

Text Formatting with L^AT_EX

A Tutorial

Academic and Research Computing, April 2007

Table of Contents

1. L^AT_EX Basics	1
1.1 What is T _E X?	1
1.2 What is L ^A T _E X?	1
1.3 How L ^A T _E X Works	2
1.4 The L ^A T _E X Input File	2
1.4.1 Entering L ^A T _E X Commands	2
1.4.2 Entering Text	3
1.4.3 Special Characters	4
1.4.4 Structure of the Input File	4
1.5 Some L ^A T _E X Vocabulary	5
2. Creating A L^AT_EX Document	7
2.1 Document Classes	7
2.2 Class Options	7
2.3 Packages	8
2.4 Making a Title Page	9
2.5 Making a Table of Contents	9
2.6 Behind the Scenes	10
2.6.1 Auxiliary Files	10
2.6.2 How a Page is Built	10
2.7 Example: Report Class	11
2.8 Example: Letter Class	12
3. Document Layout	13
3.1 Line Spacing	13
3.2 Paragraphs	13
3.3 Text Justification	14
3.4 Margins	14
3.5 Headers, Footers, and Page Numbering	15

4. Within the Text	16
4.1 Section Headings	16
4.2 Changing Type Style and Size	17
4.3 Starting New Lines and New Pages	18
4.4 Leaving Horizontal and Vertical Space	18
4.5 Drawing Rules	18
4.6 Footnotes	19
4.7 Centering	19
4.8 Quotations	19
4.9 Reproducing Text As-Is	20
4.10 Lists	21
4.11 Cross References	22
5. Tabular Material	23
5.1 Tabbing	23
5.2 Tabular	24
5.2.1 A Simple Ruled Table	25
5.2.2 Using Paragraph Columns, Spanning Columns	26
5.2.3 Aligning on the Decimal Point	27
5.2.4 Suppressing Leading or Trailing Space	27
6. Mathematics	28
6.1 In-line Math	28
6.2 Display Math (for unnumbered equations)	29
6.3 Equation Environment (for numbered equations)	29
6.4 Eqnarray Environment (for multiline equations)	30
6.5 Array Environment (for matrices, etc.)	31
6.6 Building Mathematical Expressions	32
6.6.1 Superscripts and Subscripts	32
6.6.2 Spaces in Math Mode	32
6.6.3 Dots, Braces, and Bars	32
6.6.4 Fractions	33
6.6.5 Radicals, Integrals, and Summations	33
6.6.6 Large Delimiters	34
7. Including Graphics	35
7.1 Creating the Graphics File	35
7.2 Importing the Graphic into your \LaTeX Document	35
7.3 Viewing the Output	36
8. Placing Figures & Tables (Floats)	37
8.1 Making a Caption	37
8.2 Examples	38
8.3 Overcoming Problems with Float Placement	40
8.4 Landscape Figures and Tables	40

9. Preparing a Bibliography	41
9.1 Using L ^A T _E X's built-in Method	41
9.2 Using BibT _E X	42
9.2.1 Overview	42
9.2.2 Creating the .bib File	43
9.2.3 Running BibT _E X	43
9.2.4 Bibliography Styles	44
9.2.5 For More Information	45
10. Special Topics	46
10.1 Managing a Large Document	46
10.2 Generating an Index	46
10.3 Including hyperlinks	47
10.4 Accents and Special Characters	49
Appendix A: Mathematical Symbols	50
Appendix B: Error Messages	53
Resources	55
General Information on L ^A T _E X	55
L ^A T _E X packages	55
L ^A T _E X Thesis	55
Mathematical Expressions	56
Symbols	56
Graphics	56
BibT _E X	57
Installing L ^A T _E X	57

Text Formatting with L^AT_EX

This document describes the L^AT_EX language. For specifics of how to run it on various platforms (e.g., Windows or UNIX), see the L^AT_EX Information page (<http://www.rpi.edu/dept/arc/training/latex/>), and follow the links for the on-line tutorials. The above web page also contains information on preparing a thesis or a resume, as well as many examples and links to other helpful information.

Chapter 1. L^AT_EX Basics

1.1 What is T_EX?

- T_EX is the typesetting language upon which L^AT_EX is built. It was designed and written by Donald Knuth especially for math and science. T_EX is pronounced “Tech,” similar to “Bach.”
- T_EX is portable. It is available for most computers and is used all over the world. T_EX documents can be moved easily from one system to another, as long as the required fonts are on both systems.
- T_EX comes with its own set of fonts, called “Computer Modern,” used by default. These fonts exist in a variety of styles, including serif, sans serif, typewriter (fixed pitch), and an extensive set of mathematical symbols. It’s also possible to use other font families, such as Times, Palatino, etc.
- T_EX is also a programming language, making it possible to create commands that simplify its use.

1.2 What is L^AT_EX?

- L^AT_EX is a T_EX macro package, originally written by Leslie Lamport, that simplifies the use of T_EX. All the above features of T_EX, including portability, also apply to L^AT_EX. L^AT_EX is pronounced either “Lay-tech” or “Lah-tech.”
- Most L^AT_EX 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, concentrating instead on the content. Most plain T_EX commands also work with L^AT_EX.
- The *document class* determines how the document will be formatted. L^AT_EX provides several standard document classes from which to choose.
- L^AT_EX is flexible, gives you complete control, handles big, complex documents with ease, and never crashes or corrupts your files.

