

# Text Formatting with L<sup>A</sup>T<sub>E</sub>X

## *A Tutorial*

### Table of Contents

<b>1. L<sup>A</sup>T<sub>E</sub>X Basics</b>	<b>1</b>
1.1 What is T <sub>E</sub> X?	1
1.2 What is L <sup>A</sup> T <sub>E</sub> X?	1
1.3 How L <sup>A</sup> T <sub>E</sub> X Works	1
1.4 The L <sup>A</sup> T <sub>E</sub> X Input File	2
1.4.1 Entering L <sup>A</sup> T <sub>E</sub> X Commands	2
1.4.2 Entering Text	3
1.4.3 Special Characters	3
1.4.4 Structure of the Input File	4
1.5 Some L <sup>A</sup> T <sub>E</sub> X Vocabulary	5
<b>2. Creating A L<sup>A</sup>T<sub>E</sub>X Document</b>	<b>6</b>
2.1 Document Classes	6
2.2 Class Options	6
2.3 Packages	7
2.4 Making a Title Page	8
2.5 Making a Table of Contents	8
2.6 Behind the Scenes	9
2.6.1 Auxiliary Files	9
2.6.2 How a Page is Built	9
2.7 Example: Report Class	10
2.8 Example: Letter Class	11
<b>3. Document Layout</b>	<b>12</b>
3.1 Line Spacing	12
3.2 Paragraphs	12
3.3 Text Justification	13
3.4 Margins	13
3.5 Headers, Footers, and Page Numbering	14

Note: This memo reflects the use of L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>.

<b>4. Within the Text</b>	<b>15</b>
4.1 Section Headings	15
4.2 Changing Type Style and Size	16
4.3 Starting New Lines and New Pages	17
4.4 Leaving Horizontal and Vertical Space	17
4.5 Drawing Rules	17
4.6 Footnotes	18
4.7 Centering	18
4.8 Quotations	18
4.9 Reproducing Text As-Is	19
4.10 Lists	20
4.11 Cross References	21
4.12 Placing Figures and Tables on the Page	21
4.12.1 Making a Caption	22
4.12.2 Example	22
4.12.3 Overcoming Problems with Float Placement	23
<b>5. Tables</b>	<b>24</b>
5.1 Tabbing	24
5.2 Tabular	25
5.2.1 A Simple Ruled Table	26
5.2.2 Using Paragraph Columns, Spanning Columns	27
5.2.3 Aligning on the Decimal Point	28
5.2.4 Suppressing Leading or Trailing Space	28
<b>6. Mathematics</b>	<b>29</b>
6.1 In-line Math	29
6.2 Display Math (for unnumbered equations)	30
6.3 Equation Environment (for numbered equations)	30
6.4 Eqnarray Environment (for multiline equations)	31
6.5 Array Environment (for matrices, etc.)	32
6.6 Building Mathematical Expressions	33
6.6.1 Superscripts and Subscripts	33
6.6.2 Spaces in Math Mode	33
6.6.3 Dots, Braces, and Bars	33
6.6.4 Fractions	34
6.6.5 Radicals, Integrals, and Summations	34
6.6.6 Large Delimiters	35
<b>7. Special Topics</b>	<b>36</b>
7.1 Managing a Large Document	36
7.2 Preparing a Bibliography	36
7.3 Generating an Index	38
7.4 Including and Manipulating PostScript Graphics	39
7.5 Accents and Special Characters	41
<b>Appendix A: Mathematical Symbols</b>	<b>42</b>
<b>Appendix B: Error Messages</b>	<b>45</b>

# Text Formatting with L<sup>A</sup>T<sub>E</sub>X

## Chapter 1. L<sup>A</sup>T<sub>E</sub>X Basics

### 1.1 What is T<sub>E</sub>X?

- T<sub>E</sub>X is the typesetting language, designed especially for math and science, upon which L<sup>A</sup>T<sub>E</sub>X is built. T<sub>E</sub>X is pronounced “Tech,” similar to “Bach.”
- T<sub>E</sub>X is portable. It is available for most computers and is used all over the world. T<sub>E</sub>X documents can be moved easily from one system to another, as long as the required fonts are on both systems.
- T<sub>E</sub>X comes with its own set of fonts, called “Computer Modern.” These fonts exist in a variety of styles, including serif, sans serif, and typewriter (fixed pitch).
- T<sub>E</sub>X is also a programming language, making it possible to create commands that simplify its use.

### 1.2 What is L<sup>A</sup>T<sub>E</sub>X?

- L<sup>A</sup>T<sub>E</sub>X is a T<sub>E</sub>X macro package, originally written by Leslie Lamport, that simplifies the use of T<sub>E</sub>X. All the above features of T<sub>E</sub>X, including portability, also apply to L<sup>A</sup>T<sub>E</sub>X. L<sup>A</sup>T<sub>E</sub>X is pronounced either “Lay-tech” or “Lah-tech.”
- Most L<sup>A</sup>T<sub>E</sub>X commands are “high-level” (such as `chapter` and `section`) and specify the logical structure of a document. The author rarely needs to be concerned with the details of document layout. Most plain T<sub>E</sub>X commands also work with L<sup>A</sup>T<sub>E</sub>X.
- The *document class* determines how the document will be formatted. L<sup>A</sup>T<sub>E</sub>X provides several standard document classes from which to choose.

### 1.3 How L<sup>A</sup>T<sub>E</sub>X Works<sup>1</sup>

To use L<sup>A</sup>T<sub>E</sub>X, you first create a plain ASCII text file with any text editor. In this file you type both the text of your document and the L<sup>A</sup>T<sub>E</sub>X commands to format it. You then run the L<sup>A</sup>T<sub>E</sub>X program to process the file, and L<sup>A</sup>T<sub>E</sub>X produces a new file called a DVI (“device independent”) file that is in binary format (not readable to humans). A separate program called a “device driver” converts the DVI file to a format acceptable by the printer you are using. Device drivers for the screen are called previewers.

---

<sup>1</sup>This memo describes the L<sup>A</sup>T<sub>E</sub>X language. To learn how to run, preview, and print on Rensselaer’s UNIX systems, see Quick Study#20, *Using T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X on RCS*.

## 1.4 The L<sup>A</sup>T<sub>E</sub>X Input File

L<sup>A</sup>T<sub>E</sub>X input files normally have names that end with the extension `.tex`: for example, an acceptable filename might be `myfile.tex`. The input file contains both the text of your document and the L<sup>A</sup>T<sub>E</sub>X commands needed to format it.

### 1.4.1 Entering L<sup>A</sup>T<sub>E</sub>X Commands

To distinguish them from text, all L<sup>A</sup>T<sub>E</sub>X commands (also called *control sequences*) start with a backslash “\”. There are two kinds of control sequences:

- **control words** are made up completely of letters. They must end with either a non-letter or a space so L<sup>A</sup>T<sub>E</sub>X knows where the control word ends. A space following a control word is “consumed” by L<sup>A</sup>T<sub>E</sub>X and does not appear in the output. An example is the control word `\newpage`, which forces a new page.
- **control symbols** consist of the backslash followed by exactly one non-letter. They are most often used to put a special symbol in the text. For example `\$` prints a \$ (which cannot be entered directly because L<sup>A</sup>T<sub>E</sub>X uses it to begin math mode). Control symbols do not have to be followed by a space.

L<sup>A</sup>T<sub>E</sub>X commands are case-sensitive. Most are all lowercase. A few commands use the first letter in uppercase such as `\Delta` → Δ. Fewer still use all uppercase.

Some commands take an “argument,” placed within curly braces { } after the command name. For example, `\textbf{this text is bold}` prints the text inside the braces in boldface type: **this text is bold**

L<sup>A</sup>T<sub>E</sub>X uses *grouping* to limit the effect of certain commands. Braces ({ and }) are used to begin and end groups. For example, the `\large` command is usually used inside a group: `{\large this is bigger than normal}` produces:  
this is bigger than normal

A command such as `\large` is called a “declaration” because, unless it is given within a group, its effect will continue until another declaration (in this case `\normalsize`) counteracts it. Note the difference between a declaration used inside a group and a command like `\textbf{...}`, which will not work unless an argument is specified.

The symbol % can be used to put a comment in your input file. When L<sup>A</sup>T<sub>E</sub>X sees a %, it ignores the rest of the line.

When you use commands that specify a length, such as a command to set the size of a margin or a command to leave a certain amount of space, you will need to specify the units of measurement. L<sup>A</sup>T<sub>E</sub>X recognizes the following units:

<code>cm</code>	centimeter	<code>pt</code>	printer’s point, ≈ 72 per inch
<code>mm</code>	millimeter	<code>em</code>	font-dependent width of “m”
<code>in</code>	inch	<code>ex</code>	font-dependent height of “x”

### 1.4.2 Entering Text

In the input file, words are separated by leaving one blank space. Paragraphs are separated by leaving one blank line. (You can also use the command `\par` to indicate a new paragraph.)  $\LaTeX$  ignores multiple blank spaces and multiple blank lines in input files.

Type single quotation marks by using the left (‘) and right (’) single quote marks on your keyboard. Type left double quotation marks by using two single left quotes (‘ ‘), and type right double quotation marks by using either two single right quotes (’ ’) or the double quote key (").

There are three kinds of dashes in typeset documents: the hyphen (for compound words), the endash (for such things as page number ranges), and the emdash (used as a punctuation mark in English prose). Since there is only one dash on the keyboard, type `-`, `--`, and `---` to get `-`, `–`, and `—`.

To prevent two words from being split at a line break, tie them together with the tilde character: for example `Mr.~Smith` will never appear with “Mr.” at the end of one line and “Smith” at the start of the next.

Note that some characters have special meaning to  $\LaTeX$  and must be entered in a special way, as described in the next section.

### 1.4.3 Special Characters

Certain characters have special meaning to  $\LaTeX$ . An example is the % sign, which indicates that the remainder of the line is a comment and should not be processed. Below is the complete list of special characters. To have these characters print in your output, you must type them in your input file as shown below.

<u>Character</u>	<u>Type in file</u>	<u>Special <math>\LaTeX</math> meaning</u>
#	<code>\#</code>	Parameter in a macro; also used in tables
\$	<code>\\$</code>	Used to begin and end math mode
%	<code>\%</code>	Used for comments in the source file
&	<code>\&amp;</code>	Tab mark, used in alignments
-	<code>\-</code>	Used in math mode for subscripts
^	<code>\^{\}</code>	Used in math mode for superscripts
~	<code>\~{\}</code>	Tie character, used to produce a “hard” space
{	<code>\{\$</code>	Used to begin a group
}	<code>\}\$</code>	Used to end a group
\	<code>\backslash\$</code>	Used to begin a control sequence
<	<code>\&lt;\$</code>	Without the \$ signs, produces $j$
>	<code>\&gt;\$</code>	Without the \$ signs, produces $j$

### 1.4.4 Structure of the Input File

All  $\LaTeX$  input files must conform to a certain structure. They begin with the command `\documentclass`, and all the text of the document must be contained between the commands `\begin{document}` and `\end{document}`.

```
\documentclass[options]{class}
Preamble
\begin{document}
Document text
\end{document}
```

In the first command above, *class* specifies the type of document you intend to create. You can choose from one of the  $\LaTeX$  classes described in the next chapter. If you wish, you can also include one or more *options* to modify the behavior of the document class.

The *preamble* is the section of the file between the `\documentclass{...}` command and the `\begin{document}` command. This is the place to put commands that will influence the style of your entire document and macro definitions that you will use later. You may also load *packages* that add new features to  $\LaTeX$ . Text is not allowed in the preamble.

The `\begin{document}` command indicates the end of the preamble and the beginning of your text. A corresponding `\end{document}` command always ends your files. A really short  $\LaTeX$  input file might look like:

```
\documentclass{article}
\begin{document}
This LaTeX file is short and sweet. It uses the article class,
a good all-purpose layout. There is nothing in the preamble, which
is perfectly acceptable.

This is a new paragraph. \textit{Here's some italic text}
and \textbf{some bold text}.
{\small The text inside these braces is smaller than normal.}
Now the text size is back to normal.
\end{document}
```

Longer examples, one using `report` class and one using `letter` class are included at the end of the next chapter.

