WORD PROCESSING USING LaTeX

Richard M. Felder
Department of Chemical Engineering
North Carolina State University
Raleigh, NC 27695

November 29, 1999

Contents

WORD PROCESSING USING LaTeX

Richard M. Felder
Department of Chemical Engineering
North Carolina State University
Raleigh, NC 27695

November 29, 1999

LaTeX (pronounced Latéck) is a program that enables you to create a photo-ready manuscript. It arranges text, sets up equations and tables, prints in boldface or italics, changes type size, and in general does everything a typesetter can do to arrange text in any desired format and appearance.

This document describes how to create manuscripts using LaTeX. The presumption is that you already know how to create and edit files on your computer. If you don't, find out before you attempt to learn LaTeX. For additional information, consult the LaTeX manual.[1]

1 Creating and Printing a Document

LaTeX files must be given the type .tex. Suppose you wish to create and print a document file which you give the name NAME.tex. (Any combination of letters and numbers may be substituted for NAME.) The command sequence is as follows. (The percent signs are the computer prompts.)

```
% emacs NAME.tex
```

(Create the text file according to the instructions given in the remainder of this document)

```
% latex NAME
```

This command causes two files to be created. NAME.dvi is an intermediate file which will be processed in the next step, and NAME.log contains a transcript of all the messages generated by the program, including error messages. To look at those messages again, type

`% more NAME.log`.

To get interpretations of some of the more bewildering error messages, see Section 16 of this document.

[1] Lamport, L., **LaTeX: A Document Preparation System**, Reading, Mass, Addison-Wesley Publishing Co., 1986.

You can print the file NAME.dvi with the command.

```
% dvipr NAME
```

This causes the printer to create a postcript file, which is the one that is actually printed. You never see that file. However, if you need to create a postcript file, you can do it with the following command.

```
% dvipr -N NAME
```

This converts NAME.dvi into a printable file called NAME.ps. If you are going to want to print only specific pages—say, pages $n1$ through $n2$, type instead

% `dvipr NAME -p` *n1* `-l` *n2*

You would of course substitute the desired numbers for *n1* and *n2*. *Be sure to leave spaces around* `-p` *and* `-l`.

If you would like to see the output of your latex file, you should give the command

```
% xdvi NAME
```

There are many features of the viewer that you may find useful. For example you can make dissapear the column to your right, just pressing the letter *x*. You can make it re-appear by pressing *x* again. Try it now!. On the other hand, if you want to change the size of the text in the screen you have to use the letter *s* to shrink it to other sizes. For example *2 s* should give you a bigger font.(Don't try it yet!) Now if you want to center text (and here is the tricky part) try putting the cursor in the center of the screen an press the letter *c*. Try putting the cursor in different positions in the screen and press the letter *c* in order to get a feeling of how the command will work. Now you can try changing the shrink factor and centering the text again. Notice that you can try *12 s* for a small version of your file. Notice that no record appears on the screen that you actually pressed those keys. The program knows that you did it, but it does not tell you that you did it. In any case you should not stop typing because you don't see what you type. Notice also that when you write a command like *2 s*, you should not leave spaces in between 2 and *s*, just write 2 and inmediatelly after the letter *s*. If you press the space bar at any time it will change the page that you are viewing. Centering works fine in one page, but when you go to the next page (press *n*) or the previous page (press *p*) the centered text is lost. This is because you did not press the letter *k*. If you press the letter *k*, when you go to the next/previous page, it will keep the centering as you had it before. You only need to press *k* only once during your session, not every time you change page (in general you need to press it an odd amount of times, since pressing *k* twice undoes one pressing of *k*). *u* and *d* will move the text up and down the screen, as *l* and *r* will move left and right respectively.

Now lets say you want to advance 8 pages, then simply press *8 f*, the command to go back would be *8 b*. If you want to go to an specific page number, say page 21, you have to write *21*

g. Notice that this is the 21st page of the document (including cover, index, introduction,etc..). If you want to go to the last page just press the letter g.

Just knowing these simple commands will save you time in looking through your dvi file. Maybe this is a good file for you to try them on. By the way, did I mention that q is for quit ?

2 Biting off small chunks: How to avoid frustration in one easy lesson

The good news is that LaTeX is an incredibly versatile program, and will enable you to do almost anything you might conceivably want to do in producing a document. The bad news is that this versatility comes at the expense of complexity: there are lots of commands to learn and many opportunities to make subtle errors. Moreover, LaTeX can be obscure or downright incomprehensible when it comes to diagnosing errors when it finds them.

When you are working on a large document, it takes LaTeX some time to execute the steps outlined in the previous section (LATEX NAME, DVI2PS NAME, etc.) You will frequently run into a situation in which you have trouble getting a part of your document to come out right, or you continue to get error messages that you can't figure out. If you keep trying to correct the full document file and go through the entire lengthy compilation and printing procedure each time, you'll get extremely irritated, and will end up taking much more time than is necessary or giving up and going back to a simpler system.

There's a better way. Prepare your document a little bit at a time in a separate file which contains the necessary setup commands (the *preamble*), and work on that file until you get it right. (It usually won't take that long when you break it up like this.) Then edit the complete document file, **include** the smaller file containing the correct statements, and delete the extra setup commands. Then go back to the working file, delete the part you just did, and proceed to the next section. If you're not sure how to move text back and forth between files using your editing program, consult someone who knows.

As far as those incomprehensible diagnostics go, we say something about them in Section 16 of this document, and there's more on pp. 118-130 of the LaTeX manual.

3 Forbidden characters in LaTeX–a warning

Before we tell you how to write a LaTeX file, we must warn you about a common mistake and how to avoid it. There are ten characters that have special meanings in LaTeX, and which you therefore can't use in a normal way in a text file.

Special characters: `#  $  %  &  _  {  }  ~  ^  \`

We will eventually tell you what each of these characters means to LaTeX. In the meanwhile, you will certainly encounter the need to print some of them in your text. Here's how you do it. (Note: \ is the **backslash**, and is adjacent to the return key on the VT240 keyboard.)

To print # , type `\#`
To print $, type `\$`
To print % , type `\%`
To print & , type `\&`
To print _ , type `\_` (underline character)
To print { or } , type `\{` or `\}`
To print ˜ , type `\verb'~'`
To print ˆ , type `\verb'^'`
To print \ , type `\verb'\'`

4 Setting up a text file—the Preamble

A LaTeX text file (NAME.TEX) consists of the text you actually want to print interspersed with commands that tell LaTeX how you want the document to appear. A LaTeX command is a line beginning with a backslash character, \. *The command must be in all lower case letters.* For example,

`\pagestyle{empty}` = omit numbering of pages

`\oddsidemargin 0.125in` = locate the left margin of the text 1 + 0.125 inches from the left side of the paper.

`\begin{center}` = center all the text that follows this command up to the point where another command, \end{center}, is encountered.

`\vspace{5 mm}` = skip 5 mm (vertical space) before printing the next line.

A text file contains two principal sections: the **preamble**, which contains setup instructions about how the text should appear, and the document itself, which contains the text to be typed and other commands. The format of the file is as follows:

`\documentstyle[`...
(Preamble–LaTeX commands)
`\begin{document}`
(Your text and additional LaTeX commands)
`\end{document}`

In the remainder of this section we discuss what goes in the preamble, and the rest of the guide outlines the different commands you can use in the document to make your text appear in the desired form.

4.1 A typical preamble, and how to avoid having to type it for each new document

Following is the preamble used to create the document you are reading.

```
\documentstyle[12pt,double]{article}
\oddsidemargin 0in
\textwidth 6.75in
\topmargin 0in
\textheight 8.5in
\parindent 0em
\parskip 2ex
\begin{document}
```

The text file (NAME.TEX) would begin with these commands or alternative versions of them. (Strictly speaking, the last command is not part of the preamble but begins the actual document portion of the file.)

You may be thinking what a pain in the neck it will be to type all this stuff every time you create a new text file. Fortunately, you don't have to. If you put the commands in a separate file—say, one called ARTSETUP.TEX—you may then proceed as follows:

`$ EDIT NAME.TEX` (Enter the editor)
`(GOLD)(7)` (Go into command mode)
`INCL ARTSETUP.TEX (ENTER)` (Bring ARTSETUP.TEX into this file)

The preamble commands will then be in place in your file, and you can proceed from there. You may wish to create several such preamble setup files—e.g., one for technical manuscripts, one for letters, one for memoranda, etc.—and include the appropriate one in each document you type.

Let us now look at what those commands mean, and at other instructions you might want to include in your preamble.

4.2 Document style

```
\documentstyle[12pt,double]{article}
```

The `\documentstyle` command is always the first command in the file. It is used to specify which one of several predefined styles LaTeX should use to set up your document.

The terms in the square brackets are *options* associated with the command. *12pt* specifies that 12-point type is to be used. (12-point type is what you are looking at now.) If you didn't include this option, LaTeX would default to 10 point type, and you would need a magnifying glass to read it. *double* enables you to double-space the document. Even if you're not going to double-space, it doesn't hurt to include this option. Other possible options include *twoside*, which sets up the document to be printed on both sides of the page, and *twocolumn*, which produces a document in two columns.

The term in curly brackets is the *argument* of the command. It specifies the predefined type style you want LaTeX to use. The choices are as follows:

Article: Convenient to use for most documents you are likely to produce. This guide is done in article style.

Report: Suggested for longer documents. You get one more section heading level (*Chapter*) than you do in the *article* style, and various spacing parameters are slightly different. Figures, tables, and equations are numbered within chapters (1.1, 1.2, 1.3,...) where in the *article* style they are numbered consecutively throughout the paper. Abstracts are placed on a separate page by the **abstract** environment, while they are generally located on the title page in the *article* style.

Book: Like *Report* in most respects. There is no *abstract* provision in this style.

Letter: Useful for letters. (It has easy ways to put in addresses, signatures, etc.)

Slides: Good for slides or transparencies. This style must be used in conjunction with a version of LaTeX called SliTeX, which we don't yet have on the VAX.

4.3 Setting up horizontal and vertical margins

```
\oddsidemargin 0in
\textwidth 6.75in
\topmargin 0
\headheight 0in
\textheight 8.5in
```

The `\oddsidemargin` command tells LaTeX to put the left margin 1+0 inches from the left side of the page. (You always add the given number to 1 inch to determine where the left margin will be.) The `\textwidth` command makes the width of the printed text 6.75 inches. In this case, assuming you are using standard 8.5-inch width paper, the right margin would be 8.5-6.75-1.0=0.75 inch wide.

The `\topmargin` command locates the top of the head section 1+0 inches from the top of the page. The `\headheight` command allows 0 inches for the height of the head. (The head is a section at the top of the page normally used for page numbers and/or running labels. We are not using a head in this document). The `\textheight` command specifies that the body of the text (excluding the head and the foot) occupies a vertical space of 8.5 inches.

You cannot use these commands to change margin settings within the document (i.e., after the `\begin{document}` command). If you wish to print a section of text with different margins, you must either enclose it in a box of some sort or within an environment like `tabbing` or `quote`.

Other commands given on p. 163 of the LaTeX reference manual have to do with spacing of marginal notes, the vertical separations between the head and the body and between the body and the foot, the height of the foot, and the width of the margin on even-numbered pages when you are printing on both sides.

4.4 Setting up paragraph indentation and line skipping

```
\parindent 3em
\parskip 2ex
```

The first command instructs LaTeX to indent 3 spaces—each space being the width of the letter m—at the beginning of each paragraph. If you wanted to omit indentation—that is, to use block style, as this document does—you would change 3em to 0em. The second command instructs LaTeX to skip a line between paragraphs; more precisely, to leave a vertical space the height of 2 letter x's stacked vertically. If you are double–spacing a document, you might wish to change from 2ex to 0ex.