1.3 How L^AT_EX Works

To use L^AT_EX, 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^AT_EX commands to format it. You then typeset your document, usually by clicking a button on a toolbar or selecting a menu item. Nowadays there are two routes for processing a L^AT_EX document:

- The traditional way is to run the `latex` program, which creates a DVI (Device Independent) file. This file is in binary format and not viewed directly. You then run a previewing program for viewing on screen and/or the `dvips` program to create a PostScript file for viewing or for printing via the `ghostview`/`GSView` program. `GSView` can also convert the document to PDF format.
- Alternatively you can run the relatively recent `pdflatex` program to create a PDF file for viewing or printing, usually with Adobe's Acrobat Reader.

The second method is more direct for PDF output, but the first is quicker and more convenient for previewing.

1.4 The L^AT_EX Input File

L^AT_EX input files have names that end with the extension `.tex`: for example, an acceptable file name might be `myfile.tex`. (Never use spaces in file names.) The input file contains both the text of your document and the L^AT_EX commands needed to format it. The first command in the file, `\documentclass`, defines the style of the document.

1.4.1 Entering L^AT_EX Commands

To distinguish them from text, all L^AT_EX commands (also called *control sequences*) start with a backslash “\”. A command name consists of letters only and is ended by a space or a non letter (such as a comma, period, brace, etc). If it ends with a space, the space is “consumed” by L^AT_EX and does not appear in the output. An example is `\today`, which prints the current date. To avoid having the space after the command disappear, do the following:

`\today\` is a good day. produces: April 16, 2007 is a good day.

There are also commands that 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^AT_EX uses it to begin math mode). Spaces after these control symbols are not consumed.

L^AT_EX 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^AT_EX uses *grouping* to limit the effect of certain commands. Braces { ... } are used to begin and end groups. (An *environment* is also a group; see page 5.) 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 provided inside a pair of braces *following* the command name.

The symbol % can be used to put a comment in your input file. When L^AT_EX 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^AT_EX 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 or more blank spaces. Paragraphs are separated by leaving one or more blank lines. (You can also use the command `\par` to indicate a new paragraph.) L^AT_EX 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 L^AT_EX 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 L^AT_EX. 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.

Character	Type in file	Special L^AT_EX meaning
#	<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>\&</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 or an argument
}	<code>\}</code>	Used to end a group or an argument
\	<code>\\backslash\$</code>	Used to begin a control sequence
<	<code>\\<\$</code> (or <code>\textless</code>)	Otherwise, produces <code>;</code>
>	<code>\\>\$</code> (or <code>\textgreater</code>)	Otherwise, produces <code>;</code>

1.4.4 Structure of the Input File

L^AT_EX 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 L^AT_EX 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 L^AT_EX. 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 L^AT_EX 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}
```

this produces:

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. *Here's some italic text* and **some bold text**. The text inside these braces is smaller than normal. Now the text size is back to normal.

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

1.5 Some L^AT_EX 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. Many commands take arguments, either mandatory or optional; some commands, like `\today` don't.

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 are allowed on some commands and are enclosed 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.

Declarations produce neither text nor space, but either affect the way L^AT_EX 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 will be large.}`

Environments are blocks of text that receive special processing. An environment is defined by a matching `\begin{environment name} ... \end{environment name}`. An environment is also a *group*, in the same way that a pair of braces delimits a group. For example, a quotation might be formatted as follows:

```
\begin{quote}
\small
This text is set off from surrounding text and indented from
both margins. The font size of this quotation will be smaller
because of the "small" command inside the quote environment.
\end{quote}
```

Note that a blank line before an environment ends the previous paragraph. A blank line following an environment indicates that the next line starts a new paragraph. Environments can be nested, i.e., the first started is the last ended.

- * Some commands can have a `*` appended to the name, which indicates a variation on a command or environment. For example, `\` indicates a line break. `\``*` indicates a line break with the restriction that L^AT_EX 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*`. Normally, section headings are automatically numbered, but `\section*{My Heading}` will produce an unnumbered section heading.

Chapter 2. Creating A L^AT_EX Document

2.1 Document Classes

The document class determines the overall layout of the document. There are five standard classes distributed with L^AT_EX:

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`, `\subsubsection`, 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 5×7.5 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 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 Office of Graduate Education. It was written by Academic and Research Computing (ARC) using the **report** class as a base. The file `thesis.cls` is available for download from the thesis web page, <http://helpdesk.rpi.edu/update.do?artcenterkey=325>. To use the **thesis** class, begin your document with the line:

```
\documentclass{thesis}
```

Instructions are in ARC document, *Preparing A Thesis with L^AT_EX*, available from the thesis web page.

2.2 Class Options

A document class may be modified by *options*, which are placed in square brackets after the `\documentclass` command. Multiple options are separated by commas:

