An Introduction to LaTeX

Michael Urban
TRW Software Productivity Project

February 15, 2019

Contents

Chapter 1

Introduction

This paper is intended to introduce you to the LaTeX[1] document preparation system, and to get you working with LaTeX as fast as possible. It is *not* a complete reference work on LaTeX. "*LaTeX: A Document Preparation System*," by Leslie Lamport (Addison-Wesley, 1985), is the definitive reference work on the subject, as it was written by the creator of LaTeX.

If you're already familiar with a text-processing program such as `Scribe` or *troff*, you can skip the rest of this introduction and go right on to Chapter 2.

LaTeX is a document preparation system. That is, you type your document into a file using a text editor using special notations (called *Control Sequences*) to describe how your document should look. LaTeX notation indicates such things as bold-face type or new chapters. When you have finished typing your document, you give the file as input to the LaTeX program, which then produces a file that can be output on a high-quality printer.

If you've used a 'word processor' before, you'll find using a document preparation system like LaTeX to be quite different. Word processors provide a great deal of feedback and are sometimes called "what you see is what you get" systems. Unfortunately, word processors are not very good at creating high-quality output with multiple fonts and sophisticated spacing unless they have a high-resolution display screen. Systems that *do* have such screens (like the Apple Macintosh or Commodore Amiga) tend to lack the power to do automatic sectioning, footnotes, tables of contents, cross-references, and the like.

[1]Pronounced either "lah-*tekh*" or "*lay*-tekh"

Furthermore, as others have pointed out, word processors tend to be "what you see is all you've got." With a system like LaTeX, a few simple changes to the input file can turn text originally intended as a short memo into a journal paper or a business letter.

Because LaTeX is a powerful program, it can automatically number sections of a document, produce indexes and cross-references, and allow you to specify 'floating' portions of a document like tables and figures. It knows about different *document styles* so that you can tell LaTeX what kind of document you're producing (article, letter, etc.) and LaTeX will automatically set the margins, the heading styles, and so on for you.

When dealing with LaTeX (or any computer program, for that matter), it's always a good idea to remember that computers are pretty stupid and usually can't figure out what you want to do unless you tell them. This means that there's a fair amount of detail that has to be mastered in order to get things to come out the way you want. Fortunately, LaTeX minimizes the number of details you absolutely need to know; simple documents are quite simple to prepare, while trickier things are more complicated in reasonable proportion to their complexity.

As for the method of the book, although it be not such as may in every part satisfy the curiosity of dichotomists, yet is it such as I thought most convenient for the capacity of the learner.

— THOMAS MORLEY, *A Plaine and Easie Introduction to Practicall Musicke*

Chapter 2

Getting Started

In which we learn what we need to know to produce our first LaTeX document.

2.1 General Operation

To produce a well-formatted document using LaTeX, there are 3 steps to be followed:

1. Prepare an input file in the LaTeX notation that describes what your document is supposed to look like. This description will include the *document style*, the contents of the document, changes in type style, and so on.

2. Process your input file with LaTeX. This will produce a so-called `dvi` (or *device-independent*) file as output.

3. Print the `dvi` file on the high-quality printer (laser printer or phototypesetter) that's available on your system.

That's all there is to it. If LaTeX complained that it didn't understand your input, or if the document didn't come out looking the way you wanted, you go back and edit the input file and do the steps over until everything's just right.

2.2 The Input File

Your input for LaTeX is just a normal text file that you can create with your favorite text editor. The name of the file should be *something*`.tex`, i.e., you can call it anything you like so long as the name ends with "`.tex`". The examples in this paper will assume the file is named `mydoc.tex`.

The first line of the input file **must** be something that looks like this:

```
\documentstyle{article}
```

The word `article` on this line can be replaced by `report`, `letter`, or any other document style available on your system. This line tells LaTeX just what sort of document you're producing, so that LaTeX can set up the margins, heading style, and so on. `article` is a good document style for most simple documents.

The body of the document must be surrounded by two special lines:

```
\begin{document}
```

must precede the body of the document, and (as you might have suspected),

```
\end{document}
```

is the last line of the document. Thus, a kind of minimalist LaTeX input file might appear as follows:

```
\documentstyle{article}
\begin{document}
I am a sentence.
\end{document}
```

In Chapter 4, we'll see other things that can be done with `\begin` and `\end`.

A simple text document is just a series of words, sentences and paragraphs. Words are separated by one or more blanks and/or tab characters, or the end of the input line. In fact, the number of blanks that are used to separate words is unimportant; ten blanks have exactly the same effect as one.

Sentences end with punctuation marks like periods and question marks, and are separated by blanks or the end of the input line. Remember, the number of blanks makes no difference; LaTeX knows about periods and adds the extra space after the period automatically.

LaTeX allows you to separate paragraphs easily: just put an extra blank line between paragraphs. This blank line is, by the way, just about the only case in which LaTeX cares about the end of a line. In all other cases, the end of a line has exactly the same significance as a space — it just separates words.

You're just about ready to produce a simple document. But beware! There are ten special characters that are part of LaTeX's special notation:

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

The first six of these (which are the most common in regular copy) can be placed in the document by preceding them with the back-slash character. For example, the dollar sign can be placed harmlessly in your text by typing "`\$`". Find that back-slash character on your terminal and get used to typing it; it's an important part of LaTeX's special notation. The other characters take a little more work, but we'll talk about that later.

2.3 Running LaTeX

Let's say you have the following input file, named `mydoc.tex`:

```
\documentstyle{article}
\begin{document}
But last week, humor turned to alarm when the group, acting
through a local front called the Age of Enlightenment Foundation,
bought the financially beleaguered University of the East, one of
the largest private universities in Asia with three campuses and
47,000 students.

Officials of the Education Ministry then disclosed that the
maharishi's people had made overtures to buy at least half a
dozen other debt-ridden educational institutions.  The deputy
minister of education said the group had brought millions of
dollars into the country and was keeping the money in three local
banks.
\end{document}
```

The blank line ends the previous paragraph and starts a new one. To process this file, we run LaTeX. The exact method for doing this will vary from computer to computer. On UNIX systems, the command would be:

```
        $ latex mydoc
```

The dollar sign is the prompt for the Shell. Notice that while the file name is `mydoc.tex`, we only have to say `mydoc` here; LaTeX infers the `.tex` part itself. Now LaTeX processes the file and types information on what it's doing at your terminal. For this example, you'll see something like this:

```
This is TeX, Version 1.1 for Berkeley UNIX (preloaded format=lplain 84.8.13)
(mydoc.tex
LaTeX Version 2.06a - Release 7 July 84
(/usr/lib/tex/macros/article.sty
Document Style 'article'. Version 0.91 - released 25 June 1984
(/usr/lib/tex/macros/art10.sty))
No file mydoc.aux.
[1] (mydoc.aux)
Output written on mydoc.dvi (1 page, 948 bytes).
Transcript written on mydoc.log.
```

The first line (`This is TeX`...) tells which version of TeX is in use. TeX is the basic text-processing system which is the basis of LaTeX; this distinction should not concern you now, but helps when you're trying to decipher error messages as described in Chapter 9. In the next line, you can see that `mydoc.tex` is being read. The next four lines describe the version of LaTeX being used, and the various files being read that comprise the `article` document style.

The message, "`No file mydoc.aux`" is a harmless warning; it simply means that you haven't run this document through LaTeX before. LaTeX uses `mydoc.aux` to retain cross-reference and Table-of-Contents information.

The appearance of "1" in square brackets indicates that Page 1 has been processed. When the input file has been read, `mydoc.aux` is re-read to see if any of the cross-reference or Table of Contents information has changed. You don't have to worry about this yet.

The last output lines tell where the results are. The device-independent representation of your document are kept in a file called `mydoc.dvi`, which contains one page of output and occupies 948 bytes of disk space (if you don't know about bytes and the like, don't worry about it). A transcript of all this stuff is kept in a file called `mydoc.log` in case you want to look at it later. If you had received a bunch of error messages, you might well want to have this transcript around.

Finally, when you print the results, you'll get output much like the following[1] (on its own page, with the page number '1' at the bottom):

> But last week, humor turned to alarm when the group, acting through a local front called the Age of Enlightenment Foundation, bought the financially beleaguered University of the East, one of the largest private universities in Asia with three campuses and 47,000 students.
>
> Officials of the Education Ministry then disclosed that the maharishi's people had made overtures to buy at least half a dozen other debt-ridden educational institutions. The deputy minister of education said the group had brought millions of dollars into the country and was keeping the money in three local banks.

Actually, the output here has been formatted slightly differently from the way it would actually appear on a page, and is presented here as a 'quotation'. We'll see how this `quotation` facility works in Chapter 4.

