 LaTeX Commands

There are a number of words in the file that start '`\`' (see lines 9, 10 and 13). These are LaTeX *commands* and they describe the structure of your document. There are a number of things that you should realise about these commands:

- All LaTeX commands consist of a '`\`' followed by one or more characters.
- LaTeX commands should be typed using the correct mixture of upper- and lower-case letters. `\BEGIN` is *not* the same as `\begin`.
- Some commands are placed within your text. These are used to switch things, like different typestyles, on and off. The `\em` command is used like this to emphasise text, normally by changing to an *italic* typestyle (see line 25). The command and the text are always enclosed between '`{`' and '`}`'—the '`{\em`' turns the effect on and and the '`}`' turns it off.
- There are other commands that look like

  ```
  \command{text}
  ```

  In this case the text is called the "argument" of the command. The `\section` command is like this (see line 13). Sometimes you have to use curly brackets '`{}`' to enclose the argument, sometimes square brackets '`[]`', and sometimes both at once. There is method behind this apparent madness, but for the time being you should be sure to copy the commands exactly as given.

- When a command's name is made up entirely of letters, you must make sure that the end of the command is marked by something that isn't a letter. This is usually either the opening bracket around the command's argument, or it's a space. When it's a space, that space is always ignored by LaTeX. We will see later that this can sometimes be a problem.

3.3 Overall structure

There are some LaTeX commands that must appear in every document. The actual text of the document always starts with a `\begin{document}` command and ends with an `\end{document}` command (see lines 10 and 39). Anything that comes after the `\end{document}` command is ignored. Everything that comes before the `\begin{document}` command is called the *preamble*. The preamble can only contain LaTeX commands to describe the document's style.

One command that must appear in the preamble is the `\documentstyle` command (see line 9). This command specifies the overall style for the document. Our example file is a simple technical document, and uses the `article` style, modified to print in eleven-point type on A4 paper. There are other styles that you can use, as you will find out later on in this document.

3.4 Other Things to Look At

LaTeX can print both opening and closing quote characters, and can manage either of these either single or double. To do this it uses the two quote characters from your keyboard: `‘` and `’`. You will probably think of `’` as the ordinary single quote character which probably looks like `´` or `'` on your keyboard, and `‘` as a "funny" character that probably appears as `` ` ``. You type these characters once for single quote (see line 21), and twice for double quotes (see line 20). The double quote character `"` itself is almost never used.

LaTeX can produce three different kinds of dashes. A long dash, for use as a punctuation symbol, as is typed as three dash characters in a row, like this '`---`' (see line 23). A shorter dash, used between numbers as in '10–20', is typed as two dash characters in a row, while a single dash character is used as a hyphen.

¿From time to time you will need to include one or more of the LaTeX special symbols in your text. Seven of them can be printed by making them into commands by proceeding them by backslash (see line 36). The remaining three symbols can be produced by more advanced commands, as can symbols that do not appear on your keyboard such as †, ‡, §, £, ©, ♯ and ♣.

It is sometimes useful to include comments in a LaTeX file, to remind you of what you have done or why you did it. Everything to the right of a `%` sign is ignored by LaTeX, and so it can be used to introduce a comment.

4 Document Styles and Style Options

There are four standard document styles available in LaTeX:

`article` intended for short documents and articles for publication. Articles do not have chapters, and when `\maketitle` is used to generate a title (see Section 9) it appears at the top of the first page rather than on a page of its own.

`report` intended for longer technical documents. It is similar to `article`, except that it contains chapters and the title appears on a page of its own.

`book` intended as a basis for book publication. Page layout is adjusted assuming that the output will eventually be used to print on both sides of the paper.

`letter` intended for producing personal letters. This style will allow you to produce all the elements of a well laid out letter: addresses, date, signature, etc.

These standard styles can be modified by a number of *style options*. They appear in square brackets after the `\documentstyle` command. Only one style can ever be used but you can have more than one style option, in which case their names should be separated by commas. The standard style options are:

`11pt` prints the document using eleven-point type for the running text rather that the ten-point type normally used. Eleven-point type is about ten percent larger than ten-point.