Unlike the margin–setting parameters, these parameters may be changed at will within a document. *Each such command should be preceded by a blank line.*

4.5 Page style

`\pagestyle{`*style*`}`

The `\pagestyle` declaration specifies what goes into the head and foot, including page numbers if they are desired. Instead of *style* in the given statement, you would type one of the following arguments:

plain The page number is in the foot and the head is empty. This is the default page style for the `article` and `report` document styles, so you can omit the `\pagestyle` declaration if you want to number pages at the bottom.

empty No page numbering—the head and foot are both empty.

headings The page number and other information, determined by the document style, are put in the head; the foot is empty.

myheadings Similar to the `headings` page style, except you specify the "other information" that goes in the head, using the `\markboth` and `\markright` commands described on p. 83 of the LaTeX reference manual.

5 Page numbering

5.1 Setting and resetting page numbers and numbering style

You can reset a page number to anything you want using the `\setcounter{page}` command, and you can reset the current page number to 1 and specify the type of page numbers you want (arabic numbers, lower or upper case roman numerals, lower or upper case letters) with the `\pagenumber` command. The syntaxes of these commands are as follows:

`\setcounter{page}{25}` causes the number of the current page to be set to 25. (This is useful if you are breaking up a long manuscript into several text files and want to keep the page numbers sequential.)

`\pagenumbering{style}` resets the current page to 1, and begins numbering in the style you specify as the argument of the command. In place of "style," you would type one of the following words:

arabic (to get page numbers 1,2,3...) This is the default style.
roman (i,ii,iii,iv,...)
Roman (I,II,III,IV,...)
alph (a,b,c,...)
Alph (A,B,C,...)

For example, you might want to print a cover page without a page number, then use lower case roman numerals for the first few pages of the document (abstract, table of contents, etc.), and then switch to arabic numerals for the actual text. The following sequence of commands would accomplish this (the command `\newpage` instructs the computer to skip to a new page and recommence printing):

`\pagestyle{empty}`
(Text of cover page)
`\pagestyle{plain}`
`\setcounter{page}{1}`
`\pagenumbering{roman}` (Text of pages i, ii, iii,...)
`\setcounter{page}{1}`
`\pagenumbering{arabic}`
(Text of pages 1, 2, 3...)

5.2 Cross-referencing page numbers

If you insert the command

`\label{`‘*key*’`}`

anywhere in the text, you may thereafter include the command

`~\pageref{`‘*key*’`}`

anywhere in the text, and LaTeX will automatically substitute the number of the page corresponding to the location of the **\label** command occurs. Any string of letters, numbers, and punctuation marks other than a comma may be substituted for *key*. For instance, the command

`\label{‘crefpage’}`

was typed immediately after the `\subsection{Cross-referencing...}` command that began the section we are in. Consequently, typing

`(See p. ~\ref{‘crefpage’}.)`

anywhere in the text produces

(See p. 5.2.)

Note: It takes two passes through the document file to get the cross-referencing feature to work. The first time you translate the file (`$ LATEX NAME`) with the cross-referencing commands included, LaTeX assigns numbers to keys (*crefpage*, in this case) and inserts question marks where the citations (**~\ref{}**) occur. The second time through, the page numbers are substituted from the previous run. Similar provisions can be made for cross-referencing sections, equation numbers, figures, and tables.

6 Sectioning

6.1 Section headings

LaTeX provides several section heading levels. In descending order, they are

`\part{ }` (Optional. Usually used only for books.)
`\chapter{ }` (Book and report documentstyles only)
`\section{ } \subsection{ } \subsubsection{ } \paragraph{ } \subparagraph{ }`

The desired headings are inserted between the braces. Leave blank lines before and after each such command.

For example, you are currently in Subsection 6.1—Section headings, of Section 6—Sectioning. The headings were produced with the commands

```
\section{Sectioning}
\subsection{Section headings}
```

Note several points about these commands.

- LaTeX automatically advances counters to make sure all sections, subsections, subsubsections, etc., are numbered sequentially.
- All section headings may be stored for subsequent inclusion in a table of contents. (We will discuss this point further in the next section).
- If you don't want sections numbered, add asterisks to the commands before the first brack (e.g. `section*{ }`, `subsection*{ }`, etc.) Note, however: Section headings generated in this manner will not be included in a LaTeX-generated table of contents.
- You can generate running heads, modify the table of contents, and do a variety of other things with section headings. For more information, see pp. 157-158 of the LaTeX manual.

6.2 Cross-referencing section numbers

If you insert the command

`\label{`*key*`}`

right after a section heading command, you may thereafter include the command

`~\ref{`*key*`}`

anywhere in the text, and LaTeX will automatically substitute the number of the section heading. Any string of letters, numbers, and punctuation marks other than a comma may be substituted for *key*. For instance, the command

```
\label{crefsect}
```

was typed immediately after the `\subsection{Cross-referencing...}` command that began the section we are in. Consequently, typing

```
(See Section ~\ref{crefsect}.)
```

anywhere in the text produces

(See Section 6.2.)

Note: As is the case for cross-referencing page numbers, it takes two passes through the document file to get the section cross-referencing feature to work.

6.3 Table of contents

You can use the **\tableofcontents** command to generate a table of contents, which includes all of the numbered section headings in your document. The command should be placed wherever in the document you want to print the table (as long as it comes after `\begin{document}`).

Suppose the name of your document file is **NAME.TEX**. You will have to translate the final version of the document (`$ LATEX NAME`) at least twice to get the table of contents. The first time you do it, LaTeX generates a file called **NAME.TOC** with all the headings in it. The second time, LaTeX reads this file and produces the table of contents from it. The result looks like the table of contents at the beginning of this document.

7 Spacing, filling, justifying, and centering text

7.1 Double spacing

One of the interesting features about the LaTeX manual is that it does not tell you how to double–space documents. One way to do it is to create the following file, calling it **DOUBLE.STY**, and keep it in your directory so that LaTeX will have access to it.

```
\let\single\@normalsize
\ifcase \@ptsize\relax   % 10pt option
\def\double{\@setsize\normalsize{24pt}\xpt\@xpt
\abovedisplayskip 10pt plus2pt minus5pt%
\belowdisplayskip \abovedisplayskip
\abovedisplayshortskip  \z@ plus3pt%
\belowdisplayshortskip  6pt plus3pt minus3pt}
\or  % 11pt option
\def\double{\@setsize\normalsize{27.2pt}\xipt\@xipt
\abovedisplayskip 11pt plus3pt minus6pt%
\belowdisplayskip \abovedisplayskip
\abovedisplayshortskip  \z@ plus3pt%
\belowdisplayshortskip  6.5pt plus3.5pt minus3pt}
\or  % 12pt option
\def\double{\@setsize\normalsize{30pt}\xiipt\@xiipt
\abovedisplayskip 12pt plus3pt minus7pt%
\belowdisplayskip \abovedisplayskip
\abovedisplayshortskip  \z@ plus3pt%
\belowdisplayshortskip  6.5pt plus3.5pt minus3pt}
\fi
```

Then, the following commands may be included anywhere in the text file, *preceded by blank lines.*

`\double` = begin double-spacing
`\single` = begin single-spacing

In order for these commands to work, the `\documentstyle` command at the beginning of the preamble must include the argument [*double*].

7.2 Filling and justifying, and how to avoid them

You type text as though you were using a typewriter. Paragraphs are separated by one or more blank lines in the text.

LaTeX automatically *fills* (closes up extra spaces between words) and *right-justifies* (brings the text flush with the right margin). There are frequent occasions where you don't want to right-justify your text. You can avoid doing so in several ways. One way is to type two backslashes (\\) where you want to end a line. For example, typing

```
Department of Chemical Engineering\\North Carolina State University\\
Raleigh, NC
```

will result in the printing of:

Department of Chemical Engineering
North Carolina State University
Raleigh, NC

If you want to print a large body of text without justifying (i.e., if you want a "ragged right" text), you may do so as follows:

```
\begin{raggedright}
(text)
\end{raggedright}
```

LaTeX will still fill in spaces, but will not stretch out the resulting lines to reach the right margin. However, a caution. If you type

```
\begin{raggedright}
Department of Chemical Engineering
North Carolina State University
Raleigh, NC 27695
\end{raggedright}
```

you will get

Department of Chemical Engineering North Carolina State University Raleigh, NC 27695

Remember, LaTeX still fills in spaces when you use the raggedright command, which is why the three lines you typed got strung together. If you want to avoid this occurrence, end each line with \\ as in the preceding example.

Finally, if you want LaTeX to print what you type exactly as you type it—without justifying or filling in spaces—you may use the *verbatim* environment, described in Section 7.8 on p.

7.3 Paragraphs

LaTeX starts a new paragraph whenever it comes to a blank line in the typed text. It skips the amount of vertical space you told it to skip in the **parskip** command in the preamble, and indents by the amount of horizontal space you specified in the **parindent** command.

7.4 Line breaks and page breaks

Typing \\ at the end of a line causes printing to continue on the next line.

Typing a tilde (~) between two items of text—for example, Figure~3—will assure that a line break does not occur between the items.

Typing `\newpage` causes printing to continue on the next page.

Including the **\newpage** command within a paragraph of text will cause the break after the current line is printed. You can also use the commands `\nolinebreak` and `\nopagebreak` to prevent a break where you definitely don't want one.

LaTeX gets somewhat hysterical about line and page breaks. If it is forced to make a line too short or too long it will do it, and give you an *underfull \hbox* or *overfull \hbox* warning when you translate your file, and if the same thing happens with the vertical text on a page it will do the same thing with *vbox* warnings. You can do something about these problems or ignore the messages when they occur. For more information, see pp. 87-91 of the LaTeX manual.

7.5 Inserting vertical spaces—skipping lines

LaTeX gives you a variety of ways to insert vertical spaces between lines. The easiest ways are with the three commands,

```
\smallskip
\medskip
\bigskip
```

The skip would be produced below the line in which the command appears. The size of the skip is a function of the document style. For purposes of comparison, below this line is the space you normally get by starting a new paragraph.

In the *article* style, the `\smallskip` command produces the vertical space below this line.

The `\medskip` command produces the next vertical space.

The `\bigskip` command produces the next one.

You can also insert a vertical space of any desired size with the command

`\vspace*{`*size*`}`

The *size* argument consists of a number followed by a unit. If you omit the asterisk after `vspace` and the space happens to fall at the top or bottom of a page, the command will be ignored. Examples:

`1in` = 1 inch. The command `\vspace*{1in}` skips a vertical space of one inch.

`2.34cm` = 2.34 centimeters.

`-0.7mm` = –0.7 millimeters. The command `\vspace*{-0.7mm}` causes 0.7 mm to be *subtracted* from the vertical space that would normally appear in the given location.

`2ex` = a length about equal to twice the height of the letter x. This is a good unit to use for vertical spaces.

`3em` = a length about equal to three times the width of the letter M. This is a good unit to use for horizontal spaces.

`30pt` = 30 *points*, a point being a unit of length commonly used in printing. This document is printed in 12 point type.

Finally, the command

`\vfill`

followed by text causes enough vertical space to be inserted to push the subsequent text to the bottom of the page.

7.6 Inserting horizontal spaces

`\hspace* {`*size*`}`

This command causes the insertion of a horizontal space of width *size*. For *size*, substitute a number and unit as in the previous section, e.g. `2.5em`, or `0.5in`. If the asterisk is omitted and the space falls at the beginning or end of a line, the command will be ignored.

`\hfill`

This command puts a horizontal space in a line wide enough to push the text on either side of it out to the margins. For example, typing

```
CHE 205 \hfill Spring 1987
```

leads to

CHE 205	Spring 1987

If more than one `\hfill` is put in a line, the inserted spaces will be made equal. Typing

```
CHE 205 \hfill Chemical Process Principles \hfill Spring 1987
```

leads to

CHE 205	Chemical Process Principles	Spring 1987

Sometimes you may want to put more space between two characters than LaTeX would normally provide. Use a backslash followed by a comma (`\,`) for this purpose. For example,

Typing ’ ” gives you ’”’, but
Typing ’\,” gives you ’ ”

7.7 Centering text

To center several lines of text (line1,line2,...) between the margins of your document, type the following:

```
\begin{center}
line1\\
line2\\
\end{center}
```

You don’t have to put line1, line2, etc., on separate lines as shown above. Stringing them together and separating them with double backslashes will have the same effect.