## 1.5 Some L<sup>A</sup>T<sub>E</sub>X Vocabulary

**Commands** produce text or space. For example, `\hspace{2in}` and `\vspace{2in}` are commands that create 2 inches of horizontal and vertical space, respectively, and `\textit{some italic words}` puts the contents of its argument in italic type.

**Declarations** produce neither text nor space, but either affect the way L<sup>A</sup>T<sub>E</sub>X prints the following text or provide information for later use. Font size changes are an example of declarations. `\large` will cause any text that follows to appear in a larger type size. Declarations are often used within a group to limit their scope. For example: `{\large Only the text inside these braces is large.}`

**Environments** are blocks of text that receive special processing. An environment is defined by a matching `\begin{environment name} ... \end{environment name}`. For example, text appearing between the commands `\begin{quote}` and `\end{quote}` will be set off in a single-spaced, indented block. Note that a blank line before an environment ends the previous paragraph. A blank line following an environment indicates that the line following the environment is a new paragraph. Nested environments must be ended in the reverse order of entry, i.e., the first started is the last ended.

**Mandatory arguments** supply information required for a command to execute. For example, `\hspace{2in}` needs the information provided by the argument to generate the horizontal space. Mandatory arguments are enclosed in braces: `{ }`.

**Optional arguments** follow a command name and are inclosed in square brackets: `[ ]`. For example, the size of type to be used for your main text is an optional argument in the `{\documentclass}` command. To use the article class in 11-point type, you would type `\documentclass[11pt]{article}`. Without this optional argument, you would get the default 10-point type.

\* indicates a variation on a command or environment. For example, `\` indicates a line break. `\*` indicates a line break with the restriction that L<sup>A</sup>T<sub>E</sub>X is not allowed to begin a new page at that point. Space printed by `\vspace` and `\hspace` commands is normally dropped if it appears at the beginning or end of a line or page. If you want the space printed no matter where it falls, you would use `\hspace*` or `\vspace*`.

## Chapter 2. Creating A L<sup>A</sup>T<sub>E</sub>X Document

### 2.1 Document Classes

The document class determines the overall layout of the document. There are five standard classes distributed with L<sup>A</sup>T<sub>E</sub>X:

**article** for simple or short documents, including journal articles, and short reports. A good all-purpose class.

**report** for small books and longer reports containing chapters.

**book** for books.

**letter** for letters, either business or personal.

**slides** for making transparencies for projection on a screen.

These classes provide preset formats with default margins, paragraph formatting, and special commands suitable for producing specific sections. For example, the **article**, **report**, and **book** classes include a variety of commands to format section headings (`\part`, `\chapter`, `\section`, `\subsection`, etc.), as well as commands to produce a title page and a table of contents. There are minor differences between these three classes. The **book** class, for example, uses a smaller printed page size—about 7.5 × 5 inches—and is formatted for two-sided printing by default. The **article** class is intended for shorter works and does not have chapters (so articles can be easily included in reports or books). The **letter** class provides special commands to produce the salutation, address, and closing.

These four classes are single-spaced by default, and have 10, 11, and 12-point type sizes available as options. 10 points is the default size.

The **slides** class uses sans-serif type fonts much larger than the usual ones and expects the document to be divided up into 1-page sections.

At Rensselaer, there is an additional class on RCS called **thesis**, which may be used to produce either a master's or a doctoral thesis with a format that meets the requirements of the Rensselaer Graduate School. It was written by Academic Computing Services using the **report** class as a base. To use the **thesis** class, begin your document with the line:

```
\documentclass{thesis}
```

Instructions are in ACS Memo RPI.110, *Preparing A Thesis with L<sup>A</sup>T<sub>E</sub>X*.

### 2.2 Class Options

The document class may be modified by *options*, which are placed in square brackets after the `\documentclass` command:

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

Multiple options are separated by commas. The standard class options include:

**10pt**, **11pt**, **12pt** Selects the point size of main font for the document. If no option is specified, 10pt is assumed. This memo uses 12-point type.

**twocolumn** Produces two-column pages.

**titlepage** Causes the `\maketitle` command to generate the title page on a separate page for the `article` class. This option is not necessary for the `book` and `report` classes, as they print separate title pages by default.

**leqno** Puts equation numbers on left side. (They are on the right by default.)

**fleqn** Left-aligns equations. (They are centered by default.)

**twoside** Formats for printing on both sides of paper. (Whether the document is actually printed two-sided depends on the printer.) Twoside is the default for the `book` class, but not for any of the other classes.

**openright** If the `twoside` option is in effect, chapters will begin on right hand pages. This is the default for the `book` class. It does not apply to the `article` class, which does not contain chapters. (The opposite of `openright` is `openany`.)

## 2.3 Packages

There are a large number of  $\text{\LaTeX}$  *packages* available that provide many additional features. Some packages are distributed with  $\text{\LaTeX}$ ; others are provided by expert users worldwide. If you sometimes find that the features of standard  $\text{\LaTeX}$  do not provide the special formatting you want, chances are good that you can find a package to meet your needs. A package generally consists of one or more files that contain extra definitions and macros. Some packages are simple; others are complex and can contain options. The file names usually have the extension `.sty`.

You load a package with the `\usepackage` command, which should come immediately after the `\documentclass` command in your input file. The command has the form:

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

Each package may be included with its own `\usepackage` command, or you may use one command to load several packages by separating their names with commas. For example, if you are inserting PostScript graphics in your document (see section 7.4), the package `graphicx` will make this easier for you. The beginning of your input file might look like:

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

The `\usepackage` command above instructs  $\text{\LaTeX}$  to read the file `graphicx.sty`.

In addition to `graphicx`, there are many other packages available on RCS at Rensselaer including packages to rotate text, use PostScript fonts, and customize such things as headers and footers, citations and captions. To see a current list of packages with brief descriptions, look at the file `package-summary` in the RCS directory `/campus/doc/text/Latex2e/Packages`. Under that directory are various subdirectories containing documentation for the individual packages. For example, documentation for including PostScript graphics is in the directory `/campus/doc/text/Latex2e/Packages/Graphics`.

## 2.4 Making a Title Page

If you are using the `article`, `report`, or `book` class, you may want a title page for your document. To do this, you need to supply text for the title, author, and date, and then tell  $\text{\LaTeX}$  to generate the title page with the command `\maketitle`. In your  $\text{\LaTeX}$  input file, following the `\begin{document}` command, type commands such as the ones below:

```
\title{This is the Title}
\author{My Name}
\date{the date}
\maketitle
```

This is illustrated in the example of using the `report` class later in this chapter. If there are several authors, you can separate their names with `\and`, or you can separate them with `\\` if you would like each name centered on a separate line. If you omit the `\date` command,  $\text{\LaTeX}$  will use the current date. If you want no date at all, use `\date{}`.

In the `report` and `book` classes, the title information appears on its own page. In the `article` class, it appears at the top of the first page of text. You can instruct the `article` class to make a separate page by using the class option `titlepage`.

## 2.5 Making a Table of Contents

The command `\tableofcontents`, usually placed in the input file right after the `\maketitle` command, creates a table of contents using the information in the section headings (`part`, `chapter`, `section`, etc.). Since this information is taken from the previous run, you will need to run  $\text{\LaTeX}$  twice on a new document for the entries in the table of contents to show up.

## 2.6 Behind the Scenes

### 2.6.1 Auxiliary Files

Part of the convenience of  $\LaTeX$  is its ability to do forward references (see section 4.11) and to create a table of contents, a list of tables, and a list of figures.

Forward reference numbers, as well as page numbers for sections, figures and tables, are unknown when  $\LaTeX$  is first processing the input file.  $\LaTeX$  stores this information in auxiliary files as it processes the job. A second run allows  $\LaTeX$  to extract the information from its auxiliary files and complete the table of contents, etc. Therefore *all information used by  $\LaTeX$  for tables of contents, etc. is from the previous run.* The only way to be sure that all this material is correct is to format the file twice after making any changes. Usually for drafts, the difference between runs is not enough to matter, but for final versions you should remember to run  $\LaTeX$  twice before printing.

The auxiliary files have the same “root” name as the  $\LaTeX$  input file, but different extensions. For example, all documents need an AUX file. If the input file is named `myfile.tex`, the AUX file will be named `myfile.aux`. Other auxiliary files (see the list below) are needed only if you are producing a table of contents, etc. Most computer systems automatically create the auxiliary files as they are needed.

<code>filename.aux</code>	always needed
<code>filename.toc</code>	for table of contents
<code>filename.lot</code>	for list of tables
<code>filename.lof</code>	for list of figures

### 2.6.2 How a Page is Built

When  $\TeX$  or  $\LaTeX$  builds a page, it considers all the parts (words, lines, paragraphs, etc.) to be different sized *boxes*. It starts with a simple box, an individual letter, and then builds words, which are considered to be *hboxes* (horizontal boxes). The words are then put together with *glue* to form a line, which is a larger hbox. A group of hboxes stacked together vertically (with glue between them) form a *vbox* (vertical box). A page is a large vbox made up of several smaller ones. You do not normally need to be concerned with this, but sometimes (such as when an error message refers to an “overfull hbox,” meaning a line is too long and sticks out into the margin) it is helpful to know how  $\TeX$  works.

## 2.7 Example: Report Class

```

\documentclass[11pt]{report}           % Report class in 11 points
\raggedright                           % Do not right-justify
\parindent0pt \parskip10pt           % make block paragraphs

\begin{document}                       % End of preamble, start of
                                        % document text.
\title{\bf An Example of Report Class} % Supply information
\author{for \LaTeX\ Class}            % for the title page.
\date{\today}                          % Use current date.
\maketitle                             % Print title page.
\pagenumbering{roman}                  % Roman page number for toc
\setcounter{page}{2}                   % Make it start with "ii"
\tableofcontents                       % Print table of contents

```

```

\chapter{A Main Heading}               % Print a "chapter" heading
\pagenumbering{arabic}                 % Start text with arabic 1

```

Most of this example applies to the article and book classes as well as to the report class. In article class, however, the default position for the title information is at the top of the first text page rather than on a separate page.

A blank line starts a new paragraph. `\textit{Note this: it will be printed in italic type.}`

```

\section{A Subheading}                 % Print a "section" heading
The following sectioning commands are available:\\
part \\                                % \\ Forces a new line
chapter \\
section \\
subsection \\
subsubsection \\
paragraph \\
subparagraph

```

But note that---unlike the book and report classes---the article class does not have a ‘‘chapter’’ command.

```

\end{document}                         % The required last line

```

## 2.8 Example: Letter Class

```

\documentclass[12pt]{letter}           % letter class, 12 points

\address{555 Main St.\\Somtown, NY 12345} % Return address
\signature{My Name\\My Title}         % Name for signature

\begin{document}                       % End of preamble

\begin{letter}{Mr.~Smith\\ President, % Begin letter by giving
  Big Name Co.\\Bigburg, MI 45678}    % recipient's address
\opening{Dear Mr.~Smith:}             % Name for salutation

```

This is the letter class. It provides a format for standard parts of a business letter. As you can see, it uses many commands that do not exist in the article, report, and book classes.

This is a new paragraph.

```

\closing{Sincerely,}                  % Format for the closing.
                                      % The name is taken from
                                      % \signature command above.

\cc{My Boss}                          % Name(s) of those
                                      % receiving copies

\end{letter}                           % End of letter

\end{document}                         % The required last line

```

## Chapter 3. Document Layout

Defaults for all aspects of the document layout are set by the document classes. However, if you want to change the defaults, there are commands that enable you to do so. Commands controlling features that apply to the whole document should be placed in the preamble<sup>2</sup>.

### 3.1 Line Spacing

The default is single spacing. If you want larger interline spacing for your document, you can use the `\linespread` command in the preamble. The following command produces a document with double spacing:

```
\linespread{1.6}
```

For “line and a half” spacing, use the value 1.3. The default spread is 1.