2.4 Output Devices

LaTeX was designed to produce high-quality output on such 'devices' as laser printers and phototypesetters. This allows it to use sophisticated methods for breaking up lines, paragraphs, and pages in the best possible manner. The device-independent (`dvi`) file produced by LaTeX contains all the information describing the contents of every square inch of the page for an imaginary printer with a resolution near the wavelength of visible light. The price of sophistication, in this case, is that you can't print your document on a 'dumb' device like a terminal or line printer. However, graphics devices like the Sun Microsystems workstation permit pre-viewing of LaTeX output.

Each installation will have its own output device (or devices), and its own method for printing the `dvi` file on that device. Some of the devices for which LaTeX output is supported include the QMS Lasergrafix printer, the Imagen Imprint-10 laser printer, and the DEC LN01S laser printer. The TRW Software Productivity Project, for example, uses a program called *iptex*

[1]From the Los Angeles Times, Oct. 12, 1984.

to print `dvi` files on the Imagen Imprint-10 and Imagen 5/480 laser printers. In our example, the output might be produced by typing the Shell command:

```
$ iptex mydoc
```

Notice that *iptex*, like LaTeX, infers the name `mydoc.dvi` from the name `mydoc`.

You now know enough to produce many kinds of simple documents using LaTeX. However, you'll soon discover that some of the things you thought you knew all about (like quotation marks) don't seem to work just the way you'd expect. This is because a high-quality document has a lot more characters than you have keys on your keyboard. To find out how to handle some of these characters, read on ...

2.5 Typesetting versus Typing

LaTeX was designed to produce the sort of type that you find in books. If you take a moment to look at a professionally-printed book, you'll discover quite a few differences between book printing and the sort of thing you do at a typewriter (or on a computer terminal).

2.5.1 Quotation Marks

For one thing, the quotation marks (both single and double) in books go in two directions. “See Spot run.” See? Your terminal only has one kind of double-quote mark, so it's not too useful for this purpose. On the other hand, it probably has two different kinds of single-quote marks: the normal apostrophe that you're probably accustomed to (this is oriented as a ‘closed-quote’ mark, i.e., ´), and the so-called ‘accent gràve’ character (oriented as an ‘open-quote’ mark, i.e., `). So to typeset something like ‘this’, you would use the left and right single-quote marks:

```
`this'
```

You can produce the double-quote marks simply by typing two single-quote marks. To set “See Spot run” you would use the input:

```
``See Spot run''
```

2.5.2 Dashes

Your terminal keyboard has only one kind of dash (or hyphen) on it. But a well-printed book will use different length symbols for different kinds of dashes.

The shortest dash is the hyphen, found in compounds like “well-printed”. To set a hyphen, just use the regular hyphen (dash?) key on your terminal.

The next longer dash is called an “en-dash” because it is just about as wide as the letter “N”. It is used for number ranges like “pages 13–15” or in places like “Clause 5.4–3”. Type two hyphens to produce an en-dash.

The longest dash is called an “em-dash” and is about as wide as the letter “M”—it’s the dash used in English prose. Type three hyphens to produce an em-dash.

As an example, the sentence, “The en-dash—you know what it is—is discussed on pages 14–15” is produced by the input,

```
The en-dash---you know what
it is---is discussed on pages 14--15
```

2.5.3 Ligatures

Actually, you don’t have to worry about ligatures, but it’s nice to know that LaTeX automatically takes care of things like printing the letters `f` and `i` in the word “find” as a single unit, the way the pros do it. These ligatures like “ﬁ”, “ﬃ”, “ﬂ” and “ﬄ” improve the appearance of printed text substantially. LaTeX also causes combinations of letters like `A` and `V` to be moved together to improve their appearance.

2.5.4 Line Breaking

LaTeX automatically breaks a paragraph up into lines, but there are some places where you don’t want this to happen, like after the word “Chapter” in the phrase, “Chapter 5.” To prevent this, use the tilde character (`~`) instead of a space. For example, type

```
Chapter~5
```

whenever you have to cite a chapter from a book.

2.5.5 Other Keyboard Characters

There are still other non-alphanumeric characters found on the keyboard that deserve some attention. The characters '@', '*', '+', and '=' are all produced correctly from the corresponding keyboard keys. On the other hand, the characters '`<`', '`>`', and '`|`' are not normally used in regular text copy, and should only be used in mathematics mode (discussed in Chapter 8). If you use them while producing output in normal text, you'll get the characters '¡', '¿', and '—' instead. While the first two of these are useful when typesetting Spanish text, LaTeX allows you to type them more conveniently as `!`‘ and `?`‘ respectively.

2.6 What Next?

You now know everything about creating, processing, and printing simple text documents using LaTeX. Pick up a book and copy a few paragraphs from the book into a file in LaTeX notation—find something with lots of quotations and dashes. Don't forget about the special characters mentioned in Section 2.2. Now try to get some output. If you get errors from LaTeX, type a question mark, and you will get some guidance as to what to do next. Further information on errors and error correction will be presented in Chapter 9. You may want to read through Chapter 9 now, although you might not yet understand all the details.

When you're ready to type more complex documents—with changes of type style, numbered sections, and the like—turn to the next chapter.

The further you go, the less easy will it be to withdraw; yet no oath or bond is laid on you to go further than you will.

— J. R. R. TOLKIEN, *The Fellowship of the Ring*

Chapter 3

Control Sequences

In which we learn how to direct LaTeX in more complex matters.

3.1 The Escape Character

By now, you've probably guessed that the backslash character (\) is very important to LaTeX. In LaTeX terminology, this character is the *escape* character. This simply means that your input is "escaping" from its usual text-setting operation in order to tell LaTeX about something special. **Note**: most computer terminals have a key marked `ESC`. This key has *nothing* to do with the escape character, and will *never* be used in a LaTeX input file.

Immediately following the escape character is a special sequence of characters that tells LaTeX what you want to do. Such commands are called *control sequences*. You've already seen the two basic types of control sequences:

- A *control word* consists of an escape character followed by one or more alphabetic characters. The first **non**-alphabetic character (be it a numeral, punctuation, or space) terminates the control word. An example of a control word is `\begin`.

- A *control symbol* consists of an escape character followed by exactly one **non**-alphabetic character. For example, the control symbol `\$` is used to place a dollar sign in the output.

After any control word (not symbol), a blank (or group of blanks, or the end of a line) is completely ignored; it serves only to terminate the control word.

3.1.1 Some Control Symbols

We've already seen that some control symbols like `\%` and `\&` are used to print characters like '%' and '&' (which would otherwise be significant to LaTeX). Many of LaTeX's control symbols are used to place accents on letters. For example, to generate the phrase "Piña Colada" I typed "`Pi\~na Colada`". The French word "parlé" is typed as "`parl\'e`". When you place an accent on the letter `i` or `j`, you have to get rid of the dots, so the control words `\i` and `\j`, representing dotless 'ı' and 'ȷ', can be used. For example, "manĝaĵo" is typed as "`man\^ga\^\j o`."

One special control symbol, `\\`, is used to terminate a line and begin a new one. For example,
this line was initiated by the input,

```
For example,\\this line was
```

Normally, `\\` is used in special places like titles.

3.1.2 A Control Word Example

The control word `\today` places today's date—the date on which the file was run through LaTeX—in the output. For example, "This document printed February 15, 2019" was produced by the input,

```
``This document printed \today''
```

Note that the first closing quote symbol ends the `\today` control word. Suppose we want to type, "February 15, 2019 is a nice day." We can't just type

```
\today is a nice day
```

because the blank after `\today` only serves to end the control word; it has no other function. So the output would be "February 15, 2019is a nice day." What to do? Well, you can terminate the control word with something other than a blank. Something harmless. In this case,

```
\today{} is a nice day
```

will do the job. The blank following the braces acts to separate words in the usual way. We'll see just *why* the pair of braces is harmless in Section 3.2.

3.1.3 Changing the Type Style

Let's look at some specific control words for changing the type style. These are the commands that produce **bold** or *slanted* type, or rather small type where needed.

`\rm` Causes subsequent output to appear in Roman (normal) type.

`\it` Switches further output to *italic (like this)* type.

`\bf` Switches to **bold (like this)** type.

`\sl` Switches to *slanted (like this)* type.

`\sf` Switches to sans serif (like this) type.

`\sc` Switches to SMALL CAPS (LIKE THIS) type.

`\tt` Switches to `typewriter (like this)` type.

`\em` Switches to an emphatic style: italic if currently roman, otherwise roman.

When you switch from a slanted type face to an unslanted font, you should add the control symbol `\/`, which adds a little extra space. Otherwise, you might set a word like *did*n't as *did*n't, in which the 'd' and 'n' are too close together. One correct form is

```
        \it did\/\rm n't
```

but see Section 3.2.