```
\documentclass[option,option,option]{class}
```

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 document 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 L^AT_EX *packages* available that provide a variety of additional features. Many packages are considered part of L^AT_EX; others are provided by expert users worldwide. If you sometimes find that the features of standard L^AT_EX 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 graphics in your document (see chapter 7.), the package `graphicx` provides the commands to do this. The beginning of your input file might look like:

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

The `\usepackage` command above instructs L^AT_EX to read the file `graphicx.sty`.

In addition to `graphicx` there are many other packages available, including packages to rotate text, use PostScript fonts (e.g., Times, Palatino, etc), and customize such things as headers and footers, citations and captions. Packages come with their own documentation. Many packages are routinely distributed with \LaTeX and will already be on your system. You'll find documentation for these packages in the `doc` subdirectory of your installation. For example, if you are using the TeXLive 2005 distribution for Windows, package documentation is in folders under `C:\TeXLive2005\texmf-dist\doc\latex\`.

To see a current list of all available packages with brief descriptions, look at the Comprehensive TeX Archive Network (CTAN) catalog, <http://texcatalogue.sarovar.org/brief.html>.

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 \LaTeX to generate the title page with the command `\maketitle`. In your \LaTeX input file, type commands such as the ones below. The `\maketitle` command, which actually prints the title page, must follow the `\begin{document}` command; the other commands, which just provide the information, can be placed in the preamble if you prefer.

```
\title{This is the Title} % provide title info
\author{My Name}         % provide author info
\date{the date}          % provide date
\maketitle                % format and print title page
```

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, \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 `documentclass` 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 \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 L^AT_EX is its ability to do cross 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 L^AT_EX is first processing the input file. L^AT_EX stores this information in auxiliary files as it processes the job. A second run allows L^AT_EX to extract the information from its auxiliary files and complete the table of contents, etc. Therefore *all information used by L^AT_EX 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 L^AT_EX twice before printing.

The auxiliary files have the same “root” name as the L^AT_EX 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. Auxiliary files are created automatically 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 T_EX or L^AT_EX 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 T_EX works.

2.7 Example: Report Class

```

\documentclass[11pt]{report}           % Report class in 11 points
\raggedright                           % Do not right-justify
\parindent0pt \parskip8pt             % make block paragraphs

\begin{document}                       % End of preamble, start of
                                        % document text.
\title{\bf An Example of Report Class} % Supply information
\author{Yours Truly}                   % 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}                % Make 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. Also, article class does not have a ‘‘chapter’’ command.

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

```

\section{A Subheading}                  % Make a "section" heading
The following sectioning commands are available:
\begin{quote}                            % Start "set off", indented text
part \\\                                 % "\\" forces a new line
chapter \\\                               % not available in article class
section \\\
subsection \\\
subsubsection \\\
paragraph \\\
subparagraph
\end{quote}                              % End of indented text

```

The `*-form` (e.g., ‘‘`section*`’’) suppresses the section number and does not make a TOC entry.

```

\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 preamble¹.

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 current font. (1 em is about the width of an “M”.) 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=8pt % this is variable, choose the number you want
```

¹the section between the `\documentclass` command and the `\begin{document}` command

3.3 Text Justification

By default, L^AT_EX 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

Changing default margins, which depend on the class and the font size, is not as easy as you might think. The best way to control margins is to use the `geometry` package. This package has many options and extensive documentation in `manual.pdf` found in your directory `... \doc \latex \geometry \`. The following examples should be obvious:

```
\usepackage[margin=1in]{geometry} % 1 inch margins all around
\usepackage[left=1.2in,right=1in,top=1in,bottom=.8in]{geometry}
```

If you really want to do it manually, you need to know that internally 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 `topmargin` to `-.5in` produces a top margin of .5 inches. The opposite margins (bottom and right) are determined indirectly by setting the height (`textheight`) and width (`textwidth`) of the text area.

<u>to set margin</u>	<u>change 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>

L^AT_EX 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 values for `topmargin` and `textheight`. The values below leave one inch between the paper edge and the text on all four sides.

```
\setlength{\topmargin}{-.5in} % top margin is .5 in
\setlength{\oddsidemargin}{0in} % left margin is 1 in on right pages
\setlength{\evensidemargin}{0in} % same for left pages, 2-sided document
\setlength{\textwidth}{6.5in} % leaves 1 in for right margin
\setlength{\textheight}{9in} % 9 inches reserved for the text
```

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 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{leftheader}{rightheader}
\markright{rightheader}
```

`\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{}`. To suppress the horizontal line drawn by default under the header, use `\renewcommand{\headrulewidth}{0pt}`. For more information, see `fancyhdr.pdf` in your directory `... \doc \latex \fancyhdr \`.

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>	

Note that the `\chapter` command is 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 by default.

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^AT_EX 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>	boldface	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 document 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^AT_EX 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` documentclass 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 L^AT_EX. 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.²

4.7 Centering

If you have only one line to center, it's easiest to use the plain T_EX command `\centerline`; for example,

```
\centerline{This line will be 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}
```

There is also the declaration `\centering`, which is always used within a group—either a pair of braces or an environment. This is useful when you don't want the extra vertical space surrounding the `center` environment.

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.

²This 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 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 if you want to print 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 here to delimit the verbatim text. Any character except the “*” can be used as the delimiter.

4.10 Lists

The three L^AT_EX list environments all 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.

The following examples show both the output and the input used to create them.

Example of Enumerate

```
\begin{enumerate}
  \item Sugar
  \item Cream
  \item Chocolate
\end{enumerate}
```

1. Sugar
2. Cream
3. Chocolate

Example of Itemize

```
\begin{itemize}
  \item Mix all ingredients together.
  \item Boil until the thermometer
    reaches 112  $^{\circ}$ C.
  \item Stir and cool.
\end{itemize}
```

- Mix all ingredients together.
- Boil until the thermometer reaches 112 °C.
- Stir and cool.

Example of Description