An alternative and more flexible way to control the linespacing is to use the package `setspace`. With this package, footnotes, figures, and tables remain single-spaced. The package also defines a new environment called `singlespace`, which you can use to include single-spaced sections within your document. To use the `setspace` package to produce a double-spaced document, include after the `\documentclass` command:

```
\usepackage{setspace}
```

and, in addition, put the command

```
\doublespacing
```

somewhere in the preamble.

You could use `\onehalfspacing` instead of `\doublespacing`, or you could use the command `\setstretch{n}` (specifying your own value for `n`—usually between 1 and 2) to set the spacing to whatever you want.

### 3.2 Paragraphs

To start a new paragraph, either leave a blank line or use the control sequence `\par`. By default, paragraphs are indented by 1.5em, which means 1.5 times the point size of the document. No extra blank space is inserted between paragraphs. The commands `\parindent` and `\parskip` control paragraph indentation and paragraph separation. To get block paragraphs, for example, include in the preamble the commands:

```
\parindent=0in  
\parskip=10pt
```

---

<sup>2</sup>the section between the `\documentclass` command and the `\begin{document}` command

### 3.3 Text Justification

By default, L<sup>A</sup>T<sub>E</sub>X justifies your text horizontally so that both left and right margins are smooth. If you prefer “ragged right” text, you can use the declaration:

```
\raggedright
```

Note that this has the side-effect of wiping out the paragraph indentation. (It assumes you want everything flush left.) If you want indented paragraphs, you must specifically request it (i.e., `\parindent=1.5em`) *after* the `\raggedright` declaration.

Vertical justification is controlled by using either `\flushbottom` or `\raggedbottom`. `\flushbottom` makes all text pages the same height, adding extra vertical space if necessary. `\raggedbottom` allows the height to vary a bit from page to page. `\flushbottom` is the default for the book class and for the `twoside` option in the article and report classes; otherwise `\raggedbottom` is the default.

### 3.4 Margins

Top and left margins are set in reference to a value of one inch (which means that setting these margins equal to 0 produces one-inch margins). Therefore, setting a top margin of `.5in` will actually make a top margin of 1.5 inches, and setting a top margin of `-.5in` produces a top margin of .5 inches. The opposite margins (bottom and right) are determined indirectly by setting the height and width of the text area.

<u>to set margin</u>	<u>use the command</u>
top margin	<code>\topmargin</code>
bottom margin	<code>\textheight</code>
left margin (for odd pages or single sided)	<code>\oddsidemargin</code>
left margin (for even pages, if using <code>twoside</code> )	<code>\evensidemargin</code>
right margin	<code>\textwidth</code>

The command names controlling the margins are called *lengths*, and the “official” way to set lengths in L<sup>A</sup>T<sub>E</sub>X is with a command such as: `\setlength{\oddsidemargin}{.5in}`. However, it is easier to borrow from plain T<sub>E</sub>X and use commands like those below:

```
\topmargin=-.5in      % topmargin is 1/2 inch (note negative value)
\oddsidemargin=0in   % left margin is 1 inch on right-hand pages
\evensidemargin=0in  % same for left-hand pages in a 2-sided document
\textwidth=6.5in     % leaves 1 inch for the right margin
\textheight=9in      % 9 inches reserved for the text
```

L<sup>A</sup>T<sub>E</sub>X also leaves .5 inch at the top of the page for a header and about .6 inch at the bottom of the page for a footer. You must take this into account when choosing a value for `textheight`. The values in the examples above leave one inch between the paper edge and the text on all four sides. (The spaces for the header and footer are controlled by the lengths `headheight`, `headsep`, and `footskip`. You can change these values too if you wish, but it is not usually necessary.) For a quick and easy way to set one-inch margins all around, you can use the package `fullpage`.

### 3.5 Headers, Footers, and Page Numbering

The output page consists of the *head*, the *body* and the *foot*. Header and footer material, such as page numbers and/or section titles, appear in the head or the foot. All the classes (except letter) print at least the page number by default.

If you don't like the default action of the document class, you can determine what information goes into the head and foot by using the `pagestyle` command. This command is often placed just after a `\chapter` or a similar command. There are four standard page styles:

`\pagestyle{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 classes.

`\pagestyle{empty}`: The head and foot are both empty.

`\pagestyle{headings}`: The page number and current section heading (the level of the heading is determined by the document class) is put in the head; the foot is empty. This is the default for the book class.

`\pagestyle{myheadings}`: Similar to the headings page style, except you specify the information (other than the page number) that goes in the head by using the `markboth` and `markright` commands. `markboth` is used for two-sided documents, and `markright` is used for one-sided:

```
\markboth{lefthead}{righthead}
\markright{righthead}
```

`\thispagestyle{style}`: Changes the page style *for the current page only*. For example, to have nothing in the head and foot for the current page without affecting the style for the rest of the pages, use `\thispagestyle{empty}`.

You can also specify arabic (the default) or roman page numbering either in the preamble or in the text. It is common to put `\pagenumbering{roman}` before the text begins and `\pagenumbering{arabic}` after the first `\chapter` command. These commands also set the page number to 1. You can change the page number counter yourself with a command such as `\setcounter{page}{2}`.

If the above `pagestyle` commands don't do what you want, there is a package called `fancyhdr` that allows you to customize your headers and footers in an easy way. With this package you can define three-part headers and footers (left, right, and center), multi-line headers and footers, separate headers and footers for even and odd pages, and more. To use it, include the following commands in the preamble:

```
\usepackage{fancyhdr}
\pagestyle{fancy}
```

For simple use, you need only to include the following 6 commands in your preamble, supplying your text inside the `{}` in each case: `\lhead{}`, `\chead{}`, `\rhead{}`, `\lfoot{}`, `\cfoot{}`, `\rfoot{}`. For more information, see the documentation for `fancyhdr` in the RCS directory `/campus/doc/text/Latex2e/Packages/Contrib`.

## Chapter 4. Within the Text

Within the text, there will always be certain sections that require special treatment—such as a different size of type, indentation, or special placement on the page. Some specialized areas of text (particularly those that require indentation) are formatted with the help of  $\LaTeX$  *environments*.

### 4.1 Section Headings

Since documents of any length are usually divided into sections, the classes `article`, `report`, and `book` have a set of commands which take the name of the section as an argument. The author uses these commands in the proper order, providing the section name, and  $\LaTeX$  takes care of formatting the headings (boldface, larger typesize, etc.) and numbering them appropriately. The sectioning commands are:

<code>\part</code>	<code>\section</code>	<code>\paragraph</code>
<code>\chapter</code>	<code>\subsection</code>	<code>\subparagraph</code>
	<code>\subsubsection</code>	

The first two, `\part` and `\chapter`, are not available in `article` class. The `\part` heading is rarely used. It divides a very large document into parts, and does not affect the numbering used for the other headings. In most document classes, headings made with the lowest level headings, `\paragraph` and `\subparagraph`, are not numbered.

If you include the command `\tableofcontents` at the beginning of your document,  $\LaTeX$  takes the section headings and page numbers from the previous run of the document and inserts a table of contents at the place the command was issued. (Note that you need to process the document through  $\LaTeX$  twice to ensure an up-to-date table of contents.)

Normally section headings appear in the table of contents exactly as they do in the text. However, if a heading is too long to fit nicely into the table of contents, you can provide a shorter version as an optional argument:

```
\section[A short heading for the TOC]{This is a much longer
    heading that will appear in the text of the document}
```

You can make less formal headings by using the sectioning commands with a star (\*) appended to the command names listed above. In this case, the section headings will not show up in the table of contents and will not be numbered. For example, to make a section heading called “Introduction” that is not numbered and does not appear in the table of contents, use the command:

```
\section*{Introduction}
```

## 4.2 Changing Type Style and Size

Sometimes you may want to change the style or size of text that is not a section heading. The following L<sup>A</sup>T<sub>E</sub>X commands change the style of the text you supply as an argument:

<code>\textit{...}</code>	<i>italic</i>	Italic shape, used mostly for emphasis
<code>\textsl{...}</code>	<i>slanted</i>	Slanted shape, a bit different from italic
<code>\textsc{...}</code>	SMALL CAPS	Small caps shape, use sparingly
<code>\textup{...}</code>	upright	Upright shape, usually the default
<code>\textbf{...}</code>	<b>boldface</b>	Boldface series, often used for headings
<code>\textmd{...}</code>	medium	Medium series, usually the default
<code>\textrm{...}</code>	roman	Roman family, usually the default
<code>\textsf{...}</code>	sans serif	Sans Serif family, used for posters, etc.
<code>\texttt{...}</code>	typewriter	Typewriter family, fixed-pitch characters
<code>\emph{...}</code>	<i>emphasized</i>	Use for emphasis, usually changes to italic

These commands can be combined, provided the font thus requested actually exists. For example, the command `\textbf{\textit{This is bold italic}}` produces: ***This is bold italic***.

The following commands cause subsequent output to be printed in a different type size. The actual size produced by each command depends on the initial point size selected for the document. The default document size is 10 points. Therefore `\normalsize` means 10 points for a document in which no size option has been included. This memo is printed in 12 points, so in this case, `\normalsize` is 12 points. (Note that when `\normalsize` is 12 points, there is no difference between `\huge` and `\Huge`. They are both the largest size—25 points.)

<code>\normalsize</code>	normal size	<code>\large</code>	large
<code>\small</code>	small	<code>\Large</code>	larger
<code>\footnotesize</code>	smaller than small	<code>\LARGE</code>	larger still
<code>\scriptsize</code>	smaller still	<code>\huge</code>	huge
<code>\tiny</code>	tiny	<code>\Huge</code>	hugest

The size-changing commands are declarations, and therefore they are usually used within a group (i.e., braces) to delimit the range of their action. For example:

`{\small This type will be small}` produces This type will be small.

To change both type size and style at the same time, commands can be used together. For example, the command `\textbf{\large This will be big and bold}` produces: **This will be big and bold**.

### 4.3 Starting New Lines and New Pages

Normally L<sup>A</sup>T<sub>E</sub>X decides where to start a new line and a new page, always trying to pick the most aesthetically pleasing break points. But sometimes you want to force the start of a new line or page. The command `\` will force a new line. For example,

```
This will be on one line\\ this will be on the next line
```

If you want extra space between two lines, do not use two `\` commands in a row. Instead use an *optional* parameter (given inside square brackets) to specify the amount of blank space. For example, the following command will leave an extra space of 10 points between the lines:

```
This will be on one line\\[10pt] this will be on the next line
```

To force a new page, the simplest command is `\newpage`, which starts a new page immediately. There is also the command `\clearpage`, which acts like `\newpage` except that it also forces any leftover figures or tables to print before starting the new page. With the `twoside` style option, the command `\cleardoublepage` produces a blank page, if necessary, to ensure that the new page starts on a new sheet of paper.

### 4.4 Leaving Horizontal and Vertical Space

The commands `\hspace` and `\vspace` leave horizontal and vertical space in your text. Both commands take a mandatory parameter—the amount of blank space you want to leave. For example, `\vspace{3in}` will leave 3 inches of blank space in your text. (If vertical space is requested in the middle of a paragraph, the space will appear after the current line has ended.)

Space requested by the `\hspace` and `\vspace` commands disappears if it falls at the beginning or end of a line or page. To create space that remains no matter where it falls, use the variations `\hspace*` and `\vspace*`. For example, the following lines:

```
This text starts at the left margin\\
\hspace*{1in}This text starts a new line after a one-inch space
```

produces:

```
This text starts at the left margin
      This text starts a new line after a one-inch space
```

### 4.5 Drawing Rules

To draw a line (horizontal or vertical) on the page, use the `\rule` command:

```
\rule[lift]{width}{height}
```

*width* is the horizontal dimension, *height* is the vertical dimension, and the optional parameter *lift* is the amount raised above the baseline. For example, the line below was drawn with the command `\rule{\textwidth}{1pt}`.