Printers, and text-processors, specify type size in points. Ordinary books are in ten-point type, but this paper is in twelve-point type. If you're curious, this means that the capital letter 'M' is supposed to be twelve points wide; a printer's point is $\frac{1}{72.27}''$. Anyway, you can switch to a different type size using one of the following control words:

`\normalsize` Further output appears in the normal type size (ten points in a ten-point paper).

`\large` Output appears in a larger size (twelve points in a ten-point paper).

`\Large` Larger still (fourteen points).

`\LARGE` Even larger (seventeen points).

`\huge` Twenty points in a ten-point paper.

`\Huge` Twenty-five point type.

`\small` Smaller-than-normal output (nine points in a ten-point paper).

`\footnotesize` Still smaller (eight points). Footnotes are printed in this size.

`\scriptsize` Smaller still (seven points). The size normally seen in mathematics superscripts.

`\tiny` Smallest; five-point type, for ten-point documents.

Whenever you switch type sizes, you're automatically switched to roman-style letters. So to type *small italic commentary*, you should type

```
\small\it small italic commentary
```

rather than doing the `\it` first.

Note that not every machine with LaTeX will have every style of type in every size available for its own printer. However, one can expect to find most of the sizes available for the roman, italic, and bold type styles. LaTeX should print a warning if you try to use a type size that is not available.

3.2 Groups

To see how groups work, and why one wants to use them, consider the sentence, "The **bold** Roman marched home." The control word `\bf` switches to bold-face type, and `\rm` switches to the (normal) roman lettering, so you might type,

```
The \bf bold \rm Roman marched home.
```

But there's an easier way to do it. When any LaTeX input is placed between curly braces, it takes place within its own *environment*, a kind of world of its own. Any changes that occur within this environment, such as changes in type style, are un-done when that environment finishes. So the easier way to type the example would be

```
The {\bf bold} Roman marched home.
```

because the change to bold-face type takes place within its own environment and doesn't affect what happens outside the curly-brace boundaries.

3.3 Parameters

Newspaper and television writers (and a lot of other people) use the word "parameter" to mean "limit" or "boundary," but this usage is incorrect. The actual meaning is "variable," i.e., something that changes from one occurrence to the next. Many LaTeX control words require that a parameter be supplied. For example, as we've seen, the `\documentstyle` control word requires that you specify just **which** document style you're selecting. In LaTeX, such parameters are placed between curly braces—a second use for the braces besides grouping.

We'll see later that some LaTeX commands also have *optional* parameters supplied (if at all) inside square brackets. But before we delve further into the details of LaTeX's grammar, we'll turn to a pair of control words (which require parameters) that you'll be using a lot.

Words like "abracadabra" or "osorronnophris" or "shehamphorash" may have little or no meaning to those saying them, but their meaninglessness is irrelevant.

— P. E. I. BONEWITZ, *Real Magic*

Chapter 4

`begin` and `end` Environments

In which we learn about specially-formatted sections of text.

Back in Section 2.3, we displayed the sample LaTeX output using narrower margins. This was a `quotation` and was accomplished by establishing an environment in which several aspects of page layout (left margin, right margin, paragraph indentation) were changed. Remember, these environments are just like the ones you get when you put something inside the curly braces; any changes in the style or size of type that happen inside the environment are undone when the environment finishes up.

You can make such an environment by using a command like

```
\begin{quotation}
```

and ending it with `\end{quotation}`. You've seen these `\begin` and `\end` commands before; your entire LaTeX document occurs within an environment called `document`. In other words, the environment to be bounded by the `\begin` and `\end` 'brackets' is supplied as a parameter when those control words are used. Notice that you will get an error if you try to do something like:

```
\begin{something}
\begin{somethingelse}
\end{something}
\end{somethingelse}
```

That is, environments cannot overlap; you must `\end` the ‘inner’ environment before you end the ‘outer’ one. Many people, while editing the input file, will type both the `\begin` and `\end` commands first, and then back up and insert whatever they are doing in between the two lines in order to guarantee that everything matches up OK.

Let’s look at some of the more useful formatting environments.

4.1 The `quote` Environment

The `quote` environment causes text to be set somewhat narrower than normal. The paragraphs of the quote are unindented and are separated by a little blank space. For example:

```
\begin{quote}
\small
Cardinal Jaime L. Sin, the Roman Catholic archbishop of Manila,
warned that the group had the trappings of a cult, and convened a
conference of educators, scientists, theologians and philosophers
to study its potential impact.

Meanwhile, Parliament ordered an investigation of the group's
activities after some members expressed fear that a foreign
ideology was being foisted on this poverty-stricken Roman
Catholic country by people trying to buy property at fire-sale
prices at a time of economic distress.

\end{quote}
```

would produce:

> Cardinal Jaime L. Sin, the Roman Catholic archbishop of Manila, warned that the group had the trappings of a cult, and convened a conference of educators, scientists, theologians and philosophers to study its potential impact.
>
> Meanwhile, Parliament ordered an investigation of the group’s activities after some members expressed fear that a foreign ideology was being foisted on this poverty-stricken Roman Catholic country by people trying to buy property at fire-sale prices at a time of economic distress.

Long quotes sometimes look better in a smaller size of type, as in this example. It's a good idea to end the last paragraph of a quote with a blank line before leaving the `quote` environment, especially if you've changed the type size within the `quote` environment[1]. Notice the extra spacing before and after the quote. Also notice that the change to the `\small` size type is un-done when the `quote` environment ends.

4.2 The `quotation` Environment

The `quotation` environment is just like the `quote` environment, except that each paragraph is indented:

> Cardinal Jaime L. Sin, the Roman Catholic archbishop of Manila, warned that the group had the trappings of a cult, and convened a conference of educators, scientists, theologians and philosophers to study its potential impact.
>
> Meanwhile, Parliament ordered an investigation of the group's activities after some members expressed fear that a foreign ideology was being foisted on this poverty-stricken Roman Catholic country by people trying to buy property at fire-sale prices at a time of economic distress.

4.3 The `verse` Environment

The `verse` environment is used to set poetry, and maybe for other applications. Basically, it is a paragraphing environment with narrow output like `quote`, but lines that continue onto a second output line receive extra indentation.

```
\begin{verse}
        There once was a poet named Dan,\\
        Whose limericks never would scan.\\
        When told this was so,\\
        He said, ``Yes, I know,\\
```

[1]This is because otherwise the paragraph isn't divided into lines until *after* the environment closes, and by then the change in line spacing has been un-done. A little weird, admittedly, but just follow the rule and you won't go wrong.

```
        It's because I try to put every syllable into the
        last line that I possibly can.''
\end{verse}
```

would produce the following:

> There once was a poet named Dan,
> Whose limericks never would scan.
> When told this was so,
> He said, "Yes, I know,
> It's because I try to put every syllable into the last line that I
> possibly can."

4.4 The `verbatim` Environment

The examples of LaTeX input in this paper have generally been entered using the `verbatim` environment. In this environment, all input is presented in the typewriter type-face, and all new-lines and blanks are reproduced exactly as they appear in the input. Furthermore, **all** LaTeX command sequences are treated as ordinary text in this environment; the only line you *can't* put in the `verbatim` environment is the `\end{verbatim}` line which signifies the end of that environment.

```
\begin{verbatim}
\small\it This is \today.
\end{verbatim}
```

produces:

```
\small\it This is \today.
```

Notice that verbatim output is not automatically indented—you have to add extra spaces at the beginning of each line (or put the `verbatim` environment inside a `quote` environment) if you want to indent.

LaTeX also permits you to put a single "verbatim"-type word within a paragraph by using the control word `\verb`. Because `\verb` sort of 'turns LaTeX off' as far as special characters go, it has a different grammar from all the other control sequences. The *first* character following the control word becomes the bracketing character for the parameter, and all subsequent characters (including the escape character) are output in typewriter font until the bracketing character is seen again.

For example, I typed "`\verb`" by typing

```
\verb|\verb|
```

using the vertical bar as my bracketing character, although any other character would have worked just as well.

4.5 The `center` Environment

For titles and the like, it's useful to have an environment in which each line is centered on the page, rather than being justified. The `center` environment provides this. For example, the input:

```
\begin{center}
{\bf Bitts and Bights}\\
{\small\it A Novel}
\end{center}
```

would produce the following:

Bitts and Bights
A Novel

Note that the line separation has to be explicitly typed (as `\\`). If you have a lot of text input without line breaks, you'll end up with a paragraph, each line of which is centered:

At first, many Filipinos seemed amused, even curious, about the
invasion of free-spending foreigners in suits and granny dresses
conducting seemingly nonstop symposia and spouting bizarre
scientific-sounding jargon. An example: "All possible local
gauge-invariant operators are generated by non-perturbative
quantum gravitational effects at the Planck scale."