For example, the authors names and affiliation on p. 1 of this document were produced with the following sequence of statements:

```
\begin{center}

R.M. Felder and D. Riddle\\Department of Chemical Engineering\\
North Carolina State University\\Raleigh, NC 27695

\end{center}
```

7.8 Verbatim—what you type is what you get

```
\begin{verbatim}
(text)
\end{verbatim}
```

Any text you type between the two given commands—including special LaTeX characters—will be printed verbatim, exactly as you type it. For example, typing

```
\begin{verbatim}
Department of Chemical Engineering
North Carolina State University
Raleigh, NC 27695
\end{verbatim}
```

will cause the printing of

```
Department of Chemical Engineering
North Carolina State University
Raleigh, NC 27695
```

The type font shown above is the only kind you can get in the `verbatim` environment.

You can insert verbatim text in the middle of a sentence by using the `\verb` command: Type `\verb`, followed by the desired text enclosed between two identical characters. The enclosing character may be anything except a letter, an asterisk, a space, or a character in the text you want to type.

For example, you might want to print

The cost was $4.25.

but you can't do it directly because $ is one of LaTeX's "forbidden" characters. You could instead type

The cost was `\verb'$'4.25`.

and get what you want. (Instead of the apostrophes in the `verb` command, you could have used ", +, 3, or any other character except a letter, an asterisk, a space, or in this case, $.) (An easier way to do the same thing in this case would be to type `\$4.25`.)

A final point to remember about the verbatim environment is that you can't include LaTeX commands within it. LaTeX will simply print them instead of following them.

8 Changing type style and size

8.1 Italics, boldface, and other type styles

LaTeX offers several type styles to choose from. The following declarations select the indicated type style. (We will indicate shortly how to use the declarations.)

\rm	This is ROMAN type style
\em	*This is ITALIC (or EMPHATIC) type style*
\bf	**This is BOLDFACE type style**
\sl	*This is SLANTED type style*
\sf	This is SANS SERIF type style
\sc	THIS IS SMALL CAPS TYPE STYLE
\tt	`This is TYPEWRITER type style`

Roman is the default type style, and is what you are reading now. Typewriter style is what you get when you use the `\verbatim` command.

If you have a small amount of text (say, two sentences or less) to print in a different type style, you do it by typing a left brace (`{`), followed by the declaration (e.g. \em), followed by a space, followed by the text, followed by a right brace (`}`). For example, typing

```
You can't {\em possibly} mean it.
```

results in

You can't *possibly* mean it.

If you have several lines of text to print in a certain type style, say boldface, you can do it by defining an environment as follows:

`\begin{bf}`
(text to be printed)
`\end{bf}`

Warning: If you forget the closing brace after a type style change, which you occasionally will, the rest of the document will be printed in the type style called for by the declaration. This will irritate you, especially if you're printing, say, a 60-page document, and the omission occurred on page 3. Try to get in the habit of double-checking for closing braces.

8.2 Changing type size

Just as there are declarations for changing type style, there are declarations for changing type size that work in much the same manner. The declarations and the size type they produce for a default 12-point type size are as follows:

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

You can either use a single line declaration or create a multiple-line environment to print in different type sizes. Typing

```
{\Large DEPARTMENT OF CHEMICAL ENGINEERING}
```

leads to

DEPARTMENT OF CHEMICAL ENGINEERING

Typing

```
\begin{scriptsize}
There was a young lady from France\\
Who wanted to learn how to dance\\
\end{scriptsize}
```

leads to

There was a young lady from France
Who wanted to learn how to dance

You can also change both type size and style. Typing

```
{\large\bf MEMO -- \today\ }
```

leads to

MEMO – November 29, 1999

(or whatever the current date is).

For `\large` bold letters, you must type `\large\bf`, not `\bf\large`.

9 Underlining and printing special characters

9.1 Underlining

The **\underline** command puts a horizontal line under its argument, which is enclosed in braces.

`I think \underline{Crime and Punishment} is a million laughs`.

results in

I think <u>Crime and Punishment</u> is a million laughs.

9.2 Quotation marks and dashes

LaTeX distinguishes between right quotes (’) and left quotes (‘). Here’s how to get the one you want, referring to the VT240 keyboard.

’ (single right quote) = third row of keyboard, third key from right
” (double right quote) = upper case of above
‘ (single left quote) = leftmost key, top row
“ (double left quote) = two single left quotes in a row

“‘ (double + single left quote) = ``\,` (i.e., type two single left quotes, backslash, comma, single left quote)

’” (single + double right quote) = '\," (i.e., type a single right quote, backslash, comma, double right quote)

You can produce three different sizes of dash in LaTeX by typing one, two, or three “-” characters.

To print an intra-word dash or hyphen, as in X-ray, type X-ray.
To print a medium dash, as in 1–2, type 1--2.

To print a punctuation dash—like this, type ...dash---like this.

9.3 Subscripts and superscripts

Attaching subscripts and superscripts to characters is one of many features provided by LaTeX for printing mathematical formulas and equations. We will discuss equation typesetting in detail in a separate section; however, since subscripts and superscripts occur so often, we will separately treat them here.

`$base_{subscript}$` produces $base_{subscript}$

`$base^{superscript}$` produces $base^{superscript}$

The first `$` puts you in the `math` environment, which is the only environment in which subscripts and superscripts can be generated, and the second `$` gets you back out of it. The underline character (`_`) specifies that the quantity enclosed in braces is a subscript, and the carat (`^`) denotes superscript.

`$x^{2}$` produces x^{2}

`$T_{ij}$` produces T_{ij}

`$R_{m}^{n}$` produces R_{m}^{n}

Note that when you are in the `math` environment, all letters are printed in italics. To get around this, you can use the `\rm` declaration, as in the next example.

`${\rm T}_{ij}$` produces ${\rm T}_{ij}$

You can put superscripts on subscripts, or any other combination you want, by nesting the commands.

`$x^{y_{1}}$` produces $x^{y_{1}}$

9.4 Greek letters

The commands given in Table 1 are used to generate Greek letters. *If you are in an ordinary text-writing environment when you want to print these letters, put dollar signs (`$`) on both sides of the given commands.* On the other hand, if you are in the **math** environment, in which equations and formulas are usually written, the commands should be used exactly as given.

For example, typing

```
Let $\beta$ denote Finagle's constant.
```

leads to

Let β denote Finagle's constant.

If you subsequently want a β to appear in an equation and you have previously used the **\begin{math}** statement, you only need to type `\beta`, without the dollar signs. (A complete discussion of the **math** environment begins on p. 32.)

9.5 Symbols from other languages

You can make a wide variety of accents and other symbols with LaTeX. The general form is to type a backslash, then the desired accent, then the accented letter enclosed in curly braces. Given below are illustrative examples and the commands used to produce them:

ò	`` \`{o} ``	ó	`\'{o}`	õ	`\~{o}`	o̧	`\c{o}`

If you wish to accent an i or a j, precede it with a backslash inside the curly braces so that the dot will be dropped.

Commands for other foreign symbols are given on p. 40 of the LaTeX manual.

Table 1. Greek Letters.

Lowercase

α	`\alpha`	ι	`\iota`	ϱ	`\varrho`
β	`\beta`	κ	`\kappa`	σ	`\sigma`
γ	`\gamma`	λ	`\lambda`	ς	`\varsigma`
δ	`\delta`	μ	`\mu`	τ	`\tau`
ϵ	`\epsilon`	ν	`\nu`	υ	`\upsilon`
ε	`\varepsilon`	ξ	`\xi`	ϕ	`\phi`
ζ	`\zeta`	o	`o`	φ	`\varphi`
η	`\eta`	π	`\pi`	χ	`\chi`
θ	`\theta`	ϖ	`\varpi`	ψ	`\psi`
ϑ	`\vartheta`	ρ	`\rho`	ω	`\omega`

Uppercase

Γ	`\Gamma`	Ξ	`\Xi`	Φ	`\Phi`
Δ	`\Delta`	Π	`\Pi`	Ψ	`\Psi`
Θ	`\Theta`	Σ	`\Sigma`	Ω	`\Omega`
Λ	`\Lambda`	Υ	`\Upsilon`		

9.6 Calligraphic letters

You can use the `$\cal X$` command, where any letter may be substituted for X, to get **calligraphic** or **script** versions of all upper case letters A,B,...,Z. For example, typing

```
Let $\cal F$(x) be a continuous function.
```

leads to

Let $\mathcal{F}$(x) be a continuous function.

If you are in the **math** environment and want calligraphic letters, you don't need the dollar signs in the command.

The calligraphic versions of all the letters appear as follows:

$\mathcal{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$

9.7 Miscellaneous symbols and characters

LaTeX allows you to print a large variety of special characters, including mathematical symbols. Some of these symbols are listed in Table 2 on p. 32. For a listing of many other characters you can print, see pp. 44-45 of the LaTeX manual.

To print the characters listed in Table 2, you must be in *math mode*. (We will explain what that means when we get to equations.) The easiest way to get into it is by typing a dollar sign; then after typing the instruction to get the symbol you want, type another dollar sign to get back out of math mode.

For example, suppose you want to type the chemical equation A $\longrightarrow$ B. The arrow is one of the symbols listed in Table 2. The required command would be

```
A $\longrightarrow$ B
```

LaTeX types A, enters math mode upon finding the first dollar sign, types the arrow, leaves math mode upon finding the second dollar sign, and types B.

9.8 Printing the current date.

Typing `\today\` causes the printing of the current date. The date on p. 1 was generated in this manner.

10 Printing in columns—tabbing

In the *tabbing* environment, you align text in columns by setting tab stops and tabbing to them, much as you would with a typewriter. The environment is defined as follows:

```
\begin{tabbing}
---- \= ------ \= ----- \= ... \kill
Text \> Text \> Text \\
\> Text \> Text \> Text \\
\end{tabbing}
```

The first line puts you in the tabbing environment. The second line sets the position of the Tab Stops. The \= command, which includes spaces on both sides, causes a tab stop to be located at the corresponding horizontal position on the line. The dashes are just fillers to establish the positions of the tab stops; any characters may be substituted for them. The **\kill** command instructs LaTeX not to print the line, but to use it only to set the tab stops.

In the text, a \> command in a line causes printing to begin at the next tab stop, and as usual, a \\ command at the end of a line causes printing to continue at the left margin on the next line. For example,

```
\begin{tabbing}
------ \= --- \= \kill
C *** THIS IS AN EXCERPT FROM A FORTRAN PROGRAM\\
\> DO 200J=1,15\\
\> \> Y(J) = A*X(J)\\
\> \> SUM = SUM + Y(J)\\
200  \> CONTINUE
\end{tabbing}
```

will produce

```
C *** THIS IS AN EXCERPT FROM A FORTRAN PROGRAM
      DO 200J=1,15
         Y(J) = A*X(J)
         SUM = SUM + Y(J)
200   CONTINUE
```

Notice what happened in the typed text, line by line.

In Line 1, you entered the tabbing environment.

In Line 2, you positioned tab stops at two horizontal positions (Columns 7 and 10).

In Line 3, you began printing at the left margin, and did not do any tabbing.

In Line 4, the \> command moved you to the location of Tab Stop 1, where printing commenced.

In Lines 5 and 6, the first \> command moved you to the location of Tab Stop 1, and the second one moved you to Tab Stop 2, where printing commenced.

In Line 7, you started printing at the left margin, then tabbed to Tab Stop 1, and continued printing.