```
\begin{description}
  \item[dog] A loving animal that
    likes to sleep on the furniture.
  \item[cat] Aloof creature that can
    warm your feet on a winter's night
  \item[horse] Large animal, gives
    great rides. Eats a lot, luckily
    doesn't sleep on the furniture.
\end{description}
```

dog A loving animal that likes to sleep on the furniture.

cat Aloof creature that can warm your feet on a winter's night

horse Large animal, gives great rides. Eats a lot, luckily doesn't sleep on the furniture.

Below is an example of nesting list environments:

```
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}
```

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)

4.11 Cross References

In longer documents, there are often cross references to sections, figures, tables, or equations. L^AT_EX 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 L^AT_EX'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~\ref{sec:xrefs} on page~\pageref{sec:xrefs}.	For information on cross references, see section 4.11 on page 22 .

Note that L^AT_EX 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.

Chapter 5. Tabular Material

There are two environments in \LaTeX for formatting tabular material: 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 tabbed material 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 \LaTeX reads in the entire table at once to establish correct column widths. Often material inside a `tabular` environment is placed inside the `table` environment, which ensures that if it can't fit at the current location it will be “floated” to an appropriate location. For information on using the `table` environment, see Chapter 8.

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 % draw horizontal line
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 chapter 8. on page 37) 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. \\
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 \LaTeX 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.³

\LaTeX 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      &
\multicolumn{2}{c}{Value} \\
\hline
&  $\pi$  & 3.1416
 $\pi$  &  $\pi^\pi$  & 36.46
 $\pi^{\pi}$  &  $(\pi^\pi)^\pi$  & 80662.7
 $(\pi^{\pi})^{\pi}$  & &
\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}
```

³For an another method, see the `dcolumn` package, provided as part of the \LaTeX “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. The `amssymb` package provides additional mathematical symbols; the `amsmath` package provides additional environments for building mathematical expressions. For help with using AMS \LaTeX , see *The Short Math Guide for \LaTeX* , at: <ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf>. The official documentation is `amslatex.pdf`, which you can find on your system in `... \doc \latex \amslatex \`.

6.1 In-line Math

\LaTeX 's 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.

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 inline 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 \quad \nonumber \\
& & = & a^2 + 2ab + b^2 \quad \label{eq:simp1} \\
(a + b)(a - b) & = & a^2 - ab + ba - b^2 \quad \nonumber \\
& & = & a^2 - b^2 \quad \label{eq:simp2} \\
(a + b)^3 & = & a^3 + 3a^2b + 3ab^2 + b^3 \quad \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](#).

The `amsmath` package provides additional environments for formatting multiline equations (e.g., `split`, `align`, `gather`, etc) and is highly recommended.

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 5×5 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:

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

Section 6.6.6 (Large Delimiters) on page 34 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:

```
\[ 2^{2^2} = 2^4 = 4^2 \]
\[ a^2_{i_1} = b^2_{i,j} \]
\[ {}_2F^1_3 \]
\[ {}_2^1P^3_4 \]
```

produces:

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

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

$${}_2F^1_3$$

$${}_2^1P^3_4$$

6.6.2 Spaces in Math Mode

In math mode, L^AT_EX 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 `\qquad` 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}^{\mbox{n terms}} \\
k \cdot x & = & \underbrace{x + x + \cdots + x}_{\mbox{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 plain TeX commands `\atop` and `\choose`.

```


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


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


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


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


```

produces:

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

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

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

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, *n*th 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:

```


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


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


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


```

produces:

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

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

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

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 31.)

```
\[
\mbox{det} = \left|
\begin{array}{l}
c_0 & c_1 & c_2 & \dots & c_n \\
c_1 & c_2 & c_3 & \dots & c_{n+1} \\
c_2 & c_3 & c_4 & \dots & c_{n+2} \\
\vdots & \vdots & \vdots & \& \vdots \\
c_n & c_{n+1} & c_{n+2} & \dots & c_{2n}
\end{array}
\right.
\right| > 0.
\]
```

produces:

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

```
\[
\mbox{signum}(x) = \left\{
\begin{array}{rl}
1 & \mbox{if } \$x > 0\$ \\
0 & \mbox{if } \$x = 0\$ \\
-1 & \mbox{otherwise}
\end{array}
\right.
\right.
\]
```

produces:

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

Chapter 7. Including Graphics

7.1 Creating the Graphics File

The easiest way to put graphics into your \LaTeX document is to first create the graphic using a software package, such as Maple, Matlab, CorelDRAW, Xfig, Gnuplot, etc. You can also use a Windows application such as Word or Excel to create an eps (Encapsulated PostScript) file by printing to a file following certain steps, described in <http://www.rpi.edu/dept/arc/training/latex/Examples/graphics.pdf>.

If you are using \LaTeX plus `dvips` to produce your final output, you should save the graphic as encapsulated PostScript (eps), the only acceptable format. If you are using `pdf \LaTeX` (which produces a pdf file directly), acceptable formats are pdf, jpeg, or png, but *not* eps. A general rule of thumb for graphics formats is to use pdf or eps for vector graphics (line art, graphs, etc), jpeg for photographs, and png for screen shots.