4.6 `flushleft` and `flushright`

Sometimes you need to produce something like this:

<div align="right">Samuel Klugarsh
666 Clarkle Street
Hogboro, NJ 11780</div>

in which each line is flush right with the right-hand margin. The `flushright` environment allows this. The input in this case was:

```
\begin{flushright}
Samuel Klugarsh\\
666 Clarkle Street\\
Hogboro, NJ 11780
\end{flushright}
```

Once again, each line break should be explicitly specified, or you'll get a paragraph in which each line has a "ragged" left margin.

On the other hand, sometimes a "ragged" right margin *is* desirable. In this case, the `flushleft` environment can be used. For example,

> "Honestly," said Helena Benitez, the independent assemblywoman
> who spearheaded the legislative investigation, "it looks to me like
> either an invasion from Mars or an invasion of locusts that have
> come to feast on the grain when it's ripe."

4.7 Nesting Environments

As hinted earlier, you can place environments inside one another. For example, the last quotation was input as:

```
\begin{quote}
   \small
   \begin{flushleft}
   ‘‘Honestly,’’ ...

   \end{flushleft}
\end{quote}
```

The extra indentation of the `flushleft` environment in the input file doesn't affect the output at all, since LaTeX ignores extra blanks; it's just there to improve the readability of the input.

> *This is a good place to stop and get in some practice. Put together a short LaTeX document with a centered, large, boldface title, a ragged-right paragraph, and a small italic quotation, just to get the feel of all this stuff. When you're pretty confident with all these curly braces and begins and ends, go on to the next section.*

4.8 List-Making Environments

Most documents include one or more lists of things. LaTeX provides environments that allow one to easily format three kinds of lists:

- Itemized lists (like this one).
- Enumerated lists, in which each item is numbered consecutively.
- Description lists, in which each item has a text label instead of a bullet or number.

4.8.1 Itemized Lists

The `itemize` environment is used to produce itemized lists. For example, the list above was generated with the following input:

```
\begin{itemize}
\item   Itemized lists (like this one).
\item   Enumerated lists, in which each item is numbered
        consecutively.
\item   Description lists, in which each item has a text
        label instead of a bullet or number.
\end{itemize}
```

The alert reader has noticed that each item in the list is preceded by the control word `\item`. In particular, the first thing in an `itemize` (or other list-making) environment *must* be an `\item` or an error will result. There are no other restrictions. For example a new paragraph will be indented at the same level as the "bulleted" paragraphs, so everything will look neat.

Because LaTeX ignores extra spaces, the input can be formatted neatly, as in this example, to improve readability. This sort of thing helps a lot when you're fixing things later.

4.8.2 Enumerated Lists

The `enumerate` environment is much like the `itemize` environment, except that each `\item` in the list is automatically numbered consecutively. For example, if I had used `enumerate` instead of `itemize` in the previous example, it would have appeared as follows:

1. Itemized lists (like this one).

2. Enumerated lists, in which each item is numbered consecutively.

3. Description lists, in which each item has a text label instead of a bullet or number.

4.8.3 Description Lists

The `description` environment is used to produce lists like the list of type-style-changing control words back in Section 3.1.3. In such a list, each `\item` has a specific "hanging" label. For this, the label is specified as an *optional parameter*. These are similar to the parameters we've already looked at, but because they don't **have** to be present, they are surrounded by square brackets instead of curly ones.

Let's look at a specific example. Suppose we want to produce the following list:

`Ada` The Countess of Lovelace. She was the paramour of Charles Babbage, and was probably the world's first programmer.

`Pascal` Mathematician and philosopher, Blaise Pascal is probably best known for his triangular representation of binomial expansion.

`Euclid` Greek mathematician who gave his name to the geometry of the plane, one of the few branches of ancient science still taught.

The first `\item` command must specify "`Ada`" as the hanging tag. So "`\tt Ada`" will be the parameter for `\item` and in this case the `\item` command will be

```
\item [\tt Ada]
```

Note that the change to typewriter font happens inside a separate environment because a parameter (optional or mandatory) is automatically placed inside its own group. Here's the entire specification:

```
\begin{description}
\item[\tt Ada] The Countess of Lovelace.  She was the paramour
        of Charles Babbage, and was probably the world's first
        programmer.
\item[\tt Pascal] Mathematician and philosopher, Blaise Pascal
        is probably best known for his triangular
        representation of binomial expansion.
\item[\tt Euclid] Greek mathematician who gave his name to
        the geometry of the plane, one of the few branches
        of ancient science still taught.
\end{description}
```

Many other LaTeX control sequences permit such optional parameters to be specified within square brackets. For example, the `\documentstyle` control word allows one to optionally specify a list of style options as optional parameters. The first line of the file that produced this paper was

```
\documentstyle[12pt]{report}
```

which caused this paper to be produced in twelve-point type. Note that the optional parameter must be in a particular place. For `\documentstyle`, the optional parameter precedes the mandatory parameter (the document style name `report` in this example). For other control words (such as `\begin`) an optional parameter might follow the mandatory parameters.

4.8.4 Nesting Lists

Many lists are 'hierarchical' and contain several levels of sub-lists. LaTeX automatically handles this whenever you nest list-making environments inside one another. For example, the following output:

1. Commands
    (a) Editor commands
        i. Reading
        ii. Writing
        iii. Cursor movement
    (b) Mail Commands
        i. Composing mail
            A. Editing the file
            B. Mail headers
        ii. Delivering mail
    (c) Communication commands
2. System calls
3. Subroutines

is quite a complex-looking list, but is really just a series of `enumerate` environments, nested inside one another:

```
\begin{enumerate}
 \item Commands
 \begin{enumerate}
  \item Editor commands
  \begin{enumerate}
    \item Reading
    \item Writing
    \item Cursor movement
  \end{enumerate}
  \item Mail commands
  \begin{enumerate}
    \item Composing mail
    \begin{enumerate}
      \item Editing the file
      \item Mail headers
    \end{enumerate}
    \item Delivering mail
  \end{enumerate}
  \item Communication commands
 \end{enumerate}
 \item System calls
 \item Subroutines
\end{enumerate}
```

Note that the different numbering styles associated with the different levels are handled automatically by LaTeX. Similarly, nested `itemize` environments produce different hanging tags (bullets at level 1, dashes at level two, and so on) and indentation levels for the different nesting levels.

> *So far, you know how to prepare LaTeX documents with different sizes and styles of type, quotations, verbatim input, ragged left and/or right margins, centered material, and lists with different kinds of tags. This is another good place to stop and practice. The next chapter will describe how to put a complete paper together with numbered sections, footnotes, title page, and table of contents.*

"That the environment should respond to human thought. That is the core of magic and the oldest dream of mankind; and here, on me, it is fact."

— GENE WOLFE, *The Death of Doctor Island*

Chapter 5

Putting It Together

In which we learn how to put together a beautifully formatted paper like this one.

LaTeX provides simple-to-use commands that facilitate the sort of niceties found in technical papers such as automatically numbered sections and footnotes, and a table of contents.

5.1 Numbered Sections

The way a document is broken up into sections depends on the document style. The `report` document style that was used to produce this manual divides the document up into Chapters, Sections, Subsections, and (although I haven't used them in this report) Sub-sub-sections, Paragraphs, and Sub-Paragraphs. The `article` document style is similar, but the major division is the Section—there are no Chapters in the `article` style.

A new section in a paper is created in the obvious way: this chapter began with the line

```
\chapter{Putting it Together}
```

and this section began with the line

```
\section{Numbered Sections}
```

LaTeX automatically numbers each section within its containing section, so you can add sections or subsections as you build up the document without worrying about renumbering.

Later on, we'll see how a Table of Contents is created, but for now we'll observe that you sometimes want to have a chapter or section heading that is different from what appears in the Table of Contents entry. In this case, an optional parameter for the sectioning command gives the data that will appear in the Table of Contents. For example, if I began a sub-subsection with the line,

```
\subsubsection [Crustaceans] {Spiders and Crabs}
```

then the heading of the sub-subsection would read "Spiders and Crabs", but the Table of Contents entry for it would read simply "Crustaceans".

5.2 Footnotes

There has probably never been a technical paper of any kind that appeared without footnotes, so LaTeX was designed to make footnoting easy. By now, you're pretty familiar with the way LaTeX does things, so it should come as no surprise that this footnote[1] was produced by the following input line:

```
that this footnote\footnote{which tells you nothing} was
```

Footnotes are automatically numbered within a document (within each chapter, in the `report` document style), so you don't have to worry about explicitly providing a footnote mark.

5.3 The Table of Contents

Nothing could be easier than producing a Table of Contents. Just put the control word "`\tableofcontents`" in your document at the place you want the Table to appear. This will probably be on a new page following your title page. You can cause LaTeX to force a new page with the control word "`\newpage`".