Let us consider in somewhat greater detail what happens in the tabbing environment. Tab stops are numbered 0,1,2,..., with Tab Stop 0 (the *left margin tab*) always located at the left margin of the page. At the beginning of a line of text within the tabbing environment, a counter, i, the *next tab counter*, is set to 1. Then

`\=` means set the position of Tab Stop i at this horizontal location in the line, and increase i by 1. (Note: you can do this in a real line of text, not only in a "phony" line that you then `\kill`.)

`\>` means tab to Tab Stop i (which must have been set in a previous line), and increase i by 1.

`\\` means (as usual) end the line here. This command also resets i to 1.

`\+` means print subsequent lines beginning at Tab Stop 1 (or Tab Stop 2, if two such commands have been given.) This has the effect of indenting a portion of the text.

`\-` means print subsequent lines beginning at one lower Tab Stop than you are currently using. This command offsets a previous `\+` command.

There are several other things you can do in the tabbing environment, including printing text at different locations in the region between tab stops, and changing to new tab settings and then recovering the previous settings. For details, consult pp. 180-181 of the LaTeX manual.

`\vspace` commands in a tabbing environment will be ignored. If you want to get vertical spacing between lines, include a *strut* in the second line, i.e. a command of the form `\rule{0mm}{7mm}`. The 7mm in the command can be changed to any height; the larger the number, the greater the vertical separation between lines. For example, the set of commands,

```
\begin{tabbing}
Top line of text \= \\
Second line \> no strut \\
Third line \> 5mm strut \rule{0mm}{5mm} \\
Fourth line \> 7mm strut \rule{0mm}{7mm} \\
Fifth line \>  10mm strut \rule{0mm}{10mm}
\end{tabbing}
```

will produce the following text:

Top line of text
Second line   no strut

Third line   5mm strut

Fourth line   7mm strut

Fifth line   10mm strut

For really fancy looking tables, with vertical and horizontal rules and variable column widths or whatever, you may be better off using the more complex but more versatile **tabular** environment, which is discussed on p. 47 of this document.

11 Lists

A *list* is a series of items, each item being marked with a label. The following list of characteristics is an example:

1. The labels in a list can be numbers, as in this example, or bullets (•), or anything else you want them to be.
2. LaTeX automatically indents Lines 2,3,... of each item in a list (as it did in Item 1).
3. A list should have at least two items.
4. Lists can be nested within one another.
   (a) Here, too, the indentations are done automatically, as in Item (b).
   (b) LaTeX permits at least four levels of nested lists, which is more than enough for anything you are ever likely to do.
5. Blank lines before an item have no effect

LaTeX provides three list-making environments: **enumerate**, **itemize**, and **description**. In the **enumerate** environment, the items of a list are numbered (or lettered) sequentially. In the **itemize** environment, bullets (or any other symbol you choose) are used. In the **description** environment, you provide individual labels for each item. In the sections that follow we discuss each of these environments.

11.1 The enumerate environment

```
\begin{enumerate}
  \item First item on list
  \item Second item on list
    \begin{enumerate}
       \item First item on nested sublist
       \item Second item on nested sublist
```

```
    \end{enumerate}
  \item Third item on list
\end{enumerate}
```

produces

1. First item on list

2. Second item on list

   (a) First item on nested sublist

   (b) Second item on nested sublist

3. Third item on list

Note: You don't need to indent the listed items in your typed text: LaTeX does its own indentation regardless of what you do. We indented simply because it looks neater.

The items on your list are normally labeled with arabic numerals (as in the above examples), but you can also get upper and lower case roman numerals or upper or lower case letters. The following command just before the **begin{enumerate}** command sets up the labeling for the outermost list, or for the only list if you aren't doing any nesting (the trailing parentheses show you what the labels will be—they are not part of the commands);

`\renewcommand{\labelenumi}{\arabic{enumi}.}`	(1., 2., 3.,...)
`\renewcommand{\labelenumi}{\roman{enumi}.}`	(i., ii., iii.,...)
`\renewcommand{\labelenumi}{\Roman{enumi}.}`	(I., II., III.,...)
`\renewcommand{\labelenumi}{\alph{enumi}.}`	(a., b., c.,...)
`\renewcommand{\labelenumi}{(\alph{enumi})}`	[(a), (b), (c),...]
`\renewcommand{\labelenumi}{\Alph{enumi}.}`	(A., B., C.,...)

Several points should be noted about these commands.

(a) The first command (arabic) is the default for the outermost list (the *first nesting level*). You will only need it if you previously used one of the other label styles in a list, and you want to go back to arabic numerals for this one.

(b) The period after **{enumi}** is the period after the number or letter in the label. If you don't want a period in the label, omit it from the command.

(c) The default labels are [(a),(b),...] for the second nesting level, [i.,ii.,...] for the third, and [A.,B.,...] for the fourth.

(d) If you want to change the labeling for the second nesting level, replace **enumi** with **enumii** in the command. Similarly, replace it with **enumiii** or **enumiv** for the third or fourth nesting level, if you ever use that many.

Note: In the file used to generate this document, the list you just read was preceded by the command:

```
\renewcommand{\labelenumi}{(\alph{enumi})}
```

11.2 The itemize environment

```
\begin{itemize}
  \item First item on list
  \item Second item on list
    \begin{itemize}
      \item First item on nested sublist
      \item Second item on nested sublist
    \end{itemize}
   \item Third item on list
\end{itemize}
```

produces

- First item on list
- Second item on list
  - First item on nested sublist
  - Second item on nested sublist
- Third item on list

The default labels produced by the itemize environment are bullets (•) for the first nesting level, dashes (–) for the second level, asterisks (∗) for the third level, and centered dots (·) for the fourth level.

If you want to change the label for the first nesting level, use the following command before **begin{itemize}**:

```
\renewcommand{\labelitemi}{char}
```

Replace *char* in this command with the command to generate the character you want to use as a label. To change the default label on the second, third, or fourth nesting level, replace **\labelitemi** with **\labelitemii**, **\labelitemiii**, or **\labelitemiv**, respectively.

11.3 The description environment

In the **description** environment, each **\item** command is followed by the desired label enclosed in square brackets, and then the text of the item.

```
\begin{description}
  \item [contra] {\em prep}. against; before; opposite.  {\em Remar contra a
mare}, to pursue one's course, to persist in spite of all opposition.
  \item [contradi\c{c}\~{a}o] {\em noun, fem.} contradiction.
  \item [contrap\^{e}so] {\em noun, masc.} counterbalance.
\end{description}
```

produces

contra *prep.* against; before; opposite. *Remar contra a mare*, to pursue one's course, to persist in spite of all opposition.

contradição *noun, fem.* contradiction.

contrapêso *noun, masc.* counterbalance.

11.4 Changing the formatting of a list

You can modify a number of things about the appearance of a list, such as how much the labels are indented, how much space there is between item labels and text, how much vertical space is inserted between items, etc. The parameters that must be given new values to modify the default values are listed on pp. 166-167 of the LaTeX manual. Their values may be changed using the **\setlength** command, described on p. 95 of the LaTeX manual.

12 Equations and formulas

LaTeX is extremely good at producing mathematical symbols, formulas, and equations. It must be in one of four environments to do so: **math** (for in-text formulas), **displaymath** (for single line unnumbered equations), **equation** (for single line numbered equations), and **eqnarray** (for multiple line equations). We will say that LaTeX is in *math mode* when it is in any of these environments.

You have already been introduced to some of the things LaTeX can do so in math mode, including producing Greek and calligraphic letters, superscripts, and subscripts. In this section we will discuss how it can be used to build complex equations such as

$$S(x) = \sum_{n=1}^{\infty} \frac{e^{-nx}\sin(n\pi x)}{1 + \frac{x^2}{n}}$$

12.1 In-text formulas: the Math environment

The **math** environment is used to produce formulas within text. It is defined by enclosing the desired formula between dollar signs ($formula$) or by putting it between the commands `\begin{math}` and `\end{math}`. For example, typing either

```
Pythagoras's Theorem has the form $x^{2} + y^{2} = h^{2}$.
```

or

```
Pythagoras's Theorem has the form
\begin{math}
x^{2} + y^{2} = h^{2}.
\end{math}
```

produces

Pythagoras's Theorem has the form $x^2 + y^2 = h^2$.

Three points to note about math mode.

(a) Spaces you type are generally closed up in the printed formula. (See the above example.) Include the spaces to make your program easier to read.

(b) The default type style in math mode is a type of italics called *math italic*. If you wish to print formulas in ordinary roman type, you will have to use the `\rm` declaration to do so.

(c) Sooner or later it occurs to everyone that typing `$text$` is a slightly easier way to get something in italics than typing `{\em text}`. There are two problems with doing it this way. The first is that in the **math** environment ($...$) all spaces are closed up, so if you have more than one word to italicize you won't get what you want. The second is that **em** italics are somewhat different from **math** italics. For example,

Typing `{\em different style}` produces *different style*, while

Typing `$different style$` produces $different style$.

When you go ahead and use `$...$` anyway and it doesn't work out, don't say you weren't warned.

(d) **SUGGESTION**. When you do complicated equations in math mode, you will have to enclose many things in braces (`{}`), including subscripts, superscripts, numerators and denominators of fractions, etc. Being human, you will occasionally forget to type the closing right brace. Disaster may result–the program won't run, and you may have a hard time tracking down the mistake. One way to avoid this problem is each time you type a left brace, type the closing right brace immediately, and then type what you need between them. (The same thing should be done for the dollar signs that get you in and out of math mode.)

12.2 Displayed equations: the Displaymath and Equation environments

To print a displayed formula or equation (i.e., one set off on its own line and centered on the page):

- Enclose it between `\begin{displaymath}` and `\end{displaymath}` commands or between `\[` and `\]` commands if you do not wish the equation to be numbered.
- Enclose it between `\begin{equation}` and `\end{equation}` commands if you wish numbering. LaTeX automatically increments the equation number for each equation you print in this environment.

For example, typing

```
\begin{displaymath}
x^{2} + y^{2} = h^{2}
\end{displaymath}
```

or simply

```
\[ x^{2} + y^{2} = h^{2} \]
```

results in the printing of the displayed equation

$$x^2 + y^2 = h^2$$

No equation number is printed in the **displaymath** environment. If you want to number the displayed equation, you would print it in the **equation** environment. Typing

```
\begin{equation}
x^{2}+y^{2}=h^{2}
\end{equation}
```

leads to

$$x^2 + y^2 = h^2 \tag{1}$$

Once this equation was printed, an internal counter was advanced by LaTeX from 1 to 2. The next equation printed in the **equation** environment (or the **eqnarray** environment) would then be numbered (2), and so on. Thus if you add or delete equations, you don't have to renumber them; LaTeX automatically does it for you.

The reason the equation was numbered (1) is that we are in **Article** style. If we were in **Report** style and in Chapter 1 of the text, the equation would have been numbered (1.1).

When you are typing equations in displaymath or equation mode do not press the return key. LaTeX doesn't like it, and may return an error message.

The next sections discuss the features LaTeX provides for building complex formulas and equations. Most of the discussion applies to all three environments discussed so far, although once you get much beyond simple equations like Equation (1) you are better off displaying formulas rather than trying to cram them into your text with the **math** environment.

12.3 Mathematical symbols

12.3.1 A glossary of symbols

Some common characters that can be printed in math mode—most notably, Greek letters—have already been discussed. Listed in Table 2 are other characters and the commands used to get them. Still more can be found on pp. 44-45 of the LaTeX reference manual. *You must be in math mode to use these commands.*

For example, typing

```
If $\log x \leq \epsilon$ then $\partial y / \partial x = 20 \pm 0.1$
```

leads to

If $\log x \leq \epsilon$ then $\partial y / \partial x = 20 \pm 0.1$

The third column in the table needs a word of explanation. The words shown are "log-like" functions, which you would normally want to print in roman type when they occur in a formula. Unfortunately, in math mode, LaTeX would consider `log` to be the product of three variables, l, o, and g. Thus, for example, typing `$log xy$` leads to $logxy$. The command `\log` enables you to get around this problem: typing `$\log xy$` leads to $\log xy$.