`12pt` prints the document using twelve-point type for the running text rather than the ten-point type normally used. Twelve-point type is about twenty percent larger than ten-point.

`twoside` causes documents in the article or report styles to be formatted for printing on both sides of the paper. This is the default for the book style.

`twocolumn` produces two column on each page.

`titlepage` causes the `\maketitle` command to generate a title on a separate page for documents in the `article` style. A separate page is always used in both the `report` and `book` styles.

There is one further option which, while not standard to LaTeX, is very useful in all European countries. The `a4` option causes the output in all of the standard styles to be adjusted to fit correctly on A4 paper. LaTeX was designed in America where the standard paper is shorter and slightly wider than A4; without this option you will find that your output looks a little strange.

5 Environments

We mentioned earlier the idea of identifying a quotation to LaTeX so that it could arrange to typeset it correctly. To do this you enclose the quotation between the commands `\begin{quotation}` and `\end{quotation}`. This is an example of a LaTeX construction called an *environment*. A number of special effects are obtained by putting text into particular environments.

5.1 Quotations

There are two environments for quotations: `quote` and `quotation`. `quote` is used either for a short quotation or for a sequence of short quotations separated by blank lines:

```
US presidents ... pithy remarks:
\begin{quote}
The buck stops here.

I am not a crook.
\end{quote}
```

US presidents have been known for their pithy remarks:

> The buck stops here.
>
> I am not a crook.

Use the `quotation` environment for quotations that consist of more than one paragraph. Paragraphs in the input are separated by blank lines as usual:

```
Here is some advice to remember:
\begin{quotation}
Environments for making
...other things as well.

Many problems
...environments.
\end{quotation}
```

Here is some advice to remember:

> Environments for making quotations can be used for other things as well.
>
> Many problems can be solved by novel applications of existing environments.

5.2 Centering and Flushing

Text can be centred on the page by putting it within the `center` environment, and it will appear flush against the left or right margins if it is placed within the `flushleft` or `flushright` environments. Notice the spelling of `center`—unfortunately LaTeX doesn't understand the English spelling.

Text within these environments will be formatted in the normal way, in particular the ends of the lines that you type are just regarded as spaces. To indicate a "newline" you need to type the `\\` command. For example:

```
\begin{center}
one
two
three \\
four \\
five
\end{center}
```

one two three
four
five

5.3 Lists

There are three environments for constructing lists. In each one each new item is begun with an `\item` command. In the `itemize` environment the start of each item is given a marker, in the `enumerate` environment each item is marked by a number. These environments can be nested within each other in which case the amount of indentation used is adjusted accordingly:

```
\begin{itemize}
\item Itemized lists are handy.
\item However, don't forget
  \begin{enumerate}
  \item The 'item' command.
  \item The 'end' command.
  \end{enumerate}
\end{itemize}
```

- Itemized lists are handy.
- However, don't forget
    1. The 'item' command.
    2. The 'end' command.

The third list making environment is `description`. In a description you specify the item labels inside square brackets after the `\item` command. For example:

```
Three animals that you should
know about are:
\begin{description}
  \item[gnat] A small animal...
  \item[gnu] A large animal...
  \item[armadillo] A ...
\end{description}
```

Three animals that you should know about are:

gnat A small animal that causes no end of trouble.

gnu A large animal that causes no end of trouble.

armadillo A medium-sized animal.

5.4 Tables

Because LaTeX will almost always convert a sequence of spaces into a single space, it can be rather difficult to lay out tables. See what happens in this example

```
\begin{flushleft}
Income  Expenditure Result   \\
20s 0d  19s 11d     happiness \\
20s 0d  20s 1d      misery  \\
\end{flushleft}
```

Income Expenditure Result
20s 0d 19s 11d happiness
20s 0d 20s 1d misery

The `tabbing` environment overcomes this problem. Within it you set tabstops and tab to them much like you do on a typewriter. Tabstops are set with the `\=` command, and the `\>` command moves to the next stop. The `\\` command is used to separate each line. A line that ends `\kill` produces no output, and can be used to set tabstops:

```
\begin{tabbing}
Income \=Expenditure \=    \kill
Income \>Expenditure \>Result \\
20s 0d \>19s 11d \>Happiness   \\
20s 0d \>20s 1d  \>Misery    \\
\end{tabbing}
```

Income	Expenditure	Result
20s 0d	19s 11d	Happiness
20s 0d	20s 1d	Misery

Unlike a typewriter's tab key, the `\>` command always moves to the next tabstop in sequence, even if this means moving to the left. This can cause text to be overwritten if the gap between two tabstops is too small.