LaTeX will format the Table of Contents based on the information derived from the **previous** time the document was run through LaTeX. A moment's consideration will help you understand why this is so: LaTeX doesn't

[1]which tells you nothing

know which page a given section or sub-section begins on until that page has actually been printed. So it retains this information in a file with a name like `mydoc.aux` and rereads this file on the next run when it sees the `\tableofcontents` command.

5.4 The Title

LaTeX allows you to specify the title and author of the document. As you might expect, you might begin your paper with

```
\title{Advanced Lessons\\
       in\\
       Hyperstellar Archaeology}
\author{S. Klugarsh}
\date{July 4, 2000 AD}
```

These commands actually don't output anything! But they provide information that is used by yet another LaTeX control word, `\maketitle`. In the `article` document style, the title block (automatically centered and spaced) goes on the first page of the document; in the `report` style it goes on a separate page, and a new page immediately follows, so you can immediately follow a `\maketitle` with a `\tableofcontents` if you like. If you use `\maketitle`, you **must** provide a title and author, even if you only say

```
\author{}
```

If you don't provide a `\date`, `\today` is used.

Also, you can have the title on a separate page in the `article` document style if you use the `titlepage` option in the `\documentstyle` option list, e.g.

```
\documentstyle [11pt,titlepage] {article}
```

5.5 Page Numbers

It is desirable to have the table of contents numbered differently from the main body of the paper, usually with small roman numerals instead of the usual arabic. To accomplish this, you can use the `\pagenumbering` command, which changes the style in which the current (and subsequent) pages are to be numbered, resetting the page number to '1' as it does so. It takes a single parameter which describes the numbering style; `roman` causes lowercase roman numerals to appear, and `arabic` uses normal arabic numerals.

5.6 An Example

The first several lines of the input file that produced this document were:

```
\documentstyle[12pt,twoside]{report}
\title{\Large\bf An Introduction to \LaTeX}
\author{\normalsize Michael Urban\\
        \it TRW Software Productivity Project}
\date{\normalsize\today}
\begin{document}
\maketitle
\pagenumbering{roman}
\tableofcontents
\chapter{Introduction}
\pagenumbering{arabic}
This paper is intended to introduce you to the \LaTeX\footnote
{Pronounced either ``lah-{\it tekh\/}'' or ``{\it lay\/}-tekh''}
document preparation
```

The only new item here is the `twoside` option which sets the margins correctly for two-sided copying. A complete list of the various options which can be used to set various aspects of the document's appearance appears in the LaTeX documentation but is beyond the scope of this introduction. A supplementary volume describing the different ways to control the appearance of LaTeX output is planned.

5.7 Appendices

Rather than providing a separate series of sectioning commands for appendices and their subsections, LaTeX lets you start a set of appendices with the control word `\appendix`. The effect of `\appendix` is to cause subsequent major headings (i.e., Chapters if you're doing a `report` and Sections for an `article`) to refer to numbered appendices rather than major sections. For example, if I were to say,

```
\appendix
\chapter{List of Fonts}
\chapter{Names of Suppliers}
```

then the two new chapters would be headed as "Appendix A" and "Appendix B" and would be so listed in the Table of Contents. Once you've done this, of course, it's not particularly easy to go back to regular chapter numbering (and how you **can** do it is beyond the scope of this paper).

5.8 Comments

Sometimes you want to add a comment to the input file for your own benefit while you're working on the document, but not to be output by LaTeX. In this case, you can precede the comment with a percent sign (`%`). Anything that follows a percent sign on a line of input, up to and including the end of the line, is ignored by LaTeX. So your input file might include something like:

```
As everyone knows, there are ?? % LOOK THIS UP!
artificial elements in the periodic table.
```

You now know how to produce a complete paper using LaTeX. The remaining material in this paper is "optional" stuff that talks about things like figures and tables, tabular output, and mathematics. If you can get along without these things for a while, stop reading now and start producing beautifully formatted papers.

Sheets were combined into books by gumming the right edge of one sheet to the left edge of the next; in this way many rolls were produced which were sometimes forty yards in length; they were seldom longer, for there were no verbose historians in Egypt.

— WILL DURANT, *The Story of Civilization: Our Oriental Heritage*

Chapter 6

Tables and Figures

In which we learn how to set up tabular information and let it "float" to a convenient spot.

6.1 Floating Bodies

Table 6.1 was actually typed in between the figure '6.1' and the word "was" in this sentence. It managed to migrate to the bottom of the page because the specification for the table was enclosed in a special environment (the `table` environment) which automatically moves to a convenient place where there is room.

Letter	*Meaning*
h	here
b	bottom
t	top
p	float page

Table 6.1: Figure/Table Destinations

6.1.1 The `table` Environment

The `table` environment causes its contents to be placed in a separate area from the surrounding text. The exact location in which the table ends up depends on the amount of space currently available on a page, and on an optional parameter supplied with the `\begin{table}` control sequence. Table 6.1 shows the options available.

`h` Causes the table to be placed at the current line if possible.

`b` Causes the table to be placed at the bottom of a page of text, if possible.

`t` Causes the table to be placed at the top of a page of text, if possible.

`p` Causes the table to appear on a separate "float" page reserved for it.

Several of these placement options can appear at once. For example,

```
\begin{table}[hb]
```

specifies that the table may appear at the current text position or at the bottom of a text page (possibly the next page if there isn't room for it on the current page).

If you don't specify a destination for the table, the document style determines the placement. Both the `report` and `article` document styles use a standard specification of "`tbp`" for tables. This is usually sufficient, but can cause a problem when a table appears at the beginning of a chapter or a document, since you don't want it to appear before the chapter heading at the top of the page. Table 6.1 was explicitly specified as a "`[b]`" table for this reason.

Tables are automatically numbered (throughout the document in an `article`, and within each chapter in a `report`). To place a caption with the table number in it, you simply use the `\caption` control word. For example, the caption on Table 6.1 was specified with

```
\caption{Figure/Table Destinations}
```

LaTeX also provides a means for producing a List of Tables (and List of Figures) at the beginning of the document, similar to the Table of Contents facility. For example, you can place the control word `\listoftables` in your document after the `\tableofcontents` command. `\listoffigures` works similarly.

Sequence	*Description*
`\>`	Tab to next tab stop.
`\<`	Tab backwards to previous tab stop.
`\'`	Right-justify current entry against previous tab stop.
``\` ``	Push following text to right margin.
`\+`	Set left margin right one tab stop.
`\-`	Set left margin left one tab stop.
`\=`	Set a tab stop at current position on a line
`\kill`	Start a new line without outputting the current line (but keep any tab stops that were set).
`\pushtabs`	Save tab settings.
`\poptabs`	Restore saved tab settings.

Table 6.2: `tabbing` Control Sequences

6.1.2 The `figure` Environment

The `figure` environment is almost identical to the `table` environment. In fact, the *only* difference is that figures are numbered separately from tables, and that the `\caption` control word produces the word "Figure" instead of "Table" in the caption.

Now let's take a look at what sort of things you can put in a Figure or Table.

6.2 Tabbed Lines

The simplest sort of table is the kind that typists are quite familiar with, in which the typewriter is set up with "tab stops" at various positions in the line, and a "tab key" is struck to position the carriage at the next tab stop. Of course, it's a pretty rare terminal (or typewriter, for that matter) that has a carriage, and LaTeX presents a kind of simulation of the tab-key mechanism in the `tabbing` environment.

Within the `tabbing` environment, certain control sequences are given special functions to facilitate setting up simple tables. Table 6.2 lists some of these, but let's look at an example first. Suppose we want to produce the following table:

Year	*Simple Interest*	*Annual Comp.*	*Daily Comp.*	*Continuous*
0	5000.00	5000.00	5000.00	5000.00
1	5400.00	5400.00	5416.37	5416.44
2	5800.00	5832.00	5867.42	5867.55
3	6200.00	6298.56	6356.02	6356.26

The input for this table was:

```
\begin{tabbing}
\sl Year \= \sl Simple Interest \=
\sl Annual Comp. \= \sl Daily Comp. \= \sl Continuous\\
0 \> 5000.00 \> 5000.00 \> 5000.00 \> 5000.00\\
1 \> 5400.00 \> 5400.00 \> 5416.37 \> 5416.44\\
2 \> 5800.00 \> 5832.00 \> 5867.42 \> 5867.55\\
3 \> 6200.00 \> 6298.56 \> 6356.02 \> 6356.26
\end{tabbing}
```

The `\>` control sequence behaves much like the tab key on a typewriter. The `\'` sequence is similar, but right-justifies the current entry against the *previous* tab stop (with a little extra space) and returns to the beginning of the current column. In other words, `\'` sort of "pushes" text to the left, as if you had a key on the typewriter that caused output to go to the left without moving the carriage.