Additional log-like functions include other trigonometric functions (cot, csc, sec, cosh, sinh, tanh, arccos, arcsin, arctan) and functions of interest primarily to mathematicians (arg, dim, inf, limsup, and others).

12.3.2 Square roots and other roots

The command **\sqrt{}** is used to produce the square root symbol. The overline is extended to cover whatever is between the braces. An optional argument in square brackets **\sqrt[n]{}** can be used to produce the nth root of whatever is between the braces. For example,

Typing `$\sqrt{b^{2}-4ac}$` leads to $\sqrt{b^{2}-4ac}$

Typing `$\sqrt[3]{x+y}$` leads to $\sqrt[3]{x+y}$.

As before, if you are already in math mode you should omit the dollar signs.

12.3.3 Sums ($\sum$), products ($\prod$), and integrals ($\int$)

The symbols shown above are obtained with the **\sum**, **\prod**, and **\int** commands, respectively. Subscript and superscript arguments following the commands allow you to print limits on the sums, products, and integrals.

Typing `\[ \sum_{i=1}^{n} x_{i} \]` produces

$$\sum_{i=1}^{n} x_{i}$$

Table 2. Mathematical Symbols and Special Characters

To get	Type	To get	Type
$\bullet$	`$\bullet`	$\leftarrow$	`$\leftarrow`
$\star$	`$\star`	$\rightarrow$	`$\rightarrow`
$\cdot$	`$\cdot`	$\Leftarrow$	`$\Leftarrow$`
$\diamond$	`$\diamond`	$\Rightarrow$	`$\Rightarrow`
†	`\dag`	$\longleftarrow$	`$\longleftarrow`
$\equiv$	`$\equiv`	$\longrightarrow$	`$\longrightarrow`
$\approx$	`$\approx`	$\longleftrightarrow$	`$\longleftrightarrow`
$\neq$	`$\neq`	$\Longleftrightarrow$	`$\Longleftrightarrow`
$\Box$	`$\Box`	$\Uparrow$	`$\Uparrow`
$\heartsuit$	`$\heartsuit`	$\Downarrow$	`$\Downarrow`
©	`\copyright`	¶	`\P`

$\leq$	`\leq`	$\times$	`\times`	$\log$	`\log`
$\geq$	`\geq`	$\div$	`\div`	$\ln$	`\ln`
$\ll$	`\ll`	$\pm$	`\pm`	$\exp$	`\exp`
$\gg$	`\gg`	$\mp$	`\mp`	$\min$	`\min`
$\neq$	`\neq`	∇	`\nabla`	$\max$	`\max`
$\equiv$	`\equiv`	$\bigtriangleup$	`\bigtriangleup`	$\sin$	`\sin`
$\sim$	`\sim`	∞	`\infty`	$\cos$	`\cos`
$\simeq$	`\simeq`	$\emptyset$	`\emptyset`	$\tan$	`\tan`
$\approx$	`\approx`	$\surd$	`\surd`	$\lim$	`\lim`
$\cong$	`\cong`	$\aleph$	`\aleph`	$\propto$	`\propto`
∂	`\partial`	$\therefore$	`\stackrel{\textstyle{\cdot}}{.\,\,.}`		

Typing `\[ \prod_{i=1}^{n} x_{i} \]` produces

$$\prod_{i=1}^{n} x_i$$

Instead of enclosing the given typed expressions in `\[...\]`, you could have put them between `begin{displaymath}` and `end{displaymath}` commands, or, if the sum or product is part of a numbered equation, between `begin{equation}` and `end{equation}` commands. For example, typing

```
\begin{displaymath}
\Phi (y) = \int_{0}^{y} e^{-x^{2}} \, dx
\end{displaymath}
```

produces

$$\Phi(y) = \int_0^y e^{-x^2} \, dx$$

In the last equation, the `\,` was used to put a little extra space between the integrand and dx.

The sum, product, and integral symbols in these examples are shown as they appear in the **displaymath** and **equation** environments. They look different in the **math** environment. For example, typing

`$\sum_{i=1}^{n} x_{i} = \int_{0}^{1}f \, dx$`

produces

$\sum_{i=1}^{n} x_i = \int_0^1 f \, dx$.

(Compare the appearances of the sum and integral signs here with those in the previous examples.)

12.3.4 Overlines, underlines, and braces

The **\overline{}** and **\underline{}** commands put horizontal lines above and below their arguments, respectively.

Typing `$\underline{x+y}$` produces $\underline{x+y}$

Typing `$\overline{\overline{x}^{2} + 1}$` produces $\overline{\overline{x}^{2} + 1}$

Horizontal braces are put above or below expressions with the **\overbrace** and **\underbrace** commands, respectively.

Typing `$\overbrace{a + \underbrace{b + c} + d}$` produces $\overbrace{a + \underbrace{b + c} + d}$

In a displayed formula, a subscript or superscript attached to the command argument puts a label on the brace.

Typing `\[ \underbrace{a + \overbrace{b + \cdots + y}^{24} + z}_{35} \]` produces

$$\underbrace{a + \overbrace{b + \cdots + y}^{24} + z}_{35}$$

12.3.5 Accents, tildes, and hats

A number of different symbols can be put above letters. The next table shows the required commands, using the letter x as an illustration. *You must be in math mode to use these commands.*

$\hat{x}$	`\hat{x}`	$\tilde{x}$	`\tilde{x}`
$\check{x}$	`\check{x}`	$\bar{x}$	`\bar{x}`
$\breve{x}$	`\breve{x}`	$\vec{x}$	`\vec{x}`
$\acute{x}$	`\acute{x}`	$\dot{x}$	`\dot{x}`
$\grave{x}$	`\grave{x}`	$\ddot{x}$	`\ddot{x}`

You can use **widehat** and **widetilde** commands to produce hats and tildes that extend over multiple character arguments.

12.3.6 Ellipses

LaTeX can produce four types of ellipsis:

Typing `\ldots` produces $\ldots$

Typing `\cdots` produces $\cdots$

Typing `\vdots` produces $\vdots$

Typing `\ddots` produces $\ddots$

The `\ldots` command works in any mode, but the other three commands can be used only in math mode. (Bracket the commands with dollar signs if you are not already in math mode.)

Table 3. Delimiters

<table>
<tr><td>$($</td><td><code>(</code></td><td>$)$</td><td><code>)</code></td><td></td><td></td></tr>
<tr><td>$[$</td><td><code>[</code></td><td>$]$</td><td><code>]</code></td><td></td><td></td></tr>
<tr><td>$\{$</td><td><code>\{</code></td><td>$\}$</td><td><code>\}</code></td><td></td><td></td></tr>
<tr><td>$\lfloor$</td><td><code>\lfloor</code></td><td>$\rfloor$</td><td><code>\rfloor</code></td><td></td><td></td></tr>
<tr><td>$\lceil$</td><td><code>\lceil</code></td><td>$\rceil$</td><td><code>\rceil</code></td><td></td><td></td></tr>
<tr><td>$\langle$</td><td><code>\langle</code></td><td>$\rangle$</td><td><code>\rangle</code></td><td></td><td></td></tr>
<tr><td>$\vert$</td><td><code>|</code></td><td>$\Vert$</td><td><code>\|</code></td><td></td><td></td></tr>
<tr><td>$\uparrow$</td><td><code>\uparrow</code></td><td>$\downarrow$</td><td><code>\downarrow</code></td><td>$\updownarrow$</td><td><code>\updownarrow</code></td></tr>
<tr><td>$\Uparrow$</td><td><code>\Uparrow</code></td><td>$\Downarrow$</td><td><code>\Downarrow</code></td><td>$\Updownarrow$</td><td><code>\Updownarrow</code></td></tr>
</table>

12.4 Building complex formulas and equations

12.4.1 Fractions in displayed formulas

The **\frac**{*num*}{*denom*} command is used to produce fractions in displayed formulas. In place of the first argument (*num*) you would type the numerator of the desired fraction, and in place of the second argument (*denom*) you would type the denominator. You can also imbed fractions within fractions, as the second example below shows.

Typing `\[ x = \frac{y + z/2}{y^{2} + 1} \]` produces

$$x = \frac{y + z/2}{y^2 + 1}$$

Typing `\[ \frac{ e^{xy} }{1 + \frac{y}{x+1}} \]` produces

$$\frac{x + y}{1 + \frac{y}{z+1}}$$

12.4.2 Parentheses, brackets, braces, and other delimiters

Several symbols can be used as *delimiters* of mathematical expressions, and can be made as large as they have to be to enclose the expression. The delimiters and the commands used to produce them listed in Table 3.

To make the delimiters big enough to fit the expressions they enclose, use the **\left** and **\right** commands. The words *left* and *right* are immediately followed by the desired left and right delimiters. Typing

`\left(` ⟨*expression*⟩ `\right)`

would cause the expression to be enclosed in parentheses of a suitable size. *The* **\left** *and* **\right** *commands must come in matching pairs*, although the matching delimiters need not be the same. For example, typing

```
\[ Y = 1 + \left[ \frac{x+3}{2x^{2}+4x-5} \right] \]
```

produces

$$Y = 1 + \left[\frac{x+3}{2x^{2}+4x-5} \right]$$

If you want a left delimiter but not a right one or vice versa, you must still include both **\left** and **\right** commands, but *you can make an invisible delimiter by typing a period after the* **\left** *or* **\right** *command.* Typing

```
\[ \left. \frac{x-3}{2x+7} \right]^{1}_{0} \]
```

produces

$$\left. \frac{x-3}{2x+7} \right]^{1}_{0}$$

Finally, you can make delimiters taller than the text they are supposed to bracket by adding *struts* to the text—in effect, characters of height h and zero width (so they are invisible). The command for producing a strut of height 5 millimeters is

```
\rule{0mm}{5mm}
```

Thus, for example, typing

```
\[ \left\Downarrow \rule{0mm}{12mm} x= \frac{1-y}{1+y} \right. \]
```

produces

$$\left\Downarrow \rule{0mm}{12mm} x= \frac{1-y}{1+y} \right.$$

If the strut were omitted from this command, the arrow would have been the same height as the fraction.

12.4.3 Stacking symbols

The **stackrel** command stacks one symbol above another. The command syntax is as follows:

`\stackrel{`*top*`}{`*bottom*`}`

where the arguments *top* and *bottom* would be replaced by commands to produce the upper and lower items. For example, typing

```
A $\stackrel{k_{1}}{\rightarrow}$ B
```

produces

A $\stackrel{k_{1}}{\rightarrow}$ B

The first (upper) argument is normally printed in small ("*scriptsize*") italic type, as above. If you want it printed in normal size italic type use the **\textstyle** command after the first curly bracket, and if you want it in roman use the **\rm** command. For example,

`A $\stackrel{\textstyle\rm  k_{1}}{\longrightarrow}$ B` $\Rightarrow$ A $\stackrel{\textstyle\rm k_{1}}{\longrightarrow}$ B

It is possible to nest stackrels. When you do, the bottom–most symbol will be printed flush with the current line of text: if you don't want it there you must use a `\raisebox` command, as the next example illustrates.

Typing

```
\begin{displaymath}
A\,\,\raisebox{-2.5ex}{$\stackrel{\stackrel{\textstyle k_{1}}
{\textstyle{\longleftarrow }}}{\stackrel{\textstyle{\longrightarrow}}
{\textstyle{k_{2}}}}$} \,\, B
\end{displaymath}
```

produces

$$A \,\, \stackrel{\stackrel{\textstyle k_{1}}{\textstyle{\longleftarrow}}}{\stackrel{\textstyle{\longrightarrow}}{\textstyle{k_{2}}}} \,\, B$$

Boxes (such as the raisebox) are discussed in Section 15.4. The second argument of a box command (i.e., the text enclosed within the second set of braces) is not normally in math mode, even if the box is within a math environment. That is why the second argument of the raisebox in this example had to be enclosed between $ signs.