## 4.6 Footnotes

Footnotes are numbered automatically by  $\LaTeX$ . The command `\footnote{footnote text}` should be placed exactly where you want the footnote number to appear, with no extra space between the `\footnote` command and the text before it. For example:

```
This is text with a note.\footnote{This is the note text.
Here it is at the bottom of the page.}
```

produces:

This is text with a note.<sup>3</sup>

## 4.7 Centering

If you have only one line to center, it's easiest to use the plain  $\TeX$  command `\centerline`:

```
\centerline{This line is centered}
```

If you have several lines to be centered horizontally, the `center` environment is convenient. The example below produces three lines, each horizontally centered.

```
\begin{center}
This is line one. \\
This is line two. \\
This is line three.
\end{center}
```

## 4.8 Quotations

The `quote` environment begins a new line and indents text from both sides. It is delimited with `\begin{quote}` and `\end{quote}`. Any special effects (such as changes to the type size or style) started within the `quote` environment are terminated by `\end{quote}`.

New paragraphs are block style: that is, no indent and a blank line as separation. This section is inside a `quote` environment.

There is also a very similar environment called `quotation`. The only difference is that paragraphs in the `quotation` environment are indented with no blank line between.

---

<sup>3</sup>This is the note text. Here it is at the bottom of the page.

## 4.9 Reproducing Text As-Is

To reproduce new lines and spaces exactly as they are in your input file, you have a choice of several methods.

The `verbatim` environment prints its text in typewriter-style type and sets it off from the rest of the document with blank lines before and after. (It does not indent.) To use it, surround the text with the commands `\begin{verbatim}` and `\end{verbatim}`. The only  $\LaTeX$  command obeyed inside this environment is `\end{verbatim}`. For example, the following input

```
\begin{verbatim}
    All   spacing is displayed in verbatim as entered.
So are special characters    ! @ # & * ( ) _ } ] \ | > <
    Verbatim is used to display LaTeX commands in this document.
\end{verbatim}
```

produces:

```
    All   spacing is displayed in verbatim as entered.
So are special characters    ! @ # & * ( ) _ } ] \ | > <
    Verbatim is used to display LaTeX commands in this document.
```

A variation on the `verbatim` environment, called the `alltt` environment, is provided by a user-written package. It is used in the same way as the `verbatim` environment and works the same, in that spaces and lines are retained from the input file. The difference is that  $\LaTeX$  commands are recognized inside this environment. It cannot be used to reproduce  $\LaTeX$  commands, but it is very useful for special effects, such as printing in roman or italic type instead of typewriter. Before you can use this environment, you must include the command `\usepackage{alltt}` following the `\documentclass` command.

If the text is short enough to be contained on one input line and should not be set off, you can use the `\verb` command. For example,

```
\verb+This is inside the \verb command+
```

produces:

```
This is inside the \verb command
```

(Note that the “+” is used to delimit the verbatim text. Actually any character except the “\*” can be used as the delimiter.)

## 4.10 Lists

The three L<sup>A</sup>T<sub>E</sub>X list environments are demonstrated in the following examples showing both the output and the input used to create them. All three environments use the `\item` command to start new items in the list.

The `enumerate` environment numbers items sequentially, the `itemize` environment puts a bullet in front of each item, and the `description` environment puts a boldface word or phrase in front of each item.

### Example of Enumerate

<pre>\begin{enumerate}   \item This is the first item.   \item This is the second item.   \item This is the third item. \end{enumerate}</pre>	<ol style="list-style-type: none"> <li>1. This is the first item.</li> <li>2. This is the second item.</li> <li>3. This is the third item.</li> </ol>
---	---

### Example of Itemize

<pre>\begin{itemize}   \item This is the first item.   \item This is the second item.   \item This is the third item. \end{itemize}</pre>	<ul style="list-style-type: none"> <li>• This is the first item.</li> <li>• This is the second item.</li> <li>• This is the third item.</li> </ul>
---	--

### Example of Description

<pre>\begin{description}   \item[enumerate] Puts numbers   in front of the items   \item[itemize] Puts bullets   in front of the items   \item[description] Puts words   in front of the items \end{description}</pre>	<p><b>enumerate</b> Puts numbers in front of the items</p> <p><b>itemize</b> Puts bullets in front of the items</p> <p><b>description</b> Puts words in front of the items</p>
--	--

Below is an example of nesting list environments to achieve a different format.

<pre>Here are some useful environments: \begin{itemize} \item center environment \item quote environment \item the three list environments:   \begin{enumerate}     \item enumerate (uses numbers)     \item itemize (uses bullets)     \item description (uses words)   \end{enumerate} \end{itemize}</pre>	<pre>Here are some useful environments: • center environment • quote environment • the three list environments:   1. enumerate (uses numbers)   2. itemize (uses bullets)   3. description (uses words)</pre>
--	---

## 4.11 Cross References

In longer documents, there are often cross references to sections, figures, tables, or equations.  $\LaTeX$  provides the following commands for cross referencing:

<code>\label{marker}</code>	set a marker for future reference
<code>\ref{marker}</code>	include the number of the section, figure, etc. of the corresponding <code>\label</code> command
<code>\pageref{marker}</code>	include the page number of the corresponding <code>\label</code> command

*marker* is an identifier that you choose—it may contain letters, numbers, or other characters (except for  $\LaTeX$ 's special list of characters). It can be helpful (but not necessary) to start the marker name with a tag that identifies what is being marked: for example, `sec:` for sections, `eqn:` for equations, `fig:` for figures, etc. (See the example below.) The `\label` command should be placed immediately after a sectioning command, within an equation environment, or inside a figure or table environment *immediately following* the `caption` command.

<code>\label{sec:xrefs}</code>	
For information on cross references, see section <code>\ref{sec:xrefs}</code> on page <code>\pageref{sec:xrefs}</code> .	For information on cross references, see section 4.11 on page 21.

Note that  $\LaTeX$  uses the numbers from the `.aux` file produced by the previous run, so it will take two runs (sometimes more) to get the cross references correct.

## 4.12 Placing Figures and Tables on the Page

If you have figures or tables in your document, you need to make sure that they are not broken across pages. To accomplish this,  $\LaTeX$  provides two environments, `figure` and `table`, which “float” material inside the environment to a later page if it does not fit on the current page. Meanwhile,  $\LaTeX$  fills the current page with text to avoid half empty pages. The commands to use these two environments look like:

<code>\begin{figure}</code>	<code>\begin{table}</code>
... figure material ...	... tabular material ...
<code>\end{figure}</code>	<code>\end{table}</code>

There are also “starred” versions of these environments (`table*` and `figure*`), which, in 2-column text, place floats across both columns. Note that these environments have nothing to do with actually preparing your figures or tables (discussed later in this memo); they only ensure that whatever material is inside gets “floated” and is not broken between pages.

By default,  $\LaTeX$  first tries to put the figure or table at the top of the current page. If it doesn't fit there, it next tries the bottom of the current page, then the top of the next page. Failing that, it will try to place the figure or table on a page containing only

floated material (no text). Usually, the default behavior places the floats satisfactorily. However, if you wish, you can supply an optional parameter to specify a list of preferred locations; for example, if you begin a table:

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

LaTeX will first try to place the table “here”—that is, at the spot you gave the `\begin{table}` command, then the top of the next page, and finally on a page containing only floats (which often means on a page by itself). The possible positioning parameters are:

- h**     *here*: at the place in the text where the environment occurs.
- t**     *top*: at the top of a text page.
- b**     *bottom*: at the bottom of a text page.
- p**     *page*: on a special page containing only floats.

#### 4.12.1 Making a Caption

Within the figure or table environment, you can use the command `\caption{caption text}` to supply a caption. Usually the caption for a table is typed above the table and the caption for a figure below the figure. The `table` environment prefixes the caption text with “Table *n*.” (where *n* is the number of the table supplied by LaTeX); the `figure` environment prefixes the text with “Figure *n*.” In fact, the caption prefix is the only difference between the figure and table environments. The `\caption` command also supplies information for a List of Figures and/or a List of Tables. To print out such a list at the beginning of your document, simply place the command `\listoftables` or `\listoffigures` after the `\tableofcontents` command.

If your caption is very long and you do not want the whole thing repeated in the list of figures or tables, you can use an optional parameter (specified in square brackets) to supply a shorter version for the List. For example, a table caption might look like:

```
\caption[A short caption for the list of tables]{This is a very
    long caption for this table, requiring lots of explanation.
    It could go on for several lines.}
```

You can also use the `\label` command (see section 4.11 on cross referencing) so that you can refer to your figure or table elsewhere in the document. Note that the `\label` command must immediately follow the `caption` command. If it precedes the caption, the number used in the reference will be incorrect.

#### 4.12.2 Example

The example below produces a figure, which LaTeX places “here” because there happens to be room for it. Note that you can put anything inside the figure environment: it can be blank space created with the `\vspace` command, or it can be an imported PostScript

graphic (see section 7.4). Likewise, a table environment can contain any  $\LaTeX$  commands or text, but usually contains tabular material inside the `tabular` environment (see section 5.2).

```
\begin{figure}[htp]
  \begin{center}
    \fbox{\parbox[t][2in][c]{4in} %top, 2in deep, v-centered, 4in wide
          {\centerline{This space intentionally left blank}}}
  \end{center}
  \caption{What an Interesting Picture!}
  \label{fig:space}
\end{figure}
```

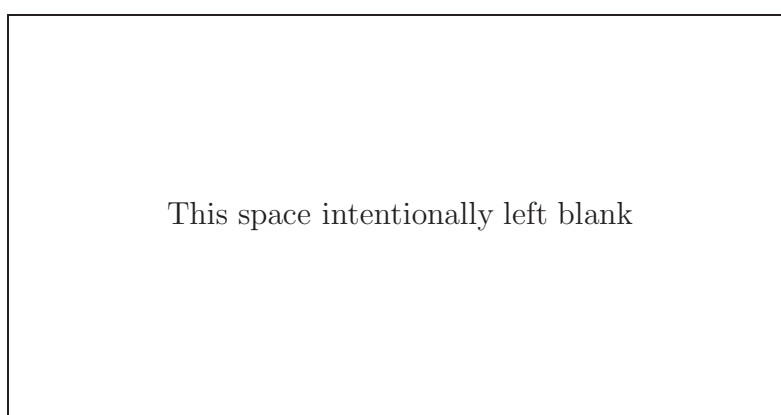


Figure 1: What an Interesting Picture!

#### 4.12.3 Overcoming Problems with Float Placement

When placing floated material,  $\LaTeX$  follows certain rules, explained in detail in Lamport's  *$\LaTeX$  User's Guide and Reference Manual*, section C.9. If you find that your table or figure is not appearing where you think it should, it could be that  $\LaTeX$ 's constraints on how much of a text page can be taken up by floats are getting in the way. You can override these constraints by including a `!` in the optional argument containing the positioning parameters. For example, if you use `\begin{table}[!htp]`,  $\LaTeX$  will try extra hard to place the float where you want it.

Do NOT give only one possible placement parameter (e.g., `h`), because if the float cannot be placed where you specified, it will be pushed to the end of the section or document with all subsequent figures or tables behind it.

$\LaTeX$  has room in its memory to hold only a certain number of floats. If you want to include many figures or tables together, without intervening text, you can avoid problems by including the command `\clearpage` after every approximately 10 floats. This command prints all queued floats and starts a new page.

## Chapter 5. Tables

There are two environments in  $\text{\LaTeX}$  for making tables: the `tabbing` environment, which uses `\begin{tabbing}... \end{tabbing}`, and the `tabular` environment, which uses `\begin{tabular}... \end{tabular}`.

The `tabbing` environment works in a manner similar to a typewriter—you set the tabs and then move from one to another. Tab stops may be reset on any line. Page breaks within the table are allowed since each line is processed individually.

The `tabular` environment allows you to construct a much fancier looking table: for example, you can specify the alignment of each column and use horizontal and vertical lines. However, these tables cannot be broken across pages because  $\text{\LaTeX}$  reads in the entire table at once to establish correct column widths.