Since there's no tab stop at the right margin, the separate control symbol `` \` `` pushes all the text that *follows* it up against the right margin in a similar way.

The `\=` control symbol is much like the "tab set" key on a typewriter keyboard, and sets a tab stop at the current line position. Frequently, you will want to set up tab stops by typing the widest entries that will appear in each column (and hitting the "tab set" control symbol `\=` after each one), but won't want to actually output a line with all those entries at once (and certainly not as the first thing in the table). In this case, don't end the line with the usual `\\`, but use `\kill` instead; this retains the tab set-ups, but throws away the current line without outputting anything.

Finally, `\pushtabs` and `\poptabs` are sort of like LaTeX environments: `\pushtabs` saves all the current tab stops; `\poptabs` restores the last set of stops that were thus saved.

There are some more oddities concerning tab stops. Consider the following input:

```
\begin{tabbing}
   Short \= A bit Longer \= Negligible\\
   A longer entry \> \> But not too long\\
   Now something ugly \> But consistent
\end{tabbing}
```

The result of this is:

```
Short A bit Longer Negligible
A longer entry     But not too long
Now something ugly
    But consistent
```

Why does this happen? Because what `\>` does is to go to the next numerical tab stop, i.e. column #1, column #2, etc. This happens even if we've gone past that tab stop, and LaTeX will move backwards, if necessary, to accomplish it. Thus, you don't actually have to know whether or not a long tabbed entry is going to slop over into the next column or not; you can just count tab stops to get you to the right place.

6.2.1 White Space

Sometimes one wants to set explicit sizes for the various fields of a tabbed table. To do this, you need to know how to do two things: tell LaTeX "leave horizontal white space of size x," and how to specify "size x."

Sizes, also known as *dimensions*, must be specified in any of LaTeX's legal units. Some of these units are:

`in`	Inches
`cm`	Centimeters
`pt`	Printer's point. 1in=72.27pt
`em`	A "printer's em," the width of the current typeface. For example, it's twelve points most of the time in this document.
`pc`	Pica (12pt)

To specify a dimension like three inches, you say "`3in`" with no space between the number and the unit. You can use decimal fractions if you like, so "`2.732in`" is a legal dimension, even if your particular printer doesn't have a resolution of one-thousandth of an inch.

Now that you know about dimensions, it's easy to tell LaTeX to skip a little space: just say `\hspace{`*dimen*`}`, where *dimen* is the size of the horizontal space you want to leave. So, to set up an output line with with four one-inch columns, you might simply say:

```
\begin{tabbing}
\hspace{1in}\=\hspace{1in}\=\hspace{1in}\=\kill
One \> Two \> \> Four\\
\> \> Three\\
This \> All \> Lines \> Up\\
Into \> Neat \>\>Columns
\end{tabbing}
```

which produces the following:

```
One       Two                 Four
                    Three
This      All       Lines     Up
Into      Neat                Columns
```

One particularly useful form of white space is `\quad`, which is simply a one-em space. This is a good-looking amount of "padding" space and works in any size type.

Horizontal space can also appear in paragraphs (or anywhere else that a word of text can appear). For example, the appearance of the limerick back in Section 4.3 could be improved by adding `\hspace{2em}` at the beginning of the third and fourth lines.

6.3 Tabular Output

Sometimes you need a table that's even slicker-looking, like the one in Table 6.1. In this case, the `tabular` environment is used. Let's look at the input for Table 6.1:

```
\begin{center}
 \begin{tabular}{|l|l|}
  \hline
  \sl Letter & \sl Meaning\\
  \hline
  h & here \\
  b & bottom \\
  t & top \\
  p & float page\\
  \hline
 \end{tabular}
\end{center}
```

The `center` environment is used in the usual way to center the table. The `tabular` environment requires a second mandatory parameter which describes how each line of the table will look. Here, we've specified that each row will have a vertical line, a **l**eft-justified entry, another line, another left-justified entry, and a final vertical line. Other options include `c` and `r` for **c**entered and **r**ight-justified entries. Also, you can have more than one vertical bar at a time, causing multiple vertical lines to appear in the output to separate the columns.

The `\hline` directive causes a horizontal line to be drawn the width of the table.

Each line in the table must be explicitly ended with `\\` and each entry in the line is followed by the special LaTeX tab character, the ampersand (one of those special characters we talked about back in Section 2.2). Also notice that each tabbed entry occurs in its own environment, so the switch to slanted font in the header line must happen separately for each entry.

You can also set up a "paragraph" entry in a table by specifying a format of `p{`*dimen*`}`, where *dimen* specifies the *width* of the paragraph. For example:

```
\begin{tabular}{|l|p{1.5in}|}
\hline
First Item: & This is a short paragraph\\
\hline
Second Item: & This is a somewhat longer paragraph\\
\hline
\end{tabular}
```

produces the following table:

First Item:	This is a short paragraph
Second Item:	This is a somewhat longer paragraph

6.4 Figures

Of course, anything can go in a `figure` environment (or a `table` environment, for that matter): program text, `tabular` tables, descriptions, quotations ... The LaTeX documentation describes how to produce box-and-arrow-type drawings using the `picture` environment, but that's somewhat outside the scope of this document. However, you should know how to produce some extra vertical space so that a drawing can be pasted in later; simply put the control sequence `\vspace{`*dimen*`}`, where *dimen* is a dimension like "`1.5in`" (for one and a half inches of vertical space). Note that `\vspace` should occur between paragraphs and that odd things will happen if you use it inside a paragraph.

"Because my mother, sir, was always very anxious for me to make a figure in the world, and when she lay a-dying I promised her that I would do so, and then she put a geas upon me to do it."

— JAMES BRANCH CABELL, *Figures of Earth*

Chapter 7

Cross-References

This paper would have been particularly hard to write if it were necessary to know about section numbering in advance, since it refers frequently to things like "Table 6.1 on page 32." LaTeX allows one to assign *labels* to various numbered entities (chapters, sections, tables and figures) and refer to those labels elsewhere in the document. The important thing is that these cross-references are made by name (referring to the label) rather than by number, and obviate the need for the document composer to know about the actual numbers.

As a specific example, Section 2.2 discusses input files. Shortly after the `\section` control word that began that section, I included the line

```
        \label{SIMPLE}
```

This makes it possible to refer to that section elsewhere in the paper by using another control sequence, `\ref`. For example, the first sentence of this paragraph began,

```
        As a specific example, Section~\ref{SIMPLE} discusses
```

Note the use of the tie character (`~`) to prevent line breaking between the word "Section" and the section number.

Labels also keep track of the current page, so I can mention that Section 2.2 is on page 4 by saying

```
        is on page~\pageref{SIMPLE} by saying
```

Cross-references always refer to the smallest division of the document (section, sub-section, sub-paragraph) in which they occur, or to the figure or table in which they are used. **Warning**: if you `\label` a figure or table, you must not do so until **after** the `\caption` control word.

Although overuse of cross-references can detract from the readability of a document, sometimes a large number of cross-references can accumulate in a document, and the writer is responsible for keeping track of the names of all the `\label`s. A block of comment lines can be used to keep the list of labels and their significance on-line within the document. If things get out of control unexpectedly, you can figure out what's what by taking advantage of the fact that all the labels and their values and page references are output by LaTeX into the `.aux` file. Scanning this file for all the lines containing the word "`newlabel`" will produce a list of all the labels you've defined in the document.

Cross-references can refer to labels that occur later in the document, but the document must be run through LaTeX twice. The first time through, LaTeX will complain about undefined labels when the corresponding `\ref` is encountered, and if you print the resulting document, the reference is printed as "??".

"The whole metaphysical thing in 'Kinda' is a reference game, if you happen to get the references in it."

— JOHN TULLOCH and MANUEL ALVARADO,
Doctor Who—The Unfolding Text

Chapter 8

Equation Formatting

Many technical papers contain equations like

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Needless to say, this sort of thing is normally pretty hard to type so that it looks good, even on a typewriter that has all the special symbols.

LaTeX has three environments which constitute "mathematics mode." In mathematics mode, LaTeX's rules and control sequences are changed somewhat so as to facilitate the formatting of equations. These environments are `math` (for formulas that appear in the text, like "$x^2 + y^2 = 0$"), `displaymath` (for formulas that are displayed separately from the text, like the quadratic formula that introduced this section), and `equation` (which is just like display math, except that it adds an equation number at the right margin).

To make all this easier to type, LaTeX lets you type `\(` and `\)` instead of `\begin{math}` and `\end{math}`, and `\[` and `\]` instead of `\begin{displaymath}` and `\end{displaymath}`. Even easier, you can type a dollar sign instead of `\(` and `\)`, so you can type `$x+y=0$` to produce "$x + y = 0$."