A relatively complex illustration of the stackrel command concludes this section. Typing

```
\begin{displaymath}
W^{min} = -T \Delta\;S_{separation} \,\, = \,\,
\underbrace{-RT \, \sum_{i} \, x_{i} \, \ln
x_{i}}_{\stackrel{\stackrel{\uparrow}{always \;\; > \; 0}}{dependent \; only
\; on\; composition \; at \; (T,P)}
\end{displaymath}
```

produces

$$W_{min} = -T\Delta\;S \,\, = \,\, \underbrace{-RT \, \sum_{i} \, x_{i} \, \ln x_{i}}_{\stackrel{\stackrel{\uparrow}{always \;\; > \; 0}}{dependent \; only \; on\; composition \; at \; (T,P)}}$$

12.4.4 Changing type in math mode

Letters are normally printed in italics in math mode, usually in normal size ("textstyle"), sometimes in small type ("scriptstyle"), as in subscripts, superscripts, and as the top member of a *stackrel* pair. The commands `\textstyle`, `\scriptstyle`, `\rm` and `\bf` can be used to print letters in normal size, small size, roman (ordinary) type, and boldface. For example,

`$y = x^{2}$` $\Rightarrow$ $y = x^2$

`$ {\rm y} = x^{\textstyle 2}$` $\Rightarrow$ ${\rm y} = x^{\textstyle 2}$

12.4.5 Spacing in math mode

You can add space between characters in math mode by using the following commands:

`\,` $\Rightarrow$ insert thin space. Typing `$a \, b$` produces $a \, b$.

`\:` $\Rightarrow$ insert thick space. Typing `$a \: b$` produces $a \: b$.

In comparison, simply typing `$ab$` produces ab. The `\,` command works in any mode, while the other two commands and the next one work only in math mode.

You can delete a small amount of space between characters with the `\!` command.

Typing `$n / \log n$` produces $n / \log n$, while
Typing `$n / \! \log n$` produces $n / \! \log n$.

12.5 Multiple-line equations

The **eqnarray** and **eqnarray*** environments are used to produce one equation or several consecutive equations that occupy more than one line. The first environment generates equation numbers, the second does not. For example,

$$
\begin{aligned}
y &= 3.0 \times 10^{-5} \exp(-12\,000/T) && (2)\\
\\
\Phi(x) &= 5 - 2x \quad (x \leq 2.5) && (3)\\
&= 0.0 \qquad\quad (x > 2.5) && (4)\\
\\
\mathcal{F} + \mathcal{G} &\leq a + b + c + d + e + f + g + \\
&\quad h + i + j + k + l + m && (5)
\end{aligned}
$$

The environment to produce equations like these is defined by the sequence

`\begin{eqnarray}`
(1-3 command sequence for each equation)
`\end{eqnarray}`

The commands to generate an equation in this environment have the form

(Left side of equation) `&` (equality or inequality symbol) `&` (right side of equation) `\\`
`& &` (continuation of right side of equation) `\\`
`\mbox{} \nonumber \\` (Optional—inserts extra space after equation)

Note: The `\\` should be omitted from the last equation in the environment.

For example, Eqs. 2– 5 shown above were produced with the commands

```
\begin{eqnarray}
y & = & 3.0 \times 10^{-5} \exp (-12 \, 000/T) \\
\mbox{} \nonumber \\
\Phi (x) & = & 5 - 2x  \hspace{5 mm} (x\leq2.5) \\
  & = & 0.0  \hspace{11 mm} (x>2.5)\\
\mbox{} \nonumber \\
{\cal F} + {\cal G} & \leq & a + b + c + d + e + f + g +  \nonumber \\
  & & h + i + j + k + l + m
\end{eqnarray}
```

Note the following points:

- Each line in the **eqnarray** environment contains three parts separated by `&` symbols, and ends with \\. The three parts correspond to the left side of the equation, an equality or inequality symbol, and the right side of the equation.

- LaTeX normally puts equation numbers on every line produced in an **eqnarray** environment. If you don't want an equation number on a particular line, put a **\nonumber** command before the \\ on that line. If you don't want to generate any equation numbers, use `eqnarray*` instead of `eqnarray` to define the environment.

- If you continue on the second line of an equation with another equality or inequality, as in Eq. 4 above, begin the second line with a single `&`.

- If you continue an equation on a second line, as in Eq. begin the second line with two `&` symbols. If you don't, LaTeX will think you are beginning a new equation, and the spacing will be wrong. For instance, if we had omitted the two `&`'s in the sixth typed line of the above example, the output would have been:

$$\begin{aligned} \mathcal{F}+\mathcal{G} &\leq a+b+c+d+e+f+g+ \\ h+i+j+k+l+m & \qquad (6) \\ & \qquad (7) \end{aligned}$$

- If in the above equation you wanted to end the first line with g (instead of $+$) and begin the second line with $+$, LaTeX would have considered $+h$ a single unit and would not have put a space between the two characters. To get the space, begin the second typed line of the equation with `& & \mbox{} + h +....`

- LaTeX doesn't put any extra space between equations printed in an **eqnarray** environment. You can skip a line (as we did) by including the `\mbox{} \nonumber \\` command.

- When you use the given commands to generate an equation that occupies several lines (like Eqn. 5 above), Lines 2,3,... are always printed to the right of the `=` sign (or whatever sign is in the middle position) on the first line. Sometimes you may prefer to print the additional lines further to the left, e.g.

$$\begin{aligned} &\mathcal{R}(y)+\int_0^y \Psi(x)\,dx = \\ &\qquad a+b+c+d+e+f+g+ \\ &\qquad h+i+j+k+l+m+o+p+q \end{aligned}$$

You can do this by typing the following commands within the **eqnarray** environment:

`\lefteqn{`(left side and equality or inequality symbol)`} \\`
`&  &` (right side of equation) `\\`

For example, the preceding equation was produced by typing

```
\begin{eqnarray*}
\lefteqn{{\cal R}(y) + \int_{0}^{y} \Psi (x) \, dx =} \\
  & & a + b + c + d + e + f + g +  \\
  & & h + i + j + k + l + m + o + p + q
\end{eqnarray*}
```

- *Warning*: Don't put blank lines in an **eqnarray** environment—LaTeX doesn't like it, and will throw you off with some indecipherable error messages if you try.

12.6 Cross-referencing equations

If you assign a *key* (i.e., a label) to a numbered equation, you can make reference to that key whenever you cite that equation in the text, and LaTeX will automatically insert the correct equation number. (A similar system is used to cross-reference section numbers.) The syntax is

`\label{`*key*`}`   (Attach to equation)

`~\ref{`*key*`}`   (Insert in text where equation is cited)

Any character string may be substituted for the argument *key* of these commands. For example, suppose you type

```
\begin{equation}
Y = 4x^{2} - 3x + 5  \label{yquad}
\end{equation}
```

and somewhere else in the text you type

```
Substitute for $x$ in Eq.~\ref{yquad} to obtain the desired result.
```

LaTeX would print

$$Y = 4x^2 - 3x + 5 \tag{8}$$

and

Substitute for x in Eq. 8 to obtain the desired result.

Note: It takes two passes through the document file to get this feature to work. The first time you translate the file (`$ LATEX NAME`) with the labels and citations included, LaTeX assigns numbers to keys, but can only insert question marks where the citations occur. The second time, the citations will be substituted from the previous runthrough.

13 Tables

13.1 The tabular environment

LaTeX does an impressive job of generating tables—organizing things in rows and columns, printing vertical and horizontal rules, and keeping track of captions for inclusion in a comprehensive listing similar to a table of contents. The format of the **tabular** environment, in which all this is done, is as follows:

begin{center}
begin{tabular}{*cols*} \hline
(text) & (text) & (text) &... \\ *arg*
(text) & (text) & (text) &... \\ *arg*
end{tabular}
end{center}

The first argument of the \begin{tabular} command (*cols*) specifies the column formatting. It consists of a series of the following specifiers:

l	(locate the entries flush left)
c	(center the entries)
r}	(locate the entries flush right)
|	(draw a vertical line the height of the table)
||	(draw a double vertical line)

Each row of the table consists of column entries separated by & symbols, followed by \\, possibly followed by additional arguments (*arg*).

Given below are commands to generate a table and the table produced by the commands.

```
\begin{center}
\begin{tabular}{|l||ccr|} \hline
Adams, John  &  I17 &  S28  &  87 \\
Adams, Cynthia &  E38 &  N17  &  68 \\ \cline{2-4}
Williams, George & I25 & N7 & 123 \\
Williams, Joan & I13 & N38 & 72 \\ \hline
\end{tabular}
\end{center}
```

Adams, John	I17	S28	87
Adams, Cynthia	E38	N17	68
Williams, George	I25	N7	123
Williams, Joan	I13	N38	72

Note the following points:

- LaTeX normally locates tables at the left margin. If you want a table centered, you must enclose the **tabular** environment in a **center** environment.
- The **\hline** command of the `\begin{tabular}` statement is optional. It instructs LaTeX to draw a horizontal line the width of the table above the first row.
- You must include \\ at the end of the last row in the table.
- The following arguments may be substituted after \\:

  \hline = draw a horizontal line across the complete table below this row. If two of these commands are typed in succession, two lines are drawn; vertical rules do not appear in the space between them.

  \cline{i-j} = draw a horizontal line across Columns i through j, inclusive. If you want the line only across Column 3, then substitute 3-3 as the argument. If you want several horizontal lines across different columns, put these commands one after another following \\.
- The command `\vline` after an entry (but before `&`) causes a vertical line the height of the row to be printed.
- LaTeX will ignore `\vspace` commands in a tabular environment. If you want vertical space in a row above a table, include a *strut* somewhere in that row—i.e., a command of the form `\rule{0mm}{7mm}`. Change the 7 to a higher number to get more vertical space, to a lower number to get less.
- You can print an item across several adjacent columns by including a **multicolumn** command within a row. The command has the syntax:

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

  This command makes *item* the text of a single entry spanning *num* columns, positioned as specified by *col*. You would use `&` symbols before the command to begin the entry in the desired column.

  The argument *col* must contain one `l`, `c`, or `r`, depending on where in its space you want the entry to be located (flush left, centered, or flush right). In addition, include `|`'s where you want vertical rules, since the `\multicolumn` command may override the requests for vertical rules in the `begin{tabular}` command.

Following are another set of commands and the table they generate.

```
\begin{center}
\begin{tabular}{|l|cc|cr|} \hline
& \multicolumn{2}{|c|}{Feed} & \multicolumn{2}{c|}{Products} \\
Substance & $n$ & $H$ & $n$ & $H$ \\ \hline
Ethanol & 90 & 30.3 & $n_{1}$ & 26.4 \\
Acetaldehyde & 10 & 22.0 & $n_{2}$ & 19.2 \\
Hydrogen & --- & --- & $n_{3}$ & 7.0 \\ \hline
\end{tabular}
\end{center}
```

	Feed		Products	
Substance	n	H	n	H
Ethanol	90	30.3	n_1	26.4
Acetaldehyde	10	22.0	n_2	19.2
Hydrogen	—	—	n_3	7.0

Two additional specifiers (besides `l`, `c`, `r`, and `|`) may be included in the `\begin{tabular}` command.

`@{\hspace{size}}` Inserts a horizontal space of the specified width at the designated position in the table. (The term `size` would be replaced by a width specification, e.g. 5mm or 2em.)

`p{size}` This specifies the width of a column (again, replace `size` with the desired column width) and allows multiple lines of text to be printed in each row of that column.