5.5 Verbatim Output

Sometimes you will want to include text exactly as it appears on a terminal screen. For example, you might want to include part of a computer program. Not only do you want LaTeX to stop playing around with the layout of your text, you also want to be able to type all the characters on your keyboard without confusing LaTeX. The `verbatim` environment has this effect:

```
The section of program in
question is:
\begin{verbatim}
{ this finds %a & %b }

for i := 1 to 27 do
  begin
  table[i] := fn(i);
  process(i)
  end;
\end{verbatim}
```

The section of program in question is:

```
{ this finds %a & %b }

for i := 1 to 27 do
   begin
   table[i] := fn(i);
   process(i)
   end;
```

6 Type Styles

We have already come across the `\em` command for changing typeface. Here is a full list of the available typefaces:

`\rm`	Roman	`\it`	*Italic*	`\sc`	SMALL CAPS
`\em`	*Emphatic*	`\sl`	*Slanted*	`\tt`	`Typewriter`
`\bf`	**Boldface**	`\sf`	Sans Serif		

Remember that these commands are used *inside* a pair of braces to limit the amount of text that they effect. In addition to the eight typeface commands, there are a set of commands that alter the size of the type. These commands are:

`\tiny`	`\small`	`\large`	`\huge`
`\scriptsize`	`\normalsize`	`\Large`	`\Huge`
`\footnotesize`		`\LARGE`	

7 Sectioning Commands and Tables of Contents

Technical documents, like this one, are often divided into sections. Each section has a heading containing a title and a number for easy reference. LaTeX has a series of commands that will allow you to identify different sorts of sections. Once you have done this LaTeX takes on the responsibility of laying out the title and of providing the numbers.

The commands that you can use are:

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

The naming of these last two is unfortunate, since they do not really have anything to do with 'paragraphs' in the normal sense of the word; they are just lower levels of section. In most document styles, headings made with `\paragraph` and `\subparagraph` are not numbered. `\chapter` is not available in document style `article`. The commands should be used in the order given, since sections are numbered within chapters, subsections within sections, etc.

A seventh sectioning command, `\part`, is also available. Its use is always optional, and it is used to divide a large document into series of parts. It does not alter the numbering used for any of the other commands.

Including the command `\tableofcontents` in you document will cause a contents list to be included, containing information collected from the various sectioning commands. You will notice that each time your document is run through LaTeX the table of contents is always made up of the headings from the previous version of the document. This is because LaTeX collects information for the table as it processes the document, and then includes it the next time it is run. This can sometimes mean that the document has to be processed through LaTeX twice to get a correct table of contents.

8 Producing Special Symbols

You can include in you LaTeX document a wide range of symbols that do not appear on you your keyboard. For a start, you can add an accent to any letter:

ò	`\‘{o}`	õ	`\~{o}`	ǒ	`\v{o}`	o̧	`\c{o}`	ó	`\’{o}`
ō	`\={o}`	ő	`\H{o}`	ọ	`\d{o}`	ô	`\^{o}`	ȯ	`\.{o}`
o͡o	`\t{oo}`	o̱	`\b{o}`						
ö	`\"{o}`	ŏ	`\u{o}`						

A number of other symbols are available, and can be used by including the following commands:

†	`\dag`	§	`\S`	©	`\copyright`
‡	`\ddag`	¶	`\P`	£	`\pounds`
œ	`\oe`	Œ	`\OE`	æ	`\AE`

Æ	`\AE`	å	`\aa`	Å	`\AA`
ø	`\o`	Ø	`\O`	ł	`\l`
Ł	`\E`	ß	`\ss`	¿	`?‘`
¡	`!‘`	...	`\ldots`	LaTeX	`\LaTeX`

There is also a `\today` command that prints the current date. When you use these commands remember that LaTeX will ignore any spaces that follow them, so that you can type '`\pounds 20`' to get '£20'. However, if you type '`LaTeX is wonderful`' you will get 'LaTeXis wonderful'—notice the lack of space after LaTeX. To overcome this problem you can follow any of these commands by a pair of empty brackets and then any spaces that you wish to include, and you will see that `\LaTeX{} really is wonderful!` (LaTeX really is wonderful!).