The TeXbook contains no fewer than four complete chapters on setting equations in mathematics mode, so it's safe to conclude that it's a pretty large subject. However, this introductory paper isn't concerned with the complete details of mathematics mode, so we'll only include a few outstanding rules.

8.1 Basic Rules of Math Mode

Mathematics mode differs from LaTeX's usual operation in a number of important ways.

1. All spaces are ignored. This includes new lines and tab characters. LaTeX is assumed to know all about spacing equations, and has a series of control characters for inserting spaces "manually", like "`\ `" (that's a control symbol consisting of a backslash followed by a space).
2. Alphabetic characters are in a special font, math italic. Math italic is a lot like text italic, but the character spacing is slightly different. You can switch to roman (or other) fonts via `\rm` and the like, but spaces are still ignored between the words.
3. Many control sequences, like the ones that produce Greek letters (α–ω), only work in mathematics mode and produce errors at other times.
4. You can't start a new paragraph inside a formula, so don't leave a blank line in any math environment. If you do, LaTeX will think that you're trying to leave mathematics mode, and an error will result.

8.2 Subscripts and Superscripts

To get something to print as a subscript, just put it in braces and put an underbar (`_`) in front of it. For example, "x_2" is typed as "`$x_{2}$`". Similarly, superscripts are represented with the circumflex (`^`, also known as a caret or "up-arrow").

You can do both at once:

```
$x^{1}_{2}$
```

produces "x^{1}_{2}". Furthermore, the superscript can be the first thing that happens in math mode. The word "su$^{\rm per}_{\rm b}$script" was typed as

```
su$^{\rm per}_{\rm b}$script
```

This example also demonstrates that subscripts and superscripts can be something other than numerals, and emphasizes that alphabetics normally appear in math italic type (requiring `\rm` to get normal roman letters).

8.3 Mathematical Symbols

LaTeX has a dazzling array of mathematical symbols, all of which are only available in math mode. To produce Greek letters like "ω" (omega) you type `$\omega$`. The capital omega (Ω) is pronounced `$\Omega$`. You can also get boldface versions of these characters. Some of the other mathematical glyphs that are recognizable by non-mathematicians are `\times` (seen in "$5 \times 5 = 31_{\rm eight}$"), `\infty` ($\infty$) and the popular `\bullet` ($\bullet$). A complete list of symbols can be found in Appendix F of *The TeXbook* on pages 434–439. A quick perusal of the complete list will show that many characters are named according to their mathematical significance, so that mathematical input is "pronounced" in a manner similar to the terminology of mathematics.

8.4 Logs and Sins

Some mathematical words like "log" and "sin" are conventionally printed in roman letters, not italic. LaTeX provides control words like `\log` that automatically produce the corresponding word in roman letters in math mode.

Some of these (notably `\lim`) behave interestingly with respect to subscripts. For example, typing

```
        \[ \lim_{n \to \infty} x_{n} = 0 \]
```

produces

$$\lim_{n \to \infty} x_{n} = 0$$

with the subscript appearing *below* the word, which is the usual mathematical appearance of "lim".

8.5 Fractions

While you can say things like `$x/y$` most of the time to produce "x/y", sometimes you need to do things like "$\frac{1}{2}$". For this, the control sequence `\frac` is available. It takes two parameters: the numerator and the denominator. For example, "$\frac{1}{2}$" can be typed as "`$\frac{1}{2}$`" and the quadratic formula which began this chapter can be typed as

```
\[ x = \frac{-b \pm \sqrt{b^{2}-4ac} }
           {2a} \]
```

Since it's possible to produce fairly ugly output using `\frac`, it is recommended that you consider carefully whether or not a simple slash will do.

This example also demonstrated the `\sqrt` control word. You can also get other roots by supplying an optional parameter. For example `$\sqrt[3]{x^{3}}$` produces $\sqrt[3]{x^3}$.

8.6 Some Random Examples

This section will close with a short chrestomathy that might enable you figure out how to format similar equations by analogy. Mathematics mode is a big topic, and you should read one of the reference works for complete details.

```
\[ \sum_{n=1}^{m} n = 5 \]
```

produces:

$$\sum_{n=1}^{m} n = 5$$

```
\begin{equation}
  \int_{-1}^{-1}x\,dx
\end{equation}
```

produces:

$$\int_{-1}^{-1} x\,dx \tag{8.1}$$

The `\,` control sequence produces a "thin space." LaTeX actually has four different math-mode spacing control characters:

Sequence	*Name*	*Example*
`\;`	Thick space	$x\;x$
`\:`	Medium space	$x\:x$
`\,`	Thin space	$x\,x$
	(no space)	xx
`\!`	Negative space	$x\!x$

Negative space is rarely used.

```
\[ \hat{a} \equiv \tilde{n} + \bar{\jmath} \]
```

produces:

$$\hat{a} \equiv \tilde{n} + \bar{\jmath}$$

There are ten different accent marks that can be placed above mathematical symbols, and the control words used for them are *different* from the control symbols used for accenting normal text characters. See page 135 of *The TEXbook* for a complete list.

```
\[ e^{i\theta} = \cos\theta + i\sin\theta \]
```

produces:

$$e^{i\theta} = \cos\theta + i\sin\theta$$

```
\[ x_{\rm max} - x_{\rm min} \]
```

produces:

$$x_{\rm max} - x_{\rm min}$$

```
\[ \sqrt{1+\sqrt{1+\sqrt{1+x}}} \]
```

produces:

$$\sqrt{1+\sqrt{1+\sqrt{1+x}}}$$

```
$ \langle x,y \rangle$
```

produces “$\langle x,y \rangle$”.

```
\[ \left( \frac {x+y}{z+2} \right) \]
```

produces:

$$\left(\frac{x+y}{z+2}\right)$$

Various kinds of brackets, parentheses, and the like can be used with `\left` and `\right`. These produce brackets big enough to surround whatever is between them. However, for every `\left`, there must be a `\right`, although they need not be the same type of bracket.

```
        \[ \left\langle
                   \frac {2(x+y)}
                         {|z|}
            \right\rangle
        \]
```

produces:

$$\left\langle\frac{2(x+y)}{|z|}\right\rangle$$

```
        \[ \left |
          \begin{array}{ccc}
             42 & 39 & 11 \\
             23 & 5  & 9  \\
              7 & 17 & 27
           \end{array}
           \right | = -18602
        \]
```

produces:

$$\left|\begin{array}{ccc} 42 & 39 & 11 \\ 23 & 5 & 9 \\ 7 & 17 & 27 \end{array}\right| = -18602$$

The `array` environment is a math-mode version of the `tabular` environment.

```
        \[ \delta_{ij} = \left\{
          \begin{array}{ll}
            y & {\rm if\ } y>0\\
            z+y & {\rm otherwise}
           \end{array}
        \right. \]
```

produces:

$$\delta_{ij} = \left\{ \begin{array}{ll} y & \mbox{if } y > 0 \\ z + y & \mbox{otherwise} \end{array} \right.$$

The `\right.` closes the `\left\{` without printing anything. Since spaces are ignored in mathematics mode, the control character `\ ` (i.e., a backslash followed by a space) is necessary to insert a space after “if”. You aren’t necessarily expected to understand all this.

The confusion and complexity of the Fourth Branch, Math,
are extreme.

— GWYN JONES and THOMAS JONES, *The Mabinogion*

Chapter 9

LaTeX Errors

In which we discover what happens when things go wrnog.

LaTeX error messages (and, for that matter, its purely informative messages) tend to be verbose, and can overwhelm the victim with spurious information. This is largely due to the way that LaTeX is layered on top of another system, TeX, which has its own error messages. When LaTeX encounters an error, what often "really" happens is that TeX perceives this as an error in the LaTeX software itself, and displays a lot of information relating to the internal structure of that software. Most of the time, you can ignore the large majority of what gets typed out when there's an error.

9.1 Errors from TeX

But, of course, some of it is useful. Let's consider the following simple erroneous input file:

```
\documentstyle{report}
\begin{document}
This document has an \invalid control word.
\end{document}
```

After the usual preliminary stuff, LaTeX will display the following message:

```
! Undefined control sequence.
l.3 This document has an \invalid
                                  control word.
?
```

and leave your cursor after the question mark. The error message in this case is straightforward. The notation "`l.3`" is very important, since it tells you which line of the file (line 3) the error is on. The message also shows you how much of the input line has been read by the time the error was noticed. In this case, the control word `\invalid` was the last thing that was read, and this caused the error.