The following example illustrates the use of these specifications.

```
\begin{center}
\begin{tabular}{|l@{\hspace{4em}}||c@{\hspace{4em}}|p{2.75in}|} \hline
\multicolumn{3}{|c|}{\bf Research Grant Funding} \\ \hline\hline
& \multicolumn{1}{||c|}{\bf Funding Agency} & \\
\multicolumn{1}{|c}{\bf P.I.(s)} &
\multicolumn{1}{||c|}{\bf (\$thousands)} &
\multicolumn{1}{c|}{\bf Grant Title} \\ \hline
Ayres & NCBC (20) & Purification of Biological Materials by Counteracting
```

```
Chromatographic  Electrophoresis \\ \hline
Ramos & NCBC (450) & Protein Solubility:  Application in Precipitation
and Aqueous Two-Phase Extraction with Liposomal Studies \\ \hline
\end{tabular}
\end{center}
```

Research Grant Funding		
P.I.(s)	**Funding Agency ($ thousands)**	**Grant Title**
Ayres	NCBC (20)	Purification of Biological Materials by Counteracting Chromatographic Electrophoresis
Ramos	NCBC (450)	Protein Solubility: Application in Precipitation and Aqueous Two-Phase Extraction with Liposomal Studies

13.2 The table environment

The **table** environment is used for two things:

(a) To guarantee that a pagebreak does not occur in the middle of a table (or any desired block of text), or that no half-empty pages are left because a table has to be moved to the next page.

(b) To generate numbered table captions and enable LaTeX to keep track of them for subsequent inclusion in a list.

The following commands set up this environment.

(Text1, containing the first reference to the table)
`\begin{table}`
(Text to generate the table, usually with the **tabular** environment)
`\caption{`(text)`}`
`\end{table}`
(Text2)

LaTeX would then "float" the table to the top of the current page or the top of the next page. If there are no blank lines in the typed text, the two segments *Text1* and *Text2* will be printed with no break between them.

Year	Price
1981	\$3.50
1982	\$4.25
1983	\$5.80
1984	\$7.25

Table 1: Average price of coffee, \$/lb

Given below are a set of typed commands and the text they generate. Recall that the tilde (~) in Table~1 is there to prevent a line break between `Table` and `1`. In this case, the table was floated to the top of this page.

```
In Table~1 we see that the average price doubled in less than three years.
\begin{table}
\begin{center}
\begin{tabular}{|cr|} \hline
Year & Price \\ \hline
1981 & \$3.50 \\
1982 & \$4.25 \\
1983 & \$5.80 \\
1984 & \$7.25 \\ \hline
\end{tabular}
\caption{Average price of coffee, \$/lb}
\end{center}
\end{table}
This dramatic increase probably accounts for the increased volume of tea
imports during the same period.
```

In Table 1 we see that the average price doubled in less than three years. This dramatic increase probably accounts for the increased volume of tea imports during the same period.

13.3 Producing a list of tables

Including a **\listoftables** command causes LaTeX to generate a list of all numbered table captions. The command works just like the **\tableofcontents** command, except that LaTeX writes a file `NAME.LOT` when making the list. As usual, you have to run through the LaTeX translation twice to print the list.

14 Arrays

Arrays are the math mode equivalent of tables, consisting of elements arranged in rows and columns. The commands defining the **array** environment are similar to those for the **tabular** environment. *You must be in math mode before you can use the* **array** *environment.*

`\[ \begin{array}{clcr}` (for a four-column array)
(text) `&` (text) `&` (text) `&` (text) `\\`
(text) `&` (text) `&` (text) `&` (text) `\\`
⋮
(text) `&` (text) `&` (text) `&` (text)
`end{array} \]`

- In this example we entered the **displaymath** environment first (`\[`), then entered the **array** environment. At the end, we exited from the **array** environment, then from the **displaymath** environment.
- The argument *clcr* in the `\begin{array}{clcr}` command specifies that elements in the four array columns should be centered, flush left, centered, and flush right, respectively.
- Don't put `\\` after the last row. (This is a difference from the **tabular** environment.)
- LaTeX is in math mode in the **array** environment, and so ignores spaces within array elements.
- The array would normally be printed flush left. If you want to center it, enclose the **array** environment in a **center** environment.

For example, typing

```
\begin{center}
\[ \begin{array}{lcr}
a & p+q & \alpha + \beta + \gamma \\
b+c & r+s+t & \delta + \epsilon \\
d+e+f & u & \zeta
\end{array} \]
\end{center}
```

produces

$$
\begin{array}{lcr}
a & p+q & \alpha+\beta+\gamma \\
b+c & r+s+t & \delta+\epsilon \\
d+e+f & u & \zeta
\end{array}
$$

You can put large delimiters around arrays (e.g. parentheses or brackets to make matrices, or vertical lines to make determinants) by enclosing the entire **array** environment in a **displaymath** environment (`\[...\]`) and using the **\left** and **\right** commands. (See p. 39.) For example, typing

```
\[ \left[ \begin{array}{cc}
a_{11} & a_{12} \\
a_{21} & a_{22} \\
a_{31} & a_{32}
\end{array} \right\ \]
```

produces

$$
\left[\begin{array}{cc}
a_{11} & a_{12} \\
a_{21} & a_{22} \\
a_{31} & a_{32}
\end{array} \right]
$$

15 Odds and Ends

If you have absorbed the contents of this document so far, you should be able to produce almost anything you might ever want in a document. LaTeX has many other features that we have not discussed, however, some of which will certainly be useful to you at some time or other. In this section, we will briefly indicate what these features are, discuss some of them, and state where in the LaTeX manual you can get information about them.

15.1 Title page and abstract

If you provide LaTeX with a title, author, and date of a document with the commands **\title{}**, **\author{}**, and **\date{}**, thereafter the command **\maketitle** will cause LaTeX to print this information. In the **article** document style the title will be contained on a text page; in other styles it goes on a separate page. The LaTeX manual discusses this command on p. 22 and pp. 163-164.

You can also custom design a title page in the **article** document style by enclosing the desired text in a **titlepage** environment. For example, the title page of this document was obtained with the commands:

```
\begin{titlepage}
\begin{center}

\vspace{2.5in}
{\LARGE\bf WORD PROCESSING USING \LaTeX\ ON THE NCSU-CHE VAX}

\bigskip
{\bf R.M. Felder and D. Riddle\\Department of Chemical
Engineering\\North Carolina State University\\Raleigh, NC 27695

\today\ }
\end{center}
\end{titlepage}
```

The **titlepage** environment produces a page with the **empty** page style (no page number), and resets the number of the following page to one.

You can produce an abstract with the set of commands

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

LaTeX will then print **Abstract** centered on the page, followed by your text in ordinary paragraph style. The abstract will be placed on the first text page in the **article** document style, and on a separate page in the **report** document style and when the **titlepage** environment is used with the **article** style.

15.2 Footnotes

The command

`\footnote[`*num*`]{`*text*`}`

produces a footnote with *text* as its text and *num* as its number. Don't leave a space between this command and the text that comes before it. If you omit the *num* argument, LaTeX uses the number of an internal counter which is increased by 1 each time this command is used. For example, typing

```
Jones\footnote{See reference on p. 384} discusses this point.
```

produces the text

Jones[2] discusses this point.

and the footnote at the bottom of this page. It is Number 2 because this is the second footnote in the text.

You can use any symbol to mark a footnote rather than using the number provided by the **\footnote** command. To do so, type the desired symbol as a superscript in the text (`$^{`*symbol*`}$`) and immediately follow it with the command

`\footnote[0]{$^{`*symbol*`}$`*text of footnote*`}`

For example, typing

```
Effective epigrams may be stated simply$^{*}$
\footnote[0]{$^{*}$Law of Probable Dispersal: Whatever hits the fan will
not be evenly distributed.}
or poetically.$^{\dag}$
```

[2] See reference on p. 384

```
\footnote[0]{$^{\dag}$The intellect of man is forced to choose, perfection
of the life, or of the work. (W.B. Yeats)}
```

produces the following text and the two associated footnotes.

Effective epigrams may be stated simply*[0] or poetically.†[0]

Information about spacing of footnotes and when the footnote command cannot be used is given on pp. 156-157 of the LaTeX manual.

15.3 Defining commands, environments, and document styles

You can define your own LaTeX command with

`\newcommand{\`*yourcommand*`}{`*LaTeX command or text*`}`

You would use this command to provide yourself with a shorthand version of a relatively lengthy sequence of characters that occur frequently in the text you are typing.

You can change the definition of an existing command with

`\renewcommand{\`*LaTeX command*`}{`*replacement command*`}`

The LaTeX manual discusses these commands on pp. 55-57 and 173.

You can also define a new environment with the **newenvironment** command, discussed on p. 57 of the manual, and you can create your own document style to do letters, memoranda, or anything else that you do often by following procedures discussed on pp. 85-87 of the manual.

15.4 Boxes

A **box** is a chunk of text that LaTeX treats as a unit—it never splits it to break a line or a page. A number of different types of boxes can be defined in LaTeX—boxes containing single lines of text, with or without frames around them; boxes containing multiple lines; and solid rectangular blocks of arbitrary dimension.

Some of the more commonly used boxes are the following:

`\mbox{`*text*`}`

[0]*Law of Probable Dispersal: Whatever hits the fan will not be evenly distributed.
[0]†The intellect of man is forced to choose, perfection of the life, or of the work. (W.B. Yeats)

The text is enclosed in a box on a single line just wide enough to contain it.

`\fbox`{*text*}

As above, only a frame is drawn about the box.

Typing `y = \fbox{25 meters}` produces y = $\boxed{\text{25 meters}}$

`\makebox`[*size*] [*pos*] {*text*}
`\framebox`[size] [pos] {text}

These are similar to `\mbox` and `\fbox` except that they allow you to specify the width of the box (first argument) and the position of the text in the box (second argument — **l** for left side, **r** for right side). If the second argument is omitted, the text is automatically centered in the box. If the box is too narrow, the text will spill over its borders.

`\parbox`[*pos*] {*size*} {*text*}

The text is put in the box of width *size*, taking up as many lines as necessary. (The multiple–line feature is what distinguishes the parbox from the previous boxes.) There is no indentation, regardless of the `\parindent` setting in the document.

The first optional argument (*pos*) may be a **t** to align the top line of the parbox with the text line, or a **b** to align the bottom line of the parbox with the text line. If this argument is omitted, the center of the box will be aligned with the text line.

`\raisebox`{*size*}{*text*}

This command raises (or lowers) the text by the indicated amount (*size*).

The manual provides more information about boxes on pp. 96-101 and pp. 194-199.

15.5 Pictures and numbered figures

The **picture** environment, discussed on pp. 101-111 of the LaTeX manual, is used to draw pictures composed of text, straight lines, arrows, and circles. It is generally necessary to sketch the figure you want on graph paper first, and then give the coordinates of each component of the figure to LaTeX.

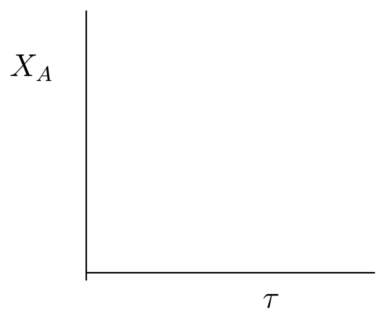
Pictures (or anything else) can be enclosed in the **figure** environment and given captions. The captions may then be printed in a LaTeX-generated list of figures, and cross-references to figure numbers may be included in the text. (These provisions are exactly equivalent to those of the **table** environment.) LaTeX discusses the necessary procedures on pp. 59-60.

There are ways to generate simple graphical figures without going into the picture environment. Check the examples below and see how they were done.