If you are using `pdf \LaTeX` and your graphics files are in eps format, you can convert them to pdf using the `epstopdf` utility, which is most likely on your system. Run this program from the command line (in Windows, open a command window and `cd` to the appropriate folder):

```
epstopdf myfigure.eps      generates myfigure.pdf
```

After doing this, you will have two files for your graphic, one eps and one pdf.

If you are using \LaTeX plus `dvips`, the `jpeg2ps` utility is handy. It converts JPEG images to eps files without uncompressing the images. The JPEG data is simply “wrapped” with PostScript, yielding considerably smaller PostScript files. Use it from the command line:

```
jpeg2ps -h image.jpg > image.eps
```

You may already have `jpeg2ps` on your system; if not, you can download it from: <http://www.ctan.org/tex-archive/support/jpeg2ps/>.

7.2 Importing the Graphic into your \LaTeX Document

LaTeX’s standard graphics bundle includes two packages for importing graphics: `graphics` and `graphicx`. The extended package, `graphicx`, provides a more convenient method of supplying parameters and is recommended. Therefore first step is to put in your preamble the command:

```
\usepackage{graphicx}
```

This package defines a new command called `\includegraphics`, which allows you to specify the name of the graphic file as well as supply optional arguments for scaling or rotating. So, at the spot you want to insert the graphic (named for example, `myfigure.eps` or `myfigure.pdf`) use the `\includegraphics` command, for example:

```
\includegraphics[width=4in]{myfigure}    note filename extension is omitted
```

If you want to be able to process your file using either `latex` or `pdflatex`, you'll need to have your graphics files in both eps format and one of the others, such as pdf. Then omit the filename extension on the `\includegraphics` command: `latex` will look for an eps file, and `pdflatex` will look for a pdf, jpg, or png file.

The `\includegraphics` command also provides optional arguments for scaling or rotating the figure. Inside the `[...]`, you can specify a number of optional arguments, separated by commas. In the example above, `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:

width scale graphic to the specified width (aspect ratio preserved)
height scale graphic to the specified height (aspect ratio preserved)
scale scale graphic by scale factor. (`scale=2` makes the graphic twice as large as its natural size; `scale=.5` makes it half as large.)
angle rotate graphic by specified number of degrees. A positive number indicates the counter-clockwise direction. (`angle=90` 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 that will just fit within the margins:

```
\includegraphics[angle=90,width=\textwidth]{myfigure}
```

Usually the `includegraphics` command is put inside the `figure` environment, which allows the image to move or “float” to another location so that it is never split between pages. The `figure` environment is described in chapter 8.

7.3 Viewing the Output

If your graphics files are in pdf, jpeg or png formats, you can run `pdfLATEX` and view the result with Acrobat or GSView. If your files are eps, it is important to note that after running `LATEX`, the resulting `.dvi` file does not actually contain the eps graphics; it only contains commands that the printer driver, `dvips`, subsequently uses to include the graphics when making the PostScript file. Most dvi previewers try to display eps files (usually by calling on `ghostscript`), but rotated material may not display properly in the dvi preview. To see a correct display, run `dvips` to make a PostScript file and then use `GSView/ghostview` to view it.

Complete documentation for the graphics bundle is in the file `grfguide.pdf`; look for it on your system. Information on the `graphicx` package is in section 4.4. For an extensive treatment of including graphics, see *Using Imported Graphics in L^AT_EX and pdfL^AT_EX* by Keith Reckdahl of Stanford University. It includes all you would ever want to know with many examples: <http://www.ctan.org/tex-archive/info/epslatex.pdf>.

Chapter 8. Placing Figures & Tables (Floats)

If you have figures or tables in your document, you'll want to make sure that they are not broken across pages. To accomplish this, \LaTeX provides two environments, `figure` and `table`, which move or “float” the material inside the environment to a location where it all fits. Meanwhile, \LaTeX fills the current page with text to avoid half empty pages. The commands to use these two environments look like:

```
\begin{figure}                \begin{table}
... figure material ...      ... tabular material ...
\end{figure}                 \end{table}
```

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 tabular material (discussed in chapters 5. and 7.); they only ensure that whatever material is inside the environment gets “floated” and is not broken between pages.

To place the floated material, \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.

8.1 Making a Caption

The floated material usually includes a numbered caption. Within the figure or table environment, you can supply a caption with the command `\caption{caption text}`. 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.

8.2 Examples

Note that you can put anything inside the figure environment: it can be text, a diagram, an image (see chapter 7.) or any other kind of illustration. Likewise, a table environment can contain any \LaTeX commands or text, but usually contains tabular material inside the `tabular` environment (see section 5.2).