Finally, LaTeX 'math' mode, normally used to layout mathematical formulae, gives access to an even larger range of symbols, including the upper and lower case greek mathematical alphabet, calligraphic letters, mathematical operators and relations, arrows and a whole lot more. This document has not mentioned math mode, and it isn't going to either, so you will have to refer to the manual if you really need something that you can't get otherwise.

9 Titles

Most documents have a title. To title a LaTeX document, you include the following commands in your document, usually just after `begin{document}`.

```
\title{required title}
\author{required author}
\date{required date}
\maketitle
```

If there are several authors, then their names should be separated by `\and`; they can also be separated by `\\` if you want them to be centred on different lines. If the `\date` command is left out, then the current date will be printed.

```
\title{Essential \LaTeX}
\author{J Warbrick\and A N Other}
\date{14th February 1988}
\maketitle
```

Essential LaTeX

J Warbrick   A N Other

14th February 1988

The exact appearance of the title varies depending on the document style. In styles `report` and `book` the title appears on a page of its own. In the `article` style it normally appears at the top of the first page, the style option `titlepage` will alter this (see Section 4).

10 Letters

Producing letters is simple with LaTeX. To do this you use the document style `letter`. You can make any number of letters with a single input file. Your name and address, which are likely to be the same for all letters, are given once at the top of the file. Each letter is produced by a `letter` environment, having the name and address of the recipient as its argument. The letter itself begins with an `\opening` command to generate the salutation.

The letter ends with a `\closing` command, you can use the commands `\encl` and `\cc` to generate lists of enclosures and people to whom you are sending copies. Any text that follows the `\closing` must be proceeded by a `\ps` command. This command produces no text—you'll have to type "P.S." yourself—but is needed to format the additional text correctly.

Perhaps an example will make this clearer:

```
\documentstyle{letter}
\begin{document}

\address{1234 Avenue of the Armadillos \\
         Gnu York, G.Y. 56789}
\signature{R. (Ma) Dillo \\ Director of Cuisine}

\begin{letter}{G. Natheniel Picking \\
               Acme Exterminators \\
               Illinois}

\opening{Dear Nat,}

I'm afraid that the armadillo problem is still with us.
I did everything ...

... and I hope that you can get rid of the nasty
beasts this time.

\closing{Best Regards,}
\cc{Jimmy Carter\\Richard M. Nixon}
\end{letter}

\end{document}
```

11 Errors

When you create a new input file for LaTeX you will probably make mistakes. Everybody does, and it's nothing to be worried about. As with most computer programs, there are two sorts of mistake that you can make: those that LaTeX notices and those that it doesn't. To take a rather silly example, since LaTeX doesn't understand what you are saying it isn't going to be worried if you mis-spell some of the words in your text. You will just have to accurately proof-read your printed output. On the other hand, if you mis-spell one of the environment names in your file then LaTeXwon't know what you want it to do.

When this sort of thing happens, LaTeX prints an error message on your terminal screen and then stops and waits for you to take some action. Unfortunately, the error messages that it produces are rather user-unfriendly and not a little frightening. Nevertheless, if you know where to look they will probably tell you where the error is and went wrong.

Consider what would happen if you mistyped `\begin{itemize}` so that it became `\begin{itemie}`. When LaTeX processes this instruction, it displays the following on your terminal:

```
LaTeX error.  See LaTeX manual for explanation.
              Type  H <return>  for immediate help.
! Environment itemie undefined.
\@latexerr ...for immediate help.}\errmessage {#1}
                                                  \endgroup
l.140 \begin{itemie}

?
```

After typing the '?' LaTeX stops and waits for you to tell it what to do.