Typing

```
\begin{center}
\begin{tabular}{lll}
&\vline &\\
$X_{A}$&\vline &\\
&\vline &\\
&\vline &\\
&\vline &\\
&\vline &\\
&\vline &$\!\!\!\!\!\!\underline{\hspace{1.5in}}$ \\
&&\hspace*{.75in}$\tau $
\end{tabular}
\end{center}
```

produces



Typing:

```
\begin{Large}
\begin{bf}
\begin{center}
$\heartsuit \heartsuit \heartsuit \heartsuit$ \\
M E M O \\
FROM    A\\
F R I E N D
\end{center}
\end{bf}
\end{Large}
\begin{center}
\bf $\bowtie $\\
$\approx \approx \approx \approx \approx$\\
\bf $\odot \hspace{1em} \odot $\\
\bf $\bigcirc $ \\
\bf $\smile $
\end{center}
```

produces

$\heartsuit\heartsuit\heartsuit\heartsuit$

M E M O

FROM A

F R I E N D

$\bowtie$

$\approx\approx\approx\approx\approx$

$\odot \hspace{1em} \odot$

$\bigcirc$

$\smile$

15.6 Bibliography and citation

LaTeX has several ways to generate bibliographies. Information can be found on pp. 72-75 and Appendix B of the LaTeX manual.

16 Troubleshooting errors

16.1 How LaTeX handles errors

When you run LaTeX on your input file for the first time, the chances are you will have made some errors. When you translate the file (`$ LATEX NAME`) LaTeX will print a number of messages followed by a question mark, and it will then stop. Most of the messages will be utterly indecipherable; sometimes, but not always, one or two of them may provide information that will help you figure out what the error is. Unless it is obvious, the fastest way to figure out what went wrong is to ask someone with a lot of LaTeX experience. If you can't find such an individual or you've already worn out your welcome on that day, consult the guide to error messages on pp. 120-130 of the LaTeX manual.

For example, suppose you type `\begin{enumrate}` instead of `\begin{enumerate}` somewhere in your text file. When you run LaTeX, the following text will appear on the screen:

```
LaTex error.  See LaTex manual for explanation.
              Type H <return> for immediate help.
!Environment enumrate undefined.
\@latexerr ...for immediate help.}\errmessage {#1}
                                                   \endgroup
1.21  \begin{enumrate}

?
```

The exclamation point on Line 3 is an *error indicator*, and is followed by the error message. In this case, it is not hard to figure out what went wrong—`enumerate` was misspelled.

The next two lines describe what is happening in the low-level typesetting commands in the neighborhood of the error. Until you know much more about LaTeX than you are likely to know for a while, ignore these messages.

Line 6 contains the *error locator*—`1.21 \begin...` It tells you that the error occured on or near Line 21 of the text, which occurs on the first printed page. (If it had been on page 4, for example, the numbers `[1],[2],[3]` would have appeared in the printout.)

The question mark **?** prompt on Line 7 indicates that LaTeX has stopped and is waiting for you to type something. If you wish to get out of LaTeX to correct the error, type **X**. If you want to go on processing the file regardless of the error, hit the return key. To get more information about the error, type **H**. The system will then give you a brief message about the error.

In our example, the system help message would be as follows:

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

If you want to correct the error at this point simply type I`\begin{enumerate}` then press return. *Caution, however*: when you change commands in this way, your document file will not be corrected permanently. Sooner or later, you will have to make the error correction in your input file.

If the given error were made in a real text file, a number of other errors would be picked up by LaTeX in consequence. Since as far as LaTeX is concerned there is no `\begin{enumerate}` command, any `\item` commands will look wrong, as will an `\end{enumerate}` command. Your best bet is to get out of LaTeX, correct the error, and run again rather than trying to continue.

If LaTeX tells you that the error occurred on Line 384 of your text file and you wish to edit the file to correct it, leave LaTeX (type X in response to the question mark prompt), edit your file, go into line editing mode (hit Control Z), type 384 and carriage return, and then return to character editing mode. You will then be at Line 384, and can make the required correction.

If you wish to go through the complete file instead of sitting at the terminal encountering errors one at a time, you can let LaTeX run unattended and see all the error messages later by printing out the NAME.LIS file. Simply type the command `\batchmode` at the very beginning of the input file. Be prepared for some long .LIS files if you do this, though.

16.2 Overfull and underfull warnings

Even if you don't make mistakes, you are likely to get **overfull** or **underfull** messages from LaTeX, followed by either **hbox** or **vbox**. These messages mean that LaTeX couldn't get a line flush with the right margin or it had to impose a line break in an inconvenient place to do so (**hbox**) or it couldn't get the text on a page flush with the bottom (**vbox**). As a rule, you will be satisfied with the way the text was printed and so may cheerfully ignore these warnings. If you're unhappy, you can generally find a way to fix the text so as to avoid the bad line or page break.

16.3 Some popular error messages and what they mean

On pp. 118-130 of the LaTeX manual there are a number of suggestions about what might be wrong when a particular error message is received. In this section we give you some messages and their interpretations to add to the list.

16.3.1 *

Normally an asterisk with no explanation means you forgot the `\end{document}` command at the end of your text file. Exit from LaTeX by holding down the CTRL key and typing C, and then correct the file.

16.3.2 ! Missing $ inserted

This could mean any of several things:

(a) You forgot either the beginning or end command of a math mode environment. Check to make sure your `$`'s and `\[`'s are paired.

(b) You tried to print something that can only be printed in math mode without first going into math mode. Popular examples are Greek letters, subscripts and superscripts, and arrays generated with the **array** environment. Go back and put in math mode delimiters.

(c) You tried to print a dollar sign without preceding it with a backslash (`\$`). Put one in.

(d) You put a blank line in an **equation** or **displaymath** or **eqnarray** environment. Go back and get rid of it.

16.3.3 ! Missing { (or }) inserted

You probably omitted a left or right brace, which is easy to do when you have lots of nested commands and environments that use braces.

16.3.4 ! Missing number, treated as zero.

It may be that you omitted a number where LaTeX expected one. If this does not appear to be the case, the most likely explanation is that you have followed one of the commands given below with either a square bracket, `[`, or an asterisk, `*`, neither of which is allowed.

```
\\  \item  \linebreak  \nolinebreak  \newcounter  \twocolumn
\pagebreak  \nopagebreak  \newtheorem
```

If this is indeed what happened, enclosing the square bracket or asterisk in curly braces, `{[}`, will eliminate the problem.

16.3.5 ! Missing begin{document}

You scratch your head on this one and say, "But I *put* a `\begin{document}` in." What has probably happened is that somewhere in the preamble, more than likely in the first line, an unacceptable character or command has been typed.

16.3.6 ! Undefined control sequence

You have mistyped a command or command sequence. A common occurrence of this error is failing to leave a blank space following a command like `\bf` or `\alpha`.

16.3.7 ! Extra alignment tab has been changed

Either you put more `&` (tabular) or `\>` (tabbing) characters on a line than you were supposed to, or you omitted the final `\\` on the previous line.

16.4 When all else fails

Terrible things happen whenever you forget to close an environment, e.g. a math environment. LaTeX will often go charging along, misinterpreting everything it finds, until it encounters something it can't do. Then it will stop with an error message and the location of what it thinks is the error. Unfortunately, the real error may be a long way back, and very difficult to detect.

A particularly gruesome problem occurs when you forget to close a `verbatim` environment. What happens then is that LaTeX assumes all the text and commands that follow are also to be printed verbatim, which it is happy to do; it doesn't know anything is wrong until it gets to another occurrence of the missing delimiter. At that point, almost any error is likely to occur, and you can pretty much abandon hope of making any sense out of the error messages.

If you get messages that make no sense, see if you can find an environment that you forgot to close. If you can't, then there's only one thing left to do. Start creating temporary .TEX files containing little pieces of your text, run them, and pin down the place where the problem first occurs. As you focus on smaller and smaller pieces of text, you will eventually identify precisely where the problem is. Usually you will then see what's wrong, slap your head, say something unprintable (Hint—the first word is "Oh"), and correct the file. If you still can't find the mistake, your local expert will probably be able to do so without much trouble.

17 User-defined keystrokes and commands

Once you do much word processing with LaTeX, you will find yourself typing certain sequences of characters over and over again: `\begin{enumerate}`, for example, or `\verb`.

You can use **\newcommand** (discussed in the previous section) to define a shorthand version of some character sequences within a specific file. Even better, for sequences that occur in every file you type—usually LaTeX commands—you can define otherwise unused keys on your terminal keyboard to generate these commands with one, two, or three keystrokes. To do this, you must create an EDTINI.EDT file in your directory, if you do not already have one, and in it define the key functions. You must also include the statement

```
edt :== edit/command=[yourdirectory]edtini.edt
```

in your login command file (LOGIN.COM), so that each time you enter the editor ($ EDT NAME.TEX) you will pass through the file where the keys are defined.

In what follows we presume that you are working at a VT-220 or VT-240 terminal. Here is a representative EDTINI.EDT file.

```
set wrap 75
set mode change
DEFINE KEY FUNCTION 18 AS "I{\bf ."
DEFINE KEY GOLD FUNCTION 18 AS "I{\sf ."
DEFINE KEY FUNCTION 19 AS "I{\em ."
DEFINE KEY GOLD FUNCTION 19 AS "I\mbox{ ."
DEFINE KEY FUNCTION 20 AS "I\bigskip ."
DEFINE KEY GOLD FUNCTION 20 AS "I\single ."
DEFINE KEY FUNCTION 21 AS "I\item ."
DEFINE KEY GOLD FUNCTION 21 AS "I\double ."
DEFINE KEY FUNCTION 23 AS "I\hspace*{."
DEFINE KEY GOLD FUNCTION 23 AS 'I\verb".'
DEFINE KEY FUNCTION 24 AS "I\vspace*{."
DEFINE KEY GOLD FUNCTION 24 AS "I\underline{."
DEFINE KEY FUNCTION 25 AS "I\begin{."
DEFINE KEY GOLD FUNCTION 25 AS "I\noindent ."
DEFINE KEY FUNCTION 26 AS "I\end{."
DEFINE KEY GOLD FUNCTION 26 AS "Iquote}."
DEFINE KEY FUNCTION 28 AS "Idocument}."
DEFINE KEY GOLD FUNCTION 28 AS "Itabbing}."
DEFINE KEY FUNCTION 29 AS "I\rule{0mm}{7mm}."
DEFINE KEY GOLD FUNCTION 29 AS "Itabular}."
DEFINE KEY FUNCTION 31 AS "Ienumerate}."
```

```
DEFINE KEY GOLD FUNCTION 31 AS "Iequation}."
DEFINE KEY FUNCTION 32 AS "Iitemize}."
DEFINE KEY GOLD FUNCTION 32 AS "Idisplaymath}."
DEFINE KEY FUNCTION 33 AS "Idescription}."
DEFINE KEY GOLD FUNCTION 33 AS "Ieqnarray}."
DEFINE KEY FUNCTION 34 AS "Icenter}."
DEFINE KEY GOLD FUNCTION 34 AS "Iarray}."
```

The first command enables word wrapping during editing, and the second puts you in character editing mode as soon as you enter the editor. (This enables you to avoid having to type *C.)

The next commands define the keys at the top of your terminal keyboard, starting with F7 (key function 18) and continuing through F20 (key function 34). Thus, for example, when you hit Key F7, the sequence `{\bf` will be typed in the file (and on the screen); when you hit GOLD(PF1) and then F20, the sequence `array}` will be typed.

The keys and substituted commands corresponding to this EDTINI.EDT file are as follows:

KEY	COMMAND	GOLD COMMAND
F7	`{\bf`	`{\sf`
F8	`{\em`	`\mbox{`
F9	`{\bigskip`	`{\single`
F10	`\item`	`{\double`
F11	`hspace*{`	`\verb`
F12	`vspace*{`	`\underline{`
F13	`\begin{`	`\noindent`
F14	`\end{`	`quote}`
HELP	`document}`	`tabbing}`
DO	`\rule{0mm}{7mm}`	`tabular}`
F17	`enumerate}`	`equation}`
F18	`itemize}`	`displaymath}`
F19	`description}`	`eqnarray}`
F20	`center}`	`array}`

Thus, for example, you can produce `\begin{itemize}` simply by hitting the F13 and F18 keys in succession.