If you type a question mark, you get a "menu" of options:

```
Type <return> to proceed, S to scroll future error messages,
R to run without stopping, Q to run quietly,
I to insert something, E to edit your file,
1 or ... or 9 to ignore the next 1 to 9 tokens of input,
H for help, X to quit.
?
```

If you type "`H`" you get an explanation of the error message. Sometimes it's helpful, not at other times. In this case, the message is:

```
The control sequence at the end of the top line
of your error message was never \def'ed. If you have
misspelled it (e.g., `\hobx'), type `I' and the correct
spelling (e.g., `I\hbox'). Otherwise just continue,
and I'll forget about whatever was undefined.

?
```

In other words, you can insert a substitute control word using the "`I`" option, or just hit a Return and ignore the bogus control sequence. This sort of error is very common, and fortunately easy to correct.

9.2 Errors from LaTeX

Now let's look at a somewhat more complex error. Consider the following input file:

```
\documentstyle{report}
\begin{document}
\begin{quoth}
The raven said, ``Nevermore.''
\end{quote}
\end{document}
```

In this case, LaTeX produces the following verbose display:

```
LaTeX error.  See LaTeX manual for explanation.
              Type  H <return>  for immediate help.
! Environment quoth undefined.
\@latexerr ...for immediate help.}\errmessage {#1}
                                                  \endgroup
l.3 \begin{quoth}

?
```

Here, the error originates from LaTeX and not TeX. The error message is still comprehensible: there's no such thing as a `quoth` environment. However, the display with `@latexerr` and all that stuff is not very helpful and should be ignored.

On the other hand, the input line that triggered the error (once again, line 3) is displayed. If we type "`H`" in this case, we see:

```
Your command was ignored.
Type  I <command> <return>  to replace it with another command,
or  <return>  to continue without it.
?
```

In this case, we might type

```
    I \begin{quote}
```

(followed by a carriage return) and LaTeX would continue to process the input file. A good rule of thumb is: if you think you can fix the error in place using "`I`", go ahead and try it. If not, just hit Return and try to get as much of the document printed as you can. Usually, if you can see what the output looks like, you can figure out what's really gone wrong. For example, if the last five pages of your output are all in italics, it's likely that you forgot a closing right-brace character somewhere; if all the italics are run together, you probably forgot a close-math-mode bracket.

9.3 Severe Errors

Sooner or later, you're going to hand LaTeX a file that it can't understand at all, because it's missing either the `\documentstyle` control word or `\begin{document}` command.

9.3.1 No Document Environment

If you try to run LaTeX on a file containing the single line,

```
This file has no document header or anything else in it.
```

you will get the following error message from LaTeX:

```
LaTeX error.  See LaTeX manual for explanation.
              Type  H <return>  for immediate help.
! Missing \begin{document}.
\@latexerr ...for immediate help.}\errmessage {#1}
                                                  \endgroup
<to be read again>
                   T
l.1 T
     his file has no document header or anything else in it.
?
```

In this case, LaTeX decided something was wrong as soon as it read the very first character of the file. At this point, discretion is the better part of valor, and you should '`X`' out of LaTeX. If you do continue, LaTeX will try to read additional input from the terminal once it finishes reading the file, and your only escape will be to hit your interrupt key.

9.3.2 No Document Style

Even if you include the `\begin{document}` and `\end{document}` commands, you'll still be in trouble if you don't specify a document style, since the document style files include critically important definitions used by LaTeX. Here's what the output from such a file might look like:

```
! Undefined control sequence.
\@floatplacement ...global \@toproom \topfraction
                                                  \@colht \global \@botnum \...

\document ... \hsize \begingroup \@floatplacement
                                                  \@dblfloatplacement \makea...

\begin ...{#1}\ifx \@currenvir \@tempa \@document
                                                  \else \@nodocument \fi
l.1 \begin{document}

?
```

This is completely mysterious to the untrained eye, since it refers to a particular internal control variable that LaTeX is trying to use to set up the `document` environment. Alas, this variable hasn't been defined by a document style description file. Complete chaos. Once again, it's best to '`X`' out of a situation like this. Attempts to continue will just produce additional cryptic error reports.

9.3.3 Invalid Document Style

Suppose you start your file with the line,

```
    \documentstyle{nosuchthing}
```

In this case, you'll get a message from TeX explaining that it can't find the file you asked for. TeX then asks for the name of a different file to read. For example,

```
! I can't find file 'nosuchthing.sty'.
<to be read again>
                   \relax
\@documentstyle ...ionfiles {}\input #2.sty\relax
                                                  \let \@elt \input \@option...
l.1 \documentstyle{nosuchthing}

Please type another input file name: report.sty
(/usr/lib/tex/macros/report.sty
Document Style 'report'. Version 0.91 - released 25 June 1984
...
```

In this example, you can recover by noticing that LaTeX was looking for the file named '`nosuchthing.sty`.' Logically, then, you can make it use the '`report`' document style by typing the file name '`report.sty`', as we did in this case (in response to "`Please type another input file name:`"). Sure enough, things proceed from there in a normal way.

9.3.4 Accidental Mathematics Mode

The special characters `$`, `^`, and `_`, all perform operations associated with mathematics mode. If they are used in a non-math context, LaTeX can become somewhat confused. For example, consider the input,

```
This line is over $5.00 in price and also refers
to the variable foo_bar which will cause a problem.
```

When LaTeX reads the first dollar sign, it will switch to mathematics mode. When it comes to the end of the paragraph, or of the document, it will conclude that a closing dollar sign was somehow left out, and print an error message like:

```
! Missing $ inserted.
<inserted text>
                $
<to be read again>
                   \par
```

followed by an assortment of miscellaneous material. If you continue, it will try to set the "equation". Since the spaces don't count, it will try to set

$$inpriceandalsoreferstothevariablefoo$$

as a single variable. Quite a mess. Suppose, now that we 'correct' the file by putting a back-slash in front of the dollar sign. Everything goes fine until the underscore is encountered, at which point LaTeX thinks that math mode is intended, and says,

```
! Missing $ inserted.
<inserted text>
                $
<to be read again>
                    _
l.4 to the variable foo_
                        bar which will cause a problem.
? h
I've inserted a begin-math/end-math symbol since I think
you left one out. Proceed, with fingers crossed.
```

In other words, LaTeX automatically switched to math mode. Now things are just like they were in the previous example, and when the end of the paragraph is reached, LaTeX will insert a closing dollar-sign to match the one it just inserted. If you really want to refer to the variable `foo_bar`, the `\verb` control word should be used.

9.4 Overfull Hboxes

It is impossible to completely cover the techniques of LaTeX error recovery in an introductory document, but there are a couple of very common errors that sound terrible yet are in reality fairly benign.

The first of these is "`Overfull \hbox (`*xx*`pt too wide)`" This is followed by a display of an output line that shows the font and the possible hyphenation points. This is just a warning, and can be fairly harmless. What it means is simply that LaTeX couldn't find a good place to break a line without producing an unacceptable amount of space between the words. So it gave up, and set the line with some amount of text sticking into the right margin. The exact amount of text sticking into the margin is displayed in printer's points (72.27 of them to the inch, if you recall).

Since this is a warning, you don't get a chance to do anything about it while LaTeX is running. If the amount of right-margin slop is unacceptable, you can either change the prose in the paragraph in which the line appears, or place the entire paragraph (including the blank line that ends the paragraph) inside an environment called `sloppypar`; this will allow LaTeX to set lines with somewhat unaesthetic amounts of space between words. In any case, you'll have to run the document again to get it right.

If you see that LaTeX has failed to notice a good hyphenation spot near the end of the line, you can salvage the situation by telling LaTeX about how to hyphenate that word. For example, LaTeX doesn't know how to hyphenate the word "hello". You can add the following command before the `\begin{document}`:

```
\hyphenation{hel-lo}
```

which will tell LaTeX how to do it. This may enable it to reset the offending line in an acceptable way on the next run. You can also insert "discretionary hyphens" in a word to indicate a possible hyphenation point on a one-time-only basis. For example,

```
The rec\-ord was re\-cord\-ed yesterday.
```

could be hyphenated at any point indicated by the control symbol `\-`, or between the syllables of "yesterday" (which LaTeX knows how to hyphenate).

9.5 Underfull Vboxes

Almost as often, the message, "`Underfull \vbox (badness ...`" appears. This just means that LaTeX can't find a really good place to break the current page, so it's "stretching" the page more than it would like (producing extra white space in the page). Once again, the results may or may not be acceptable; you can always insert `\newpage` to force a page break.

In summary, while LaTeX errors may sometimes be cryptic or even frightening, they can usually be deciphered by the patient user. Since the line number of the input which caused the error is printed, one can always determine *where* the error is, even if it isn't immediately obvious *what* the error might be. There's no substitute for experience, though, so don't be intimidated by error or warning messages. Producing high-quality documents with LaTeX is, errors notwithstanding, a highly rewarding experience.

There is no prompting or chatter, and error messages are curt and sometimes unhelpful.

— BRIAN KERNIGHAN and ROB PIKE, *The UNIX Programming Environment*