### 5.1 Tabbing

Tabbing uses the following commands:

`\=` Set a tab stop

`\>` Move right to the next tab stop

`\&` Terminate a line

Tabs are usually set in the first line but may also be added in later lines. A special line ending with the command `\kill` may be used to set tabs but not print the line. Below are two examples of `tabbing`. Note that the last entry does not require a `\&` to end the line.

#### Example 1: A Very Simple Case

```
\begin{tabbing}
Column 1 \= Column 2 \= Column 3 \= Column4 \&
Col 1 \> Col 2 \> Col 3 \> Col 4 \&
one \> two \> three \> four
\end{tabbing}
```

Produces:

Column 1	Column 2	Column 3	Column 4
Col 1	Col 2	Col 3	Col 4
one	two	three	four

**Example 2: A Little More Complex**

```

\begin{tabbing}
\hspace{2in} \= \hspace{2in} \= \kill
First column \> Second column \> Third column \\
\> Second          \> Third \\
\hspace{1in} \\ % make a blank line
This Text extends past tab 1 \>\> Third column \\
\> Text spans columns two and three \\
xxxxxxxx \= xxxxxxxx \= xxxxxxxx \= \kill % set up new tab stops
Col 1 \> Col 2 \> Col 3 \> Col 4
\end{tabbing}

```

Produces:

```

First column          Second column          Third column
                    Second                 Third

This Text extends past tab 1                Third column
Text spans columns two and three
Col 1   Col 2   Col 3   Col 4

```

**5.2 Tabular**

The `tabular` environment requires an additional argument that specifies the alignment of each column (centered, left justified, etc.):

```
\begin{tabular}{align}
```

You may substitute any combination of the following symbols for the *align* argument:

- l Left-justified column entry
- c Centered column entry
- r Right-justified column entry
- p Paragraph column entry
- | Vertical rule column
- || Double vertical rule column

The width necessary for each column is determined automatically from the widest entry. Inside the `tabular` environment, use the tab character (`&`) to move to the next column, `\\` to end each line (except the last one), and `\hline` to insert a horizontal line.

The following examples illustrate the `tabular` environment. Note that you can center a table by enclosing the `tabular` environment inside a `center` environment.

## 5.2.1 A Simple Ruled Table

```

\begin{center}
\begin{tabular}{|l|c|r|} % 3 cols (left, ctr, right); vert. lines
\hline
Name & Oblateness & Diameter \\
\hline
Mercury & 0 & 3,100 \\
Venus & 0 & 7,700 \\
Earth & 1/297 & 7,927 \\
Mars & 1/192 & 4,200 \\
Jupiter & 1/15 & 88,700 \\
Saturn & 1/9.5 & 75,100 \\
Uranus & 1/14 & 32,100 \\
Neptune & 1/40 & 27,700 \\
Pluto & ? & 3,600 \\
\hline
\end{tabular}
\end{center}

```

Produces:

Name	Oblateness	Diameter
Mercury	0	3,100
Venus	0	7,700
Earth	1/297	7,927
Mars	1/192	4,200
Jupiter	1/15	88,700
Saturn	1/9.5	75,100
Uranus	1/14	32,100
Neptune	1/40	27,700
Pluto	?	3,600

### 5.2.2 Using Paragraph Columns, Spanning Columns

The following example illustrates making paragraphs within a table and placing text across multiple columns using the `\multicolumn` command. This command has the form:

```
\multicolumn{n}{pos}{item}
```

where  $n$  is the number of columns to be spanned,  $pos$  specifies the alignment of the item, and  $item$  is the text.

This example is inside the `table` environment, which ensures that the table will be “floated” rather than broken across a page (see section 4.12) and also provides the opportunity to supply a caption. In this example, the optional parameter `[hb]` specifies that the table should appear “here” or at the bottom of the page.

To center tabular material inside the `table` environment, you can use the declaration `\centering`, which has the same effect as the `center` environment. (The centering action will be confined to inside the `table` environment.)

```
\begin{table}[hb] % place table ‘‘here’’ or at bottom of page
\centering
\caption{More Things You can Do With Tabular}
\bigskip
\begin{tabular}{|l|l|p{2.5in}|}
\hline
\multicolumn{2}{|c|}{Text in columns 1 and 2}
& A paragraph 2.5 inches wide \\
\hline
Column 1 & Column 2 & This text is a paragraph. It will wrap
& around to the next line if necessary. \\
\hline
Column 1 & Column 2 & The paragraph column \\
\hline
\end{tabular}
\end{table}
```

Table 1: More Things You can Do With Tabular

Text in columns 1 and 2		A paragraph 2.5 inches wide
Column 1	Column 2	This text is a paragraph. It will wrap around to the next line if necessary.
Column 1	Column 2	The paragraph column

### 5.2.3 Aligning on the Decimal Point

Although L<sup>A</sup>T<sub>E</sub>X does not provide a way to align numeric columns on a decimal point, it is possible to accomplish this by using two columns (the first right-aligned and the second left-aligned) and changing the column separator to be a decimal point.<sup>4</sup>

L<sup>A</sup>T<sub>E</sub>X allows you to change the column separator with the command `@{...}`, which replaces the usual intercolumn space with whatever you put inside the curly braces. Therefore, if you use `@{.}` in the `\begin{tabular}` line, a “.” will be placed between columns. When you enter the data, you must replace the decimal point in your numbers with a column separator (`&`). The example below illustrates how to do this.

```
\begin{tabular}{c r @{.} l}
Pi expression      &                               & Value
\multicolumn{2}{c}{Value} \\
\hline
 $\pi$                 & 3&1416 & 3.1416
 $\pi^\pi$               & 36&46  & 36.46
 $(\pi^\pi)^\pi$          & 80662&7 & 80662.7
\end{tabular}
```

### 5.2.4 Suppressing Leading or Trailing Space

Another creative use of the `@{...}` command is to suppress leading or trailing space in a table. In this case the command would be `@{}`, indicating no space should be inserted:

```
\begin{tabular}{@{} l @{} }
\hline
no leading or trailing space \\
\hline
\end{tabular}
```

```
\begin{tabular}{l}
\hline
leading space left and right \\
\hline
\end{tabular}
```

---

<sup>4</sup>For an another method, see the `dcolumn` package, provided as part of the L<sup>A</sup>T<sub>E</sub>X “tools” bundle.

## Chapter 6. Mathematics

One of the greatest strengths of  $\LaTeX$  is its ability to typeset formulas and equations. To make it easier to enter mathematical text,  $\LaTeX$  has defined several hundred Greek symbols, mathematical symbols, delimiters, and operators. These are listed in Appendix A of this memo.

$\LaTeX$  has several modes for setting math text, which are described below. When in math mode,  $\LaTeX$  sets type differently than when in text mode. For example, all letters are set in the math italic typeface, and spaces in the input are ignored because  $\LaTeX$  uses its own “mathematically correct” spacing. Therefore, if you want to use normal text or retain spaces while in math mode, you must enclose this text with an “mbox”: `\mbox{this is normal text}`. Also, new paragraphs are not allowed within math mode, so be sure not to leave any blank lines.

If your mathematical expressions are particularly complex or sophisticated, you may want to look at AMS- $\LaTeX$ , a collection of packages that provides extensions to  $\LaTeX$ 's mathematical capabilities.<sup>5</sup>

### 6.1 In-line Math

The in-line math environment allows you to place mathematical formulas in the midst of ordinary text. Typically such mathematical formulas are roughly the same size as the text they're embedded in. This environment can be invoked in one of three ways:

```

$ ... $           (Used in plain TeX)
\< ... \)
\begin{math} ... \end{math}

```

The following paragraph is typical of the use of in-line math:

```

The quadratic equation  $ax^2+bx+c = 0$  has two roots
whose nature depends on the sign of the discriminant
 $d = \sqrt{b^2 - 4ac}$ . If  $d>0$ , then there are
two distinct real roots; if  $d=0$ , then there are two
real and equal roots; and if  $d<0$ , then the two roots
are conjugate complex numbers.

```

produces:

The quadratic equation  $ax^2 + bx + c = 0$  has two roots whose nature depends on the sign of the discriminant  $d = \sqrt{b^2 - 4ac}$ . If  $d > 0$ , then there are two distinct real roots; if  $d = 0$ , then there are two real and equal roots; and if  $d < 0$ , then the two roots are conjugate complex numbers.

---

<sup>5</sup>AMS- $\LaTeX$  is available on RCS; for more information, see the section starting “AMSLATEX” in the file `/campus/doc/text/Latex2e/Packages/package-summary`.

## 6.2 Display Math (for unnumbered equations)

The `displaymath` environment puts space before and after equations and centers them by default. For left-justified equations, include `fleqn` as an option in the `documentclass` command: for example, `\documentclass[fleqn]{article}`. Like in-line math, `displaymath` can be invoked in one of three ways:

```

    $$ ... $$                (Used in plain TEX)
    \[ ... \]
    \begin{displaymath} ... \end{displaymath}

```

The next paragraph illustrates the solution to the quadratic equation:

```

    The quadratic equation $ax^2 + bx + c = 0$ has two roots
    $$ x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} $$
    where the nature of the roots is determined by the sign
    of the discriminant $b^2 - 4ac$.

```

produces:

The quadratic equation  $ax^2 + bx + c = 0$  has two roots

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

where the nature of the roots is determined by the sign of the discriminant  $b^2 - 4ac$ .

## 6.3 Equation Environment (for numbered equations)

The `equation` environment is exactly like the `displaymath` environment except that an equation number in parentheses is placed to the right of the displayed formula. (For left-side numbering, include `leqno` as an option to the `documentclass` command.) The environment is invoked with `\begin{equation}... \end{equation}` as illustrated below:

```

    The derivative of the function $f(x)$ at the point $x_0$ is
    \begin{equation}
    f'(x_0) =
    \lim_{x \rightarrow x_0}
    \frac{f(x) - f(x_0)}{x - x_0}
    \end{equation}

```

produces:

The derivative of the function  $f(x)$  at the point  $x_0$  is

$$f'(x_0) = \lim_{x \rightarrow x_0} \frac{f(x) - f(x_0)}{x - x_0} \quad (1)$$

## 6.4 Eqnarray Environment (for multiline equations)

The `eqnarray` environment builds a three-column array of numbered equations, with the first column right-justified, the second centered, and the third left-justified. It is used mainly for displaying multi-line formulas. It numbers each line by default, but you can include the command `\nonumber` on any line to suppress the equation number. (Or you can use the alternative environment `eqnarray*`, which does not number any lines.) The following example illustrates this environment. Note that you can use the `\label` and `\ref` commands to refer to equation numbers in the text.

```

These three products can be expanded and simplified to
produce these equations:
\begin{eqnarray}
(a + b)(a + b) & = & a^2 + ab + ba + b^2 & \nonumber \\
& & & \label{eq:simp1} \\
(a + b)(a - b) & = & a^2 - ab + ba - b^2 & \nonumber \\
& & & \label{eq:simp2} \\
(a + b)^3 & = & a^3 + 3a^2b + 3ab^2 + b^3 & \label{eq:simp3}
\end{eqnarray}

```

The results after simplification are shown in equations `\ref{eq:simp1}`, `\ref{eq:simp2}`, and `\ref{eq:simp3}`.

produces:

These three products can be expanded and simplified to produce these equations:

$$\begin{aligned} (a + b)(a + b) &= a^2 + ab + ba + b^2 \\ &= a^2 + 2ab + b^2 \end{aligned} \tag{1}$$

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

$$(a + b)^3 = a^3 + 3a^2b + 3ab^2 + b^3 \tag{3}$$

The results after simplification are shown in equations [1](#), [2](#), and [3](#).

## 6.5 Array Environment (for matrices, etc.)

The `array` environment is not itself a math environment; *it must be enclosed within a math environment of your choosing*. It is used for building rectangular arrays of numbers, matrices, etc. `array` is the math equivalent of `tabular` and uses the same syntax.