The example below encloses tabular material in the `table` environment, which allows it to float to an appropriate spot in the text. This table is floated to the top of the next page because there is not enough room for it to be placed “here”. Note the placement of the `\caption` and `\label` commands in this example and the example figure on the next page.

```
\begin{table}[htp]
\caption{Contract Expenses to Date}
\label{tab:exp}
\begin{center}
\begin{tabular}{lr}
Item & Amount \\
\hline
Salaries (research assistants, secretary) & \$24,000 \\
Travel expenses & \$8,000 \\
Software & \$2,000 \\
\cline{2-2} \[-8pt]
Total & \$34,000
\end{tabular}
\end{center}
\end{table}
```

Table 2: Contract Expenses to Date

Item	Amount
Salaries (research assistants, secretary)	\$24,000
Travel expenses	\$8,000
Software	\$2,000
Total	\$34,000

The following example produces a figure that uses a line drawing of a cat, which \LaTeX places “here” because there happens to be room for it. There are two versions of the image, `cat.eps` and `cat.pdf`, so that either \LaTeX or `pdf \LaTeX` can be used. Note that the `\includegraphics` command below does not specify an extension on the file name, leaving it up to \LaTeX or `pdf \LaTeX` to find the appropriate file.

```
\begin{figure}[htp]
  \centering
  \includegraphics[angle=30,width=.6\textwidth]{cat}
  \caption{A Topsy Cat}
  \label{fig:cat}
\end{figure}
```

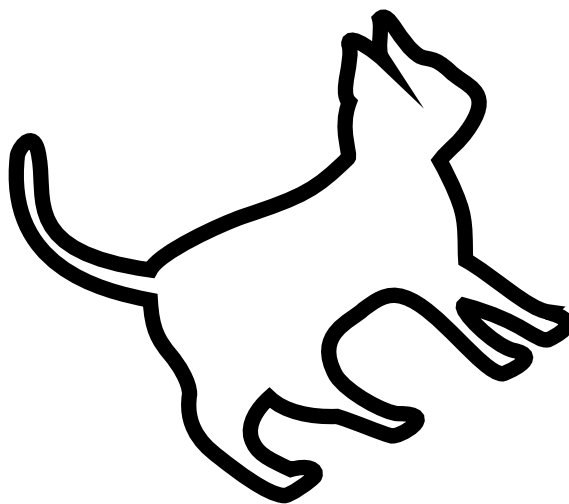


Figure 1: A Topsy Cat

8.3 Overcoming Problems with Float Placement

When placing floated material, L^AT_EX follows certain rules, explained in detail in Lamport's *L^AT_EX 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 L^AT_EX'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]`, L^AT_EX 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 could be pushed to the end of the section or document with all subsequent figures or tables behind it.

L^AT_EX 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.

8.4 Landscape Figures and Tables

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

```
\usepackage{rotating}
```

This package defines two new environments, `sidewaysfigure` and `sidewaystable`, which you then use in place of the standard L^AT_EX environments `figure` and `table`. The sideways environments always put the landscape table or figure on a page by itself. For example, to make a landscape page containing a rotated graphic and caption, use commands such as:

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

The file <http://www.rpi.edu/dept/arc/training/latex/Examples/exrotating.tex> contains examples of using both the `graphicx` and the `rotating` packages to include landscape figures and tables. To see the results after running `pdflatex`, view the output in the file

<http://www.rpi.edu/dept/arc/training/latex/Examples/exrotating.pdf>.

Full documentation for the `rotating` package is in the book, *The L^AT_EX Companion* by Mittlebach and Goossens.

Chapter 9. Preparing a Bibliography

\LaTeX provides built-in support for citing references, which is straightforward and relatively easy to use. However, it's not very flexible. If your bibliography is large or if you use the same references in several documents, you may prefer to use `BibTeX`, a companion program to \LaTeX that allows you to store all your references in an external database file. You can then refer to this database in your \LaTeX document, and cite any reference within it. Both methods are described below.

9.1 Using \LaTeX 's built-in Method

To prepare a bibliography the “standard” way, 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 automatically generated in the bibliography. (Remember to run \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 format the bibliography, sorting the entries as you wish them to appear, 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 “9” indicates there are fewer than 10 entries. If there are 10 or more 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{lamport} Leslie Lamport. \textit{\LaTeX\ -- A Document
    Preparation System}. Addison-Wesley, second edition, Reading, MA, 1994.
  \bibitem{kopka} Helmut Kopka and Patrick W.~Daly. \textit{A Guide
    to \LaTeX}. Addison-Wesley, fourth edition, Boston, MA, 2004.
  \bibitem{goossens} Michel Goossens, Frank Mittelbach, et al. \textit{The
    \LaTeX\ Companion}. Addison-Wesley, second edition, Boston, MA, 2004.
\end{thebibliography}
```

References

- [1] Leslie Lamport. *L^AT_EX – A Document Preparation System*. Addison-Wesley, Reading, MA, second edition, 1994.
 - [2] Helmut Kopka and Patrick W. Daly. *A Guide to L^AT_EX*. Addison-Wesley, fourth edition, Boston, MA, 2004.
 - [3] Michel Goossens, Frank Mittelbach et al. *The L^AT_EX Companion*. Addison-Wesley, second edition, Boston, MA, 2004.
-

If you want to change the heading for the reference listing, just change the value of `\refname` (for article class) or `\bibname` (book or report class) by adding a line like this in your preamble: `\renewcommand{\bibname}{Literature Cited}`.

If you like, you can choose a label to identify the work in both the text and in the bibliography, instead of the number assigned by L^AT_EX. 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, et al.  
  \textit{The LATeX Companion}. Addison-Wesley, second edition,  
  Boston, MA, 2004.
```

9.2 Using BibT_EX

9.2.1 Overview

BibT_EX automatically generates a list of references from information contained in a bibliographic database—a file you create whose name ends with the extension `.bib`. The entries in your `.bib` file are called in your L^AT_EX file with various `\cite` commands, just as in the built-in method described above. In your L^AT_EX file, you provide just two more pieces of information: A `bibliographystyle` command to specify the formatting style of the citations and the bibliography (this can appear anywhere in your L^AT_EX file), and at the point you want the bibliography printed, a `bibliography` command to provide the name of the `.bib` file.