The first two lines of the message just tell you that the error was detected by LaTeX. The third line, the one that starts '!' is the *error indicator*. It tells you what the problem is, though until you have had some experience of LaTeX this may not mean a lot to you. In this case it is just telling you that it doesn't recognise an environment called `itemie`. The next two lines tell you what LaTeX was doing when it found the error, they are irrelevant at the moment and can be ignored. The final line is called the *error locator*, and is a copy of the line from your file that caused the problem. It start with a line number to help you to find it in your file, and if the error was in the middle of a line it will be shown broken at the point where LaTeX realised that there was an error. LaTeX can sometimes pass the point where the real error is before discovering that something is wrong, but it doesn't usually get very far.

At this point you could do several things. If you knew enough about LaTeX you might be able to fix the problem, or you could type 'X' and press the return key to stop LaTeX running while you go and correct the error. The best thing to do, however, is just to press the return key. This will allow LaTeX to go on running as if nothing had happened. If you have made one mistake, then you have probably made several and you may as well try to find them all in one go. It's much more efficient to do it this way than to run LaTeX over and over again fixing one error at a time. Don't worry about remembering what the errors were—a copy of all the error messages is being saved in a *log* file so that you can look at them afterwards. See your *local guide* to find out what this file is called.

If you look at the line that caused the error it's normally obvious what the problem was. If you can't work out what you problem is look at the hints below, and if they don't help consult Chapter 6 of the manual. It contains a list of all of the error messages that you are likely to encounter together with some hints as to what may have caused them.

Some of the most common mistakes that cause errors are

- A mis-spelt command or environment name.
- Improperly matched '`{`' and '`}`'—remember that they should always come in pairs.
- Trying to use one of the ten special characters `# $ % & _ { } ~ ^` and `\` as an ordinary printing symbol.
- A missing `\end` command.
- A missing command argument (that's the bit enclosed in '`{`' and '`}`').

One error can get LaTeX so confused that it reports a series of spurious errors as a result. If you have an error that you understand, followed by a series that you don't, then try correcting the first error—the rest may vanish as if by magic.

Sometimes LaTeX may write a `*` and stop without an error message. This is normally caused by a missing `\end{document}` command, but other errors can cause it. If this happens type `\stop` and press the return key.

Finally, LaTeX will sometimes print *warning* messages. They report problems that were not bad enough to cause LaTeX to stop processing, but nevertheless may require investigation. The most common problems are 'overfull' and 'underfull' lines of text. A message like:

```
Overfull \hbox (10.58649pt too wide) in paragraph at lines 172--175
[]\tenrm Mathematical for-mu-las may be dis-played. A dis-played
```

indicates that LaTeX could not find a good place to break a line when laying out a paragraph. As a result, it was forced to let the line stick out into the right-hand margin, in this case by 10.6 points. Since a point is about 1/72nd of an inch this may be rather hard to see, but it will be there none the less.

This particular problem happens because LaTeX is rather fussy about line breaking, and it would rather generate a line that is too long than generate a paragraph that doesn't meet its high standards. The simplest way around the problem is to enclose the entire offending paragraph between `\begin{sloppypar}` and `\end{sloppypar}` commands. This tells LaTeX that you are happy for it to break its own rules while it is working on that particular bit of text.

Alternatively, messages about "Underfull `\hbox'es''` may appear. These are lines that had to have more space inserted between words than LaTeX would have liked. In general there is not much that you can do about these. Your output will look fine, even if the line looks a bit stretched. About the only thing you could do is to re-write the offending paragraph!

12 A Final Reminder

You now know enough LaTeX to produce a wide range of documents. But this document has only scratched the surface of the things that LaTeX can do. This entire document was itself produced with LaTeX (with no sticking things in or clever use of a photocopier) and even it hasn't used all the features that it could. From this you may get some feeling for the power that LaTeX puts at your disposal.

Please remember what was said in the introduction: if you **do** have a complex document to produce then **go and read the manual**. You will be wasting your time if you rely only on what you have read here.

One other warning: having dabbled with LaTeX your documents will never be the same again . . . .