Here is an example of the `array` environment used to build a  $5 \times 5$  magic square:

```


$$\begin{array}{ccccc}
17& 24& 1& 8& 15 \\
23& 5& 7& 14& 16 \\
4& 6& 13& 20& 22 \\
10& 12& 19& 21& 3 \\
11& 18& 25& 2& 9
\end{array}$$


```

produces:

$$\begin{array}{ccccc}
17 & 24 & 1 & 8 & 15 \\
23 & 5 & 7 & 14 & 16 \\
4 & 6 & 13 & 20 & 22 \\
10 & 12 & 19 & 21 & 3 \\
11 & 18 & 25 & 2 & 9
\end{array}$$

Here is an example for a general matrix:

```


$$\begin{equation}
A = \left( \begin{array}{cccc}
a_{1,1} & a_{1,2} & \dots & a_{1,n} \\
a_{2,1} & a_{2,2} & \dots & a_{2,n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{m,1} & a_{m,2} & \dots & a_{m,n}
\end{array} \right)
\end{equation}$$


```

produces:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{pmatrix} \tag{4}$$

Section 6.6.6 (Large Delimiters) on page 35 has more examples of arrays.

## 6.6 Building Mathematical Expressions

### 6.6.1 Superscripts and Subscripts

In math mode, the symbols ‘^’ and ‘\_’ are used to make superscripts and subscripts. If more than one character is to be used as a superscript or subscript, enclose the characters in braces. For example:

```


$$2^{2^2} = 2^4 = 4^2$$


$$a^2_{i_1} = b^2_{i,j}$$


$${}_2F^1_3$$


$$\{ {}_2^1P^3 \}_3_4$$


```

produces:

$$2^{2^2} = 2^4 = 4^2$$

$$a^2_{i_1} = b^2_{i,j}$$

$${}_2F^1_3$$

$$\{ {}_2^1P^3 \}_3_4$$

### 6.6.2 Spaces in Math Mode

In math mode, L<sup>A</sup>T<sub>E</sub>X ignores blanks typed in the input. However, the following symbols allow you to add (or subtract) small amounts of space in your equations:

`\;`; thick space    `\:`; medium space    `\,`; thin space    `\!`; negative space

In addition, you can make a larger space (about the width of an “M”) with the command `\quad`. The command `\qqquad` provides twice as much space.

### 6.6.3 Dots, Braces, and Bars

The commands `\ldots`, `\cdots`, `\vdots`, and `\ddots` are used to produce a string of three dots on the base line, on the math centerline, vertically, or on the diagonal, respectively. The commands `\overline`, `\underline`, `\overbrace`, and `\underbrace` are used to build horizontal groups. For example:

```

\begin{eqnarray*}
2^n & = & \overbrace{2 \times 2 \times \cdots \times 2}^{\text{n terms}} \\
k \cdot x & = & \underbrace{x + x + \cdots + x}_{\text{k terms}}
\end{eqnarray*}

```

produces:

$$2^n = \overbrace{2 \times 2 \times \cdots \times 2}^{\text{n terms}}$$

$$k \cdot x = \underbrace{x + x + \cdots + x}_{\text{k terms}}$$

### 6.6.4 Fractions

The ‘/’ can be used to make simple fractions, but the `\frac` command is used for most fractions; its two arguments are the numerator and denominator. Two variations of a fraction are provided by the commands `\atop` and `\choose`.

```


$$\frac{a}{b} \quad \frac{a/b}{c/d}$$


$$a \atop b \quad a \choose b$$


$$1 + 2 + 3 + \dots + n = \frac{n(n+1)}{2}$$


$$\lim_{n \rightarrow \infty} \frac{1}{n} = 0 \quad y'' = \frac{d^2y}{dx^2}$$


```

produces:

$$\frac{a}{b} \quad \frac{a/b}{c/d} \quad a \atop b \quad \begin{pmatrix} a \\ b \end{pmatrix}$$

$$1 + 2 + 3 + \dots + n = \frac{n(n+1)}{2}$$

$$\lim_{n \rightarrow \infty} \frac{1}{n} = 0 \quad y'' = \frac{d^2y}{dx^2}$$

### 6.6.5 Radicals, Integrals, and Summations

The `\sqrt` command creates a square root sign for its mandatory argument. An optional argument for the radicand allows you to construct cube roots, nth roots, etc. The sign automatically grows to fit the argument, as these examples show:

```


$$\sqrt{2} = 1.4142 \quad c = \sqrt{a^2 + b^2}$$


$$\sqrt[3]{8} = 2 \quad \sqrt{\frac{n(n+1)}{2}}$$


```

produces:

$$\sqrt{2} = 1.4142 \quad c = \sqrt{a^2 + b^2} \quad \sqrt[3]{8} = 2 \quad \sqrt{\frac{n(n+1)}{2}}$$

Integral signs are produced with the `\int` command, and summation signs are produced with the `\sum` command. These symbols are often used with superscripts and subscripts:

```


$$\sum_{n=1}^{\infty} \quad \int_a^b e^{x^2} dx$$


$$\sqrt{\sum_{i=1}^n i} \quad \sum_{j=2}^{\infty}$$


$$\sum \Delta V = \iiint_V dv$$


```

produces:

$$\sum_{n=1}^{\infty} \quad \int_a^b e^{x^2} dx$$

$$\sqrt{\sum_{i=1}^n i} \quad \sum_{j=2}^{\infty}$$

$$\sum \Delta V = \iiint_V dv$$

### 6.6.6 Large Delimiters

The commands `\left` and `\right` are used to put large delimiters (such as  $( ) [ ]$ ) in displayed formulas. (For braces, use `\{` and `\}`). See Appendix A for a complete list of delimiters. There must be a `\right` for every `\left` and vice versa *for each output line*, although the left and right delimiters need not be the same type. You can even use a period to create an invisible delimiter—very useful if you need an odd number of delimiters. Large delimiters are often used with the `array` environment. Below are two examples of large delimiters. (For another example, see Section 6.5 on page 32.)

```


$$\det = \begin{vmatrix} c_0 & c_1 & c_2 & \dots & c_n \\ c_1 & c_2 & c_3 & \dots & c_{n+1} \\ c_2 & c_3 & c_4 & \dots & c_{n+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_n & c_{n+1} & c_{n+2} & \dots & c_{2n} \end{vmatrix} > 0.$$


```

produces:

$$\det = \begin{vmatrix} c_0 & c_1 & c_2 & \dots & c_n \\ c_1 & c_2 & c_3 & \dots & c_{n+1} \\ c_2 & c_3 & c_4 & \dots & c_{n+2} \\ \vdots & \vdots & \vdots & & \vdots \\ c_n & c_{n+1} & c_{n+2} & \dots & c_{2n} \end{vmatrix} > 0.$$

```


$$\text{signum}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{otherwise} \end{cases}$$


```

produces:

$$\text{signum}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{otherwise} \end{cases}$$

## Chapter 7. Special Topics

### 7.1 Managing a Large Document

If you are preparing a book, thesis, or any document with multiple parts, it's convenient to work on the different chapters or sections separately. To do this, create a separate `.tex` file for each section, and create a “root” `.tex` file that contains the `\documentclass` command, the preamble material, the `\begin{document}` and `\end{document}` commands, and multiple `\include` commands. The `\include` commands tell  $\text{\LaTeX}$  to read the separate `.tex` files that contain the text of the document. Each `\include` command starts a new page in your output. Note that the individual section files do not include `\begin{document}` and `\end{document}` commands.

The command, `\includeonly`, *which must go in the preamble*, can be used to select from the `\include` list only certain file(s) to be processed.

As an example, the root file for a book with 6 chapters might be called `mybook.tex`, and the files for the chapters might be named `chap1.tex`, `chap2.tex`, etc. The root file `mybook.tex`, *which is the file on which you run  $\text{\LaTeX}$* , would look something like:

```
\documentclass{book}
\includeonly{chap1, chap2}    % only these files will be processed
\begin{document}
\include{chap1}
\include{chap2}
\include{chap3}
\include{chap4}
\include{chap5}
\include{chap6}
\end{document}
```

### 7.2 Preparing a Bibliography

To prepare a bibliography, the first step is to cite various works within your text using the `\cite{key}` command. *key* is a reference keyword of your choosing that identifies the work. For example your document might include, at the appropriate places, the commands: `\cite{lamport} \cite{kopka} \cite{goossens}`.

These commands place numbers (enclosed in square brackets) in the text that match the numbers which will be automatically generated in the bibliography. (Remember to run  $\text{\LaTeX}$  twice to get correct numbers in the text!)

To include an item in the bibliography without citing it in the text, use the `\nocite` command; for example: `\nocite{smith, jones}`.

Then, at the end of the document, you can format the bibliography using the a special environment called `thebibliography`.

Below is an example of a `thebibliography` environment, followed by the output it produces. On the `\begin{thebibliography}` line, the width of the item in the second pair of braces determines the indent of the entries. This item is usually a dummy number with as many digits as the highest-numbered entry. In this case, the number 9 indicates there are fewer than 10 entries. If there are 10 or more entries but less than 100, use the number 99.

Within this environment, each entry starts with the `\bibitem` command. The argument for this command is the same keyword used in the corresponding `\cite` command in the text. The `\bibitem` commands automatically generate a number in square brackets before each entry. The first entry in the list will be numbered 1.

Note that in the `book` and `report` classes, **Bibliography** is used as the chapter title; in the `article` class, **References** is used as the section heading.

---

```
\begin{thebibliography}{9}
  \bibitem{lampport} Leslie Lamport. \textit{\LaTeX\ -- A Document
    Preparation System}. Addison-Wesley, Reading, MA, 2nd edition, 1994.
  \bibitem{kopka} Helmut Kopka and Patrick W.~Daly. \textit{A Guide
    to \LaTeXe}. Addison-Wesley, Reading, MA, 1995.
  \bibitem{goossens} Michel Goossens, Frank Mittelbach
    & Alexander Samarin. \textit{The \LaTeX\ Companion}.
    Addison-Wesley, Reading, MA, 1994.
\end{thebibliography}
```

## References

- [1] Leslie Lamport. *L<sup>A</sup>T<sub>E</sub>X – A Document Preparation System*. Addison-Wesley, Reading, MA, second edition, 1994.
  - [2] Helmut Kopka and Patrick W. Daly. *A Guide to L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>*. Addison-Wesley, Reading, MA, 1995.
  - [3] Michel Goossens, Frank Mittelbach & Alexander Samarin. *The L<sup>A</sup>T<sub>E</sub>X Companion*. Addison-Wesley, Reading, MA, 1994.
- 

If you prefer, you can choose a label to identify the work in both the text and in the bibliography, instead of the number assigned by L<sup>A</sup>T<sub>E</sub>X. You do this by including an optional label on the `\bibitem` command. (The `\cite` command doesn't change.) For example, to refer to the third book as “companion,” its entry in the `thebibliography` environment would be:

```
\bibitem[companion]{goossens} Michel Goossens, Frank Mittelbach
  & Alexander Samarin. \textit{The \LaTeX\ Companion}.
  Addison-Wesley, Reading, MA, 1994.
```

For more information, including how to use the BIB<sub>T</sub>E<sub>X</sub> program to generate a bibliography from a database, see any of the above three references.

### 7.3 Generating an Index

It is relatively easy to generate an index by using  $\LaTeX$  commands combined with the support program *MakeIndex*. This memo covers just the basics of *MakeIndex*; details are in the documentation accompanying the program, found on RCS in the file `/campus/doc/text/makeindex.dvi`.

Suppose your  $\LaTeX$  input file is called `myfile.tex`. To make an index, you need to do the following things in that file:

1. Load the package `makeidx` with the command: `\usepackage{makeidx}`
2. In the preamble, place the command: `\makeindex`
3. At the place you want the index to appear (usually the end of the document), put the command: `\printindex`
4. Put index entries throughout the text (as close as possible to the actual word or phrase being indexed) using the command `\index{entry}`, where *entry* is the index entry. Examples of various index entries are shown in the table at the bottom of the page.