After processing your document through L^AT_EX (or pdfL^AT_EX) you run BibT_EX, which will format the citations according to the style you've specified. You then need to run L^AT_EX (or pdfL^AT_EX) again to process the output created by BibT_EX, and finally once more to resolve the references. Your resulting output will then contain your citations and formatted bibliography.

9.2.2 Creating the .bib File

The .bib file is a plain text file that you create and edit with any text editor. The following example of a BibTeX entry illustrates the structure. Each entry begins with the document type preceded by the “@” sign. This is followed by a unique label that you make up and that you will use to cite the entry in your text. Then each field (in any order) uses the format keyword = {data}. Note that the data can be enclosed in either quotation marks or braces, and each line except the last must end with a comma.

```
@article{Hefferon04,
  author = "Jim Hefferon",
  title = "{CTAN} for {S}tarters",
  year = "2004",
  journal = "{TUGB}oat",
  volume = "25",
  number = "2",
  pages = "126--127"
}
```

The above example refers to an article within a journal. There are many different document types: book, article (in a journal), article (in a collection), chapter (in a book), thesis, report, paper (in a Proceedings), etc. There is a prescribed set of fields for each type of document. A book entry might look like:

```
@book{Kopka04,
  author = {Helmut Kopka and Patrick W.~Daly},
  title = {Guide to {\LaTeX}},
  edition = {fourth},
  publisher = {Addison-Wesley},
  address = {Boston, MA},
  year = {2004}
}
```

If a work has more than one author, separate them with the keyword “and” so BibTeX can distinguish the different names. If the BibTeX style does not preserve all your capitalization (for example within titles), you can tell BibTeX to keep them by using braces around the letters in question.

9.2.3 Running BibTeX

After processing your file with L^AT_EX (or pdfL^AT_EX), you run the BibTeX program by clicking on the BibTeX toolbar icon (if your editor has one), or by typing the command `bibtex` followed by the name of your document *without the .tex extension*. The information for every document you have cited will be extracted from your database, formatted according to the style you specify, and stored in a bibliographic (.bbl) file ready for L^AT_EX to use. When you run L^AT_EX again it will build the bibliography from the .bbl file and generate warnings about unresolved references. Subsequent runs will resolve the references, and the warnings will disappear.

Suppose you have a file called `myrefs.bib` that contains the two bibliographic entries above, and the file `testbib.tex` contains the following:

```
\documentclass{article}
\begin{document}
We are testing the use of Bib\TeX\ using two entries.
One is a book \cite{kopka04}, and the other is an
article in a journal. \cite{hefferon04}
\bibliographystyle{plain} % use style plain.bst
\bibliography{myrefs}     % find references in myrefs.bib
\end{document}
```

The result of running \LaTeX , then $\text{Bib}\TeX$ then \LaTeX twice more would be:

We are testing the use of $\text{Bib}\TeX$ using two entries. One is a book [2], and the other is an article in a journal. [1]

References

- [1] Jim Hefferon. CTAN for Starters. *TUGBoat*, 25(2):126–127, 2004.
 - [2] Helmut Kopka and Patrick W. Daly. *Guide to \LaTeX* . Addison-Wesley, Boston, MA, fourth edition, 2004.
-

9.2.4 Bibliography Styles

Bibliography styles are files (with the extension `.bst`), that tell $\text{Bib}\TeX$ how to format the information in the `.bib` file. The standard styles are:

- `plain` entries are sorted alphabetically and are labelled with numbers.
- `unsrt` the same as `plain` except entries appear in the order in which they were cited.
- `abbrv` the same as `plain` except names, month names, journal names are abbreviated.
- `alpha` entries are sorted alphabetically, and the labels are made from the author’s or authors’ (or editor’s) name and the year of publication.

The `.../bst` subdirectory of your \TeX installation contains the styles (`.bst` files) installed by default, and the *The \LaTeX Companion*, chapter 12, describes others.

There are also various \LaTeX packages that provide other styles, such as author-date citations instead of numbers. To use one of these packages, you need to include its name with the normal `\usepackage` command in your preamble. One highly recommended and flexible package is `Natbib`. In addition to loading the `Natbib` package in the preamble, you need to use a bibliography style that supports the author-date style. (Some possible choices are “`plainnat`”, “`unsrtnat`”, “`abbrvnat`” and “`apalike`”.) Also, `Natbib` uses slight variations of the `\cite` command: you just append a “`t`” (for textual) or “`p`” (for parenthetical), to the basic `\cite` command, for example:

Citation command	Natbib output
<code>\citet{hefferon04}</code>	Hefferon (2004)
<code>\citep{hefferon04}</code>	(Hefferon, 2004)
<code>\citet[chapter 4]{kopka04}</code>	Kopka and Daly (2004, chapter 4)
<code>\citep[see][chap.~4]{kopka04}</code>	(see Kopka and Daly, 2004, chap. 4)

Using the same `myrefs.bib` file as in the previous example, and changing the `testbib.tex` file to read:

```

\documentclass{article}
\usepackage{natbib}
\begin{document}
We are testing the use of BibTeX using two entries.
One is a book by \citet{kopka04}, and the second is an
article in a journal \citep{hefferon04}.
\bibliographystyle{apalike} % use style apalike.bst
\bibliography{myrefs}      % find references in myrefs.bib
\end{document}

```

produces, after running L^AT_EX, then BibTeX, then L^AT_EX twice more:

We are testing the use of BibTeX using two entries. One is a book by Kopka and Daly (2004); the second is an article in a journal (Hefferon, 2004).