Now, generate the index by running  $\LaTeX$ , then running *MakeIndex*, and finally rerunning  $\LaTeX$ , as shown below. (Note that including the extensions `.tex` and `.idx` on the file names is optional.)

<code>latex myfile.tex</code>	$\LaTeX$ generates the file <code>myfile.idx</code> , containing all the index entries and page numbers.
<code>makeindex myfile.idx</code>	<i>MakeIndex</i> produces the file <code>myfile.ind</code> , which contains the $\LaTeX$ commands to format the index.
<code>latex myfile.tex</code>	When $\LaTeX$ finds the <code>\printindex</code> command, it reads in the file <code>myfile.ind</code> . The resulting <code>.dvi</code> file includes the formatted index.

If you change your document after producing the `.ind` file, be sure to run *MakeIndex* again to create an up-to-date `.ind` file before running  $\LaTeX$  for the final time.

The package `showidx`, which comes with  $\LaTeX$ , prints all index entries in the margin of the text. This is a useful tool for checking the index in draft copies.

#### Examples of Index Entries

$\LaTeX$ Input	Index Entry	Comments
<code>\index{colors}</code>	colors, 14	Main entry
<code>\index{colors!blue}</code>	blue, 14	Subentry under "colors"
<code>\index{colors!blue!navy}</code>	navy, 15	Subsubentry
<code>\index{gnu@\textsl{gnu}}</code>	<i>gnu</i> , 22	Formatted text
<code>\index{gnat \textbf}</code>	gnat, <b>25</b>	Formatted page number

## 7.4 Including and Manipulating PostScript Graphics<sup>6</sup>

The easiest way to put graphics into your L<sup>A</sup>T<sub>E</sub>X document is to first create the graphic using a software package (such as xfig, Gnuplot, MATLAB, CorelDraw, etc.) and save the graphic in EPS (Encapsulated PostScript) format.

You can then include the graphic in your document by using the L<sup>A</sup>T<sub>E</sub>X `graphicx` package (part of the L<sup>A</sup>T<sub>E</sub>X graphics bundle) and the PostScript printer driver `dvips`. To include EPS graphics, first put in your preamble the command:

```
\usepackage{graphicx}
```

The `graphicx` package provides the command `\includegraphics`, which allows you to specify the name of the PostScript file as well as optional arguments for scaling or rotating the graphic. The command (which you will normally put inside the `figure` environment, described in section 4.12) will look something like:

```
\includegraphics[width=4in]{myfigure.eps}
```

Inside the `[...]`, you can specify a number of optional arguments, separated by commas. In this case, `width=4in` will scale the width of graphic to four inches. Since no height is specified, the height will be scaled so that the figure remains in proportion. The most commonly used arguments are:

<b>width</b>	scale graphic to the specified width
<b>height</b>	scale graphic to the specified height
<b>scale</b>	scale graphic by scale factor. ( <code>scale=2</code> makes the graphic twice as large as its natural size; <code>scale=.5</code> makes it half as large.)
<b>angle</b>	rotate graphic by specified number of degrees. A positive number indicates the counter-clockwise direction. ( <code>angle=90</code> rotates 90 degrees counter-clockwise.)

Note that in the following example, the graphic is *first* rotated counter-clockwise by 90 degrees and *then* scaled to a width of 8 inches:

```
\includegraphics[angle=90,width=8in]{myfigure.eps}
```

Complete documentation for the graphics bundle is in the file

```
/campus/doc/text/Latex2e/Packages/Graphics/grfguide.dvi
```

You can view it with `xdvi`; information on the `graphicx` package is in section 4.4.

To print an entire figure, including the caption, in landscape orientation, use the `rotating` package in addition to `graphicx`. Load it as usual with the `\usepackage` command:

---

<sup>6</sup>This section assumes you are using the PostScript driver `dvips` and have a PostScript printer available. Alternatively, if you have a PC with a non-PostScript printer, you can still include EPS graphics if you have both `dvips` and `Ghostview` installed.

```
\usepackage{rotating}
```

This package defines two new environments, `sidewaysfigure` and `sidewaystable` (the latter used for placing wide tables in landscape orientation), which you then use in place of the standard  $\LaTeX$  environments `figure` and `table`, described in section 4.12. For example, to make a landscape page containing a rotated graphic and caption, use commands such as:

```
\begin{sidewaysfigure}
\includegraphics{myfigure.eps}
\caption{Turn the page sideways to look at this figure.}
\end{sidewaysfigure}
```

After you have put the appropriate commands for including EPS graphics into your  $\LaTeX$  file, create the `.dvi` file as usual by running  $\LaTeX$ . It is important to note here that the `.dvi` file does not actually contain the graphics; it only contains commands that the printer driver, `dvips`, subsequently uses to include the graphics. Even so, much of the time `xdvi` is able to display the included PostScript graphics (by calling the `ghostview` program). However, there are some cases, for example if the graphic is in landscape orientation, where `xdvi` does not display the graphic properly. In this case, you can use `dvips` to put the output in a PostScript file and then call `ghostview` directly to view it. Assuming your `dvi` file is called `myfile.dvi`, issue the commands:

```
dvips myfile -o          use -o option to put output in file myfile.ps
ghostview myfile.ps     view the PostScript output on the screen
```

Since `ghostview` has menus, it is quite easy to use. However, there is also a man page for more information.

In the RCS directory, `/campus/doc/text/Latex2e/Packages/Contrib`, there are examples of using both the `graphicx` and the `rotating` packages to include PostScript figures and landscape tables. To see the  $\LaTeX$  commands, use an editor to look at the file `exrotating.tex`; to see the results after running `latex` and `dvips`, use `ghostview` to view the file `exrotating.ps`:

```
ghostview exrotating.ps
```

(If the landscape figures disappear after they are initially displayed, click the “Orientation” button and select “Landscape.”)

For an extensive treatment of EPS graphics, see the document *Using EPS Graphics in  $\LaTeX$ 2e Documents* by Keith Reckdahl of Stanford University. It includes all you would ever want to know with many examples. On RCS, this 53-page PostScript document is in the directory `/campus/doc/text/Latex2e/Packages/Graphics`, in the file `epslatex.ps`. To view it, first change to this directory and then use the `ghostview` program:

```
ghostview epslatex.ps
```

## 7.5 Accents and Special Characters

L<sup>A</sup>T<sub>E</sub>X provides commands that allow you to print accents required by many European languages. The table below shows how to apply a variety of accents to the letter o. Of course, any other letter can be used in place of o.

If you want to place an accent on top of an i or a j, however, you need to print them without their dots. You can print a dotless i or j by typing `\i` and `\j`. For example, `\i` is formed by typing `\u{\i}`.

ò	<code>\' {o}</code>	ó	<code>\' {o}</code>	ô	<code>\^ {o}</code>	õ	<code>\~ {o}</code>
ō	<code>\= {o}</code>	ô	<code>\. {o}</code>	ö	<code>\" {o}</code>	ö	<code>\r {o}</code>
ö	<code>\u {o}</code>	ö	<code>\v {o}</code>	ö	<code>\H {o}</code>	q	<code>\c {o}</code>
o	<code>\d {o}</code>	o	<code>\b {o}</code>	oo	<code>\t {oo}</code>		

Note that the accent commands that are control symbols (i.e., they consist of just one non-letter) can also be used without braces. For example `\^o` and `\^ {o}` both produce ô.

The special letters that are part of European languages can be generated with the following commands:

œ	<code>{\oe}</code>	Œ	<code>{\OE}</code>	æ	<code>{\ae}</code>	Æ	<code>{\AE}</code>
å	<code>{\aa}</code>	Å	<code>{\AA}</code>	ø	<code>{\o}</code>	Ø	<code>{\O}</code>
ß	<code>{\ss}</code>	SS	<code>{\SS}</code>	ł	<code>{\l}</code>	Ł	<code>{\L}</code>
ı	<code>!'</code>	ı	<code>?'</code>				

For example:

```
na\"{\i}ve, r\'esum\'e, {\AA}ngstr{\o}m, se\~norita, stra{\ss}e
```

produces:

naïve, résumé, Ångstrøm, señorita, straÙe

## Appendix A

### Mathematical Symbols

All the following symbols (except the “non-mathematical symbols” listed on the next page) must be used in math mode. To use one of the symbols in ordinary text, put it in math mode by surrounding it with dollar signs:  $\alpha \rightarrow \alpha$ .

The package `amssymb`, available on RCS, provides additional symbols. For a list, use `xdvi` to view the file `/campus/doc/text/Latex2e/Packages/Amslatex/amssymblist.dvi`.

#### Greek Alphabet

$\alpha$	<code>\alpha</code>	$\iota$	<code>\iota</code>	$\varrho$	<code>\varrho</code>
$\beta$	<code>\beta</code>	$\kappa$	<code>\kappa</code>	$\sigma$	<code>\sigma</code>
$\gamma$	<code>\gamma</code>	$\lambda$	<code>\lambda</code>	$\varsigma$	<code>\varsigma</code>
$\delta$	<code>\delta</code>	$\mu$	<code>\mu</code>	$\tau$	<code>\tau</code>
$\epsilon$	<code>\epsilon</code>	$\nu$	<code>\nu</code>	$\upsilon$	<code>\upsilon</code>
$\varepsilon$	<code>\varepsilon</code>	$\xi$	<code>\xi</code>	$\phi$	<code>\phi</code>
$\zeta$	<code>\zeta</code>	$o$	<code>o<sup>1</sup></code>	$\varphi$	<code>\varphi</code>
$\eta$	<code>\eta</code>	$\pi$	<code>\pi</code>	$\chi$	<code>\chi</code>
$\theta$	<code>\theta</code>	$\varpi$	<code>\varpi</code>	$\psi$	<code>\psi</code>
$\vartheta$	<code>\vartheta</code>	$\rho$	<code>\rho</code>	$\omega$	<code>\omega</code>
$\Gamma$	<code>\Gamma</code>	$\Xi$	<code>\Xi</code>	$\Phi$	<code>\Phi</code>
$\Delta$	<code>\Delta</code>	$\Pi$	<code>\Pi</code>	$\Psi$	<code>\Psi</code>
$\Theta$	<code>\Theta</code>	$\Sigma$	<code>\Sigma</code>	$\Omega$	<code>\Omega</code>
$\Lambda$	<code>\Lambda</code>	$\Upsilon$	<code>\Upsilon</code>		

Capitals not shown are produced with English capital letters.

<sup>1</sup>An ordinary “o” looks Greek when used in a math environment:  $\$o\$ \rightarrow o$ .

#### Miscellaneous Symbols

$\aleph$	<code>\aleph</code>	$\prime$	<code>\prime</code>	$\forall$	<code>\forall</code>
$\hbar$	<code>\hbar</code>	$\emptyset$	<code>\emptyset</code>	$\exists$	<code>\exists</code>
$\imath$	<code>\imath</code>	$\nabla$	<code>\nabla</code>	$\neg$	<code>\neg</code>
$\jmath$	<code>\jmath</code>	$\surd$	<code>\surd</code>	$\flat$	<code>\flat</code>
$\ell$	<code>\ell</code>	$\top$	<code>\top</code>	$\natural$	<code>\natural</code>
$\wp$	<code>\wp</code>	$\perp$	<code>\perp</code>	$\sharp$	<code>\sharp</code>
$\Re$	<code>\Re</code>	$\parallel$	<code>\parallel</code>	$\clubsuit$	<code>\clubsuit</code>
$\Im$	<code>\Im</code>	$\angle$	<code>\angle</code>	$\diamondsuit$	<code>\diamondsuit</code>
$\partial$	<code>\partial</code>	$\triangle$	<code>\triangle</code>	$\heartsuit$	<code>\heartsuit</code>
$\infty$	<code>\infty</code>	$\diamond$	<code>\diamond</code>	$\spadesuit$	<code>\spadesuit</code>
$\mathcal{U}$	<code>\mathcal{U}</code>	$\square$	<code>\square</code>	$\backslash$	<code>\backslash</code>
$\Sigma$	<code>\Sigma</code>	$\prod$	<code>\prod</code>	$\coprod$	<code>\coprod</code>
$\int$	<code>\int</code>	$\oint$	<code>\oint</code>	$\Join$	<code>\Join</code>

The symbols whose names are underlined require `latexsym`, a standard L<sup>A</sup>T<sub>E</sub>X package. Load it with the command: `\usepackage{latexsym}`.

## Non-Mathematical Symbols

These symbols can be used in either text mode or math mode:

† `\dag`      ‡ `\ddag`      § `\S`      ¶ `\P`      © `\copyright`      £ `\pounds`

## Function Names

When using function names in a math environment, you can prefix them with a `\` so that they will print as normal text rather than in italics:

<code>\arccos</code>	<code>\arcsin</code>	<code>\arctan</code>	<code>\arg</code>	<code>\cos</code>	<code>\cosh</code>
<code>\cot</code>	<code>\coth</code>	<code>\csc</code>	<code>\deg</code>	<code>\det</code>	<code>\dim</code>
<code>\exp</code>	<code>\gcd</code>	<code>\hom</code>	<code>\inf</code>	<code>\ker</code>	<code>\lg</code>
<code>\lim</code>	<code>\liminf</code>	<code>\limsup</code>	<code>\ln</code>	<code>\log</code>	<code>\max</code>
<code>\min</code>	<code>\Pr</code>	<code>\sec</code>	<code>\sin</code>	<code>\sinh</code>	<code>\sup</code>
<code>\tan</code>	<code>\tanh</code>				

## Delimiters

The following symbols can expand in size to “fit around” the expressions they delimit. To make a delimiter the right size, use it with the `\left ... \right` commands described in section 6.6.6.

<code>(</code>	<code>(</code>	<code>)</code>	<code>)</code>	<code>\lfloor</code>	<code>\rfloor</code>
<code>[</code>	<code>[</code>	<code>]</code>	<code>]</code>	<code>\lceil</code>	<code>\rceil</code>
<code>{</code>	<code>\{</code>	<code>}</code>	<code>\}</code>	<code>\langle</code>	<code>\rangle</code>
<code> </code>	<code> </code>	<code>\ </code>	<code>\ </code>	<code>\uparrow</code>	<code>\Uparrow</code>
<code>/</code>	<code>/</code>	<code>\</code>	<code>\backslash</code>	<code>\downarrow</code>	<code>\Downarrow</code>
				<code>\updownarrow</code>	<code>\Updownarrow</code>

## Relational Operators

<code>\le</code>	<code>\le</code> or <code>\leq</code>	<code>\ge</code>	<code>\geq</code>	<code>\equiv</code>	<code>\equiv</code>
<code>\prec</code>	<code>\prec</code>	<code>\succ</code>	<code>\succ</code>	<code>\sim</code>	<code>\sim</code>
<code>\preceq</code>	<code>\preceq</code>	<code>\succeq</code>	<code>\succeq</code>	<code>\simeq</code>	<code>\simeq</code>
<code>\ll</code>	<code>\ll</code>	<code>\gg</code>	<code>\gg</code>	<code>\asymp</code>	<code>\asymp</code>
<code>\subset</code>	<code>\subset</code>	<code>\supset</code>	<code>\supset</code>	<code>\approx</code>	<code>\approx</code>
<code>\subseteq</code>	<code>\subseteq</code>	<code>\supseteq</code>	<code>\supseteq</code>	<code>\cong</code>	<code>\cong</code>
<code>\sqsubset</code>	<code>\sqsubset</code>	<code>\sqsupset</code>	<code>\sqsupset</code>	<code>\perp</code>	<code>\perp</code>
<code>\sqsubseteq</code>	<code>\sqsubseteq</code>	<code>\sqsupseteq</code>	<code>\sqsupseteq</code>	<code>\bowtie</code>	<code>\bowtie</code>
<code>\in</code>	<code>\in</code>	<code>\ni</code>	<code>\ni</code>	<code>\propto</code>	<code>\propto</code>
<code>\vdash</code>	<code>\vdash</code>	<code>\dashv</code>	<code>\dashv</code>	<code>\models</code>	<code>\models</code>
<code>\smile</code>	<code>\smile</code>	<code>\mid</code>	<code>\mid</code>	<code>\doteq</code>	<code>\doteq</code>
<code>\frown</code>	<code>\frown</code>	<code>\parallel</code>	<code>\parallel</code>		
<code>\not&lt;</code>	<code>\not&lt;</code>	<code>\not&gt;</code>	<code>\not&gt;</code>	<code>\neq</code>	<code>\neq</code>

Note that `\not` before a relation will negate it.

The symbols whose names are underlined require the `latexsym` package.

## Binary Operators

$\pm$	<code>\pm</code>	$\cap$	<code>\cap</code>	$\vee$	<code>\vee</code>
$\mp$	<code>\mp</code>	$\cup$	<code>\cup</code>	$\wedge$	<code>\wedge</code>
$\setminus$	<code>\setminus</code>	$\uplus$	<code>\uplus</code>	$\oplus$	<code>\oplus</code>
$\cdot$	<code>\cdot</code>	$\sqcap$	<code>\sqcap</code>	$\ominus$	<code>\ominus</code>
$\times$	<code>\times</code>	$\sqcup$	<code>\sqcup</code>	$\otimes$	<code>\otimes</code>
$*$	<code>\ast</code>	$\triangleleft$	<code>\triangleleft</code>	$\oslash$	<code>\oslash</code>
$\star$	<code>\star</code>	$\triangleright$	<code>\triangleright</code>	$\odot$	<code>\odot</code>
$\diamond$	<code>\diamond</code>	$\wr$	<code>\wr</code>	$\dagger$	<code>\dagger</code>
$\circ$	<code>\circ</code>	$\bigcirc$	<code>\bigcirc</code>	$\ddagger$	<code>\ddagger</code>
$\bullet$	<code>\bullet</code>	$\triangleup$	<code>\triangleup</code>	$\amalg$	<code>\amalg</code>
$\div$	<code>\div</code>	$\triangledown$	<code>\triangledown</code>	$\triangleleft$	<code>\lhd</code>
$\triangleleft$	<code>\unlhd</code>	$\triangleright$	<code>\unrhd</code>	$\triangleright$	<code>\rhd</code>

The symbols whose names are underlined require the `latexsym` package.

## Math Accents

$\hat{a}$	<code>\hat{a}</code>	$\check{a}$	<code>\check{a}</code>	$\breve{a}$	<code>\breve{a}</code>
$\acute{a}$	<code>\acute{a}</code>	$\grave{a}$	<code>\grave{a}</code>	$\tilde{a}$	<code>\tilde{a}</code>
$\bar{a}$	<code>\bar{a}</code>	$\vec{a}$	<code>\vec{a}</code>	$\dot{a}$	<code>\dot{a}</code>
$\ddot{a}$	<code>\ddot{a}</code>	$\widehat{x-y}$	<code>\widehat{x-y}</code>	$\widetilde{xyz}$	<code>\widetilde{xyz}</code>

## Arrows

$\leftarrow$	<code>\leftarrow</code>	$\longleftarrow$	<code>\longleftarrow</code>	$\uparrow$	<code>\uparrow</code>
$\Leftarrow$	<code>\Leftarrow</code>	$\Lleftarrow$	<code>\Lleftarrow</code>	$\Uparrow$	<code>\Uparrow</code>
$\rightarrow$	<code>\rightarrow</code>	$\longrightarrow$	<code>\longrightarrow</code>	$\downarrow$	<code>\downarrow</code>
$\Rightarrow$	<code>\Rightarrow</code>	$\Longrightarrow$	<code>\Longrightarrow</code>	$\Downarrow$	<code>\Downarrow</code>
$\leftrightarrow$	<code>\leftrightarrow</code>	$\longleftrightarrow$	<code>\longleftrightarrow</code>	$\updownarrow$	<code>\updownarrow</code>
$\Leftrightarrow$	<code>\Leftrightarrow</code>	$\Llongleftrightarrow$	<code>\Llongleftrightarrow</code>	$\Updownarrow$	<code>\Updownarrow</code>
$\mapsto$	<code>\mapsto</code>	$\longmapsto$	<code>\longmapsto</code>	$\nearrow$	<code>\nearrow</code>
$\hookrightarrow$	<code>\hookrightarrow</code>	$\hookrightarrow$	<code>\hookrightarrow</code>	$\searrow$	<code>\searrow</code>
$\leftharpoonup$	<code>\leftharpoonup</code>	$\rightharpoonup$	<code>\rightharpoonup</code>	$\swarrow$	<code>\swarrow</code>
$\leftharpoondown$	<code>\leftharpoondown</code>	$\rightharpoondown$	<code>\rightharpoondown</code>	$\nwarrow$	<code>\nwarrow</code>
$\rightleftharpoons$	<code>\rightleftharpoons</code>	$\leadsto$	<code>\leadsto</code>		

The symbols whose names are underlined require the `latexsym` package.

## Appendix B

### Error Messages

Most files need some debugging before they print properly. This section describes some common errors and shows how to correct them.

#### Errors About Boxes

The most common errors refer to an `underfull hbox` (or `vbox`). These errors are harmless and may be safely ignored. They simply mean there is more blank space on a line (or page) than  $\LaTeX$  thinks is aesthetically pleasing. ( $\LaTeX$  is very fussy.)

Sometimes `underfull hbox` messages are the result of using the `linebreak` command (`\`) inappropriately. For example,  $\LaTeX$  will generate this error message if you use `\` just before or after a new paragraph or before starting an environment that begins on a new line anyway.

Errors that refer to “`overfull hboxes`” mean that a line is too long. For example, you might get a message such as this one:

```
Overfull \hbox (19.45158pt too wide) in paragraph at lines 1206--1215
```

$\LaTeX$  is warning you that a line sticks out into the margin. It was forced to do this because otherwise the line would contain unacceptably wide white spaces. Depending on how the output actually looks, you may or may not want to correct the problem. To correct it, you could reword the sentence so that it breaks better, or instruct  $\LaTeX$  how to hyphenate a certain word. Suppose  $\LaTeX$  doesn't know how to hyphenate the word “environment” that occurs in the line. You can make  $\LaTeX$  put the hyphens in “environment” temporarily or permanently. To do it permanently, put the following command in the preamble (note: more than one word can be added in the command if words are separated by spaces):

```
\hyphenation{en-vi-ron-ment}
```

For a temporary fix, put discretionary hyphens `\-` in the offending word where it appears in the text. This is a one-time fix, and unused discretionary hyphens do not show up in the text:

```
en\-\-vi\-\-ron\-\-ment
```

## Other Common Errors

- You misspelled a command, and  $\LaTeX$  doesn't recognize it.
- You failed to match delimiters. For example, you have a  $\$$  with no matching  $\$$ .
- You used a special or math character in ordinary text. The most common problem character is  $\&$ . Others are  $\$$ ,  $\#$ , and  $\%$ .
- You inserted too many tabs on the line before the  $\backslash$ .
- Whatever you  $\backslash\text{begin}\{\dots\}$ , you must  $\backslash\text{end}\{\dots\}$ . This includes braces:  $\{ \dots \}$ .
- A command wasn't given all its arguments and is trying to use the rest of the file for an argument.

## Errors Relating to Memory

Sometimes  $\LaTeX$  complains that it's out of memory, but it usually isn't. The symptoms listed below can be fixed as described:

It won't do all your tables	Put $\backslash\text{clearpage}$ between them.
It won't do a three-page table	That's right, it must be broken up.
You just want one big paragraph	Sorry, $\LaTeX$ needs breaks.
You are using <code>tabular</code>	Did you leave out a $\backslash$ or have more too many tabs on a line?
You used some $\&$ 's or $\$$ 's in text	Enter them as $\backslash\&$ or $\backslash\$$
It won't take the caption	Use $\backslash\text{caption}[short list entry]\{\text{Big huge long caption}\}$ to trim size for listoftables/figures entry
You have nested environments	Are they in First-In, Last-Out order?