References

- Hefferon, J. (2004). CTAN for Starters. *TUGBoat*, 25(2):126–127.
- Kopka, H. and Daly, P. W. (2004). *Guide to L^AT_EX*. Addison-Wesley, Boston, MA, fourth edition.
-

9.2.5 For More Information

For more information on L^AT_EX's built-in method and on using BibTeX, see any of the three books in section 9.1.

For on-line information on BibTeX, there is an excellent tutorial by Andrew Roberts: <http://www.andy-roberts.net/misc/latex/latextutorial3.html>

A longer, more complete document covering both LaTeX's basic bibliography method and BibTeX is *Tame the BeaST* by Nicolas Markey: http://tug.ctan.org/tex-archive/info/bibtex/tamethebeast/ttb_en.pdf

An excellent, concise overview of the Natbib package is the file `natnotes.pdf`, and the Natbib manual is in `natbib.pdf`. You can find both these files at: <http://www.ctan.org/tex-archive/macros/latex/contrib/natbib/>

Chapter 10. Special Topics

10.1 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 \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 \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}
```

10.2 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, `makeindex.dvi`, which should be already on your system under the `doc` folder. For example, if you are using TeXLive 2005 on Windows, you'll find the documentation in:

```
C:\TeXLive2005\texmf-dist\doc\makeindex\base\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 running `LATEX` once more. If you don't have a "makeindex" button on the toolbar of your editor/shell, you can run it from the command line or use the editor's Help feature to add it to the toolbar. Below is an example of doing the 3-step process from the command line. Note that you should not include the filename extensions on your commands; the programs will then do the right thing.

```
latex (or pdflatex) myfile  In addition to producing the .dvi or pdf file, this generates
                             the file myfile.idx, which contains all the index entries
                             and page numbers.

makeindex myfile           MakeIndex processes myfile.idx and produces the file
                             myfile.ind, which contains the LATEX commands to for-
                             mat the index.

latex (or pdflatex) myfile  When LATEX or pdfLATEX finds the \printindex com-
                             mand, it reads in the file myfile.ind. The resulting out-
                             put 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

<u>L^AT_EX Input</u>	<u>Index Entry</u>	<u>Comments</u>
<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, 25	Formatted page number

10.3 Including hyperlinks

Both `LATEX` and `pdfLATEX` can use the `hyperref` package to place live links in the PDF file. By including `\usepackage{hyperref}` as the last package in your preamble, you will automatically get bookmarks corresponding to your sections, and hyperlinks from your tables of contents and `\cite` and `\ref` references. The package has a number of options; some of the most useful are:

colorlinks makes the text of the link colored instead of framed with a box. It's also possible to choose the color of the links with the `linkcolor`, `urlcolor`, and `citecolor` options.

plainpages=false distinguishes between frontmatter and mainmatter page numbers. With this option, `hyperref` writes different anchors for pages “ii” and “2”.

pdfpagelabels sets PDF page labels so that Acrobat Reader displays the page number as (say) “iv (4 of 40)” rather than simply “4 of 40”. `plainpages=false` and `pdfpagelabels` are usually used together.

breaklinks allows a line break in a long link (such as a TOC entry). *Works with pdfL^AT_EX only.*

linktocpage makes the page number, not the text, the link in TOC, LOF, and LOT. It's especially useful if you use L^AT_EX and `dvips` to get your PDF file, as this method cannot use `breaklinks`. If you also have long URLs in the body of the thesis, the `breakurl` package can help. For documentation, look on your system for the file `breakurl.pdf`.

The `hyperref` package also provides the command `\url{URL}` to link to a URL from the text, for example: `\url{http://www.ctan.org/}` or `\url{mailto:email@email.com}`. The command `\href{URL}{text}` provides a more sophisticated method, where “URL” is the address and “text” is the text displayed in the document. For example, to avoid having the text “mailto:” appear in the document, use `\href{mailto:myfriend@rpi.edu}{myfriend@rpi.edu}`.

Be sure to run L^AT_EX or pdfL^AT_EX twice to ensure correct hyperlinks.

Abbreviated example file:

```
\documentclass{report}
\usepackage{ifpdf} % to use same .tex file for both latex & pdflatex
% the following specifies different options to hyperref depending on
% whether latex or pdflatex is being run.
\ifpdf
  \usepackage[colorlinks,linkcolor=blue,urlcolor=blue,citecolor=blue,
  plainpages=false,pdfpagelabels,breaklinks]{hyperref}
\else
  \usepackage[colorlinks,linkcolor=blue,urlcolor=blue,citecolor=blue,
  plainpages=false,pdfpagelabels,linktocpage]{hyperref}
\fi
\begin{document}
  ... body of your document goes here ...
\end{document}
```

The official documentation for `hyperref` is `manual.pdf`. Look for it on your system in `.../doc/latex/hyperref/`.

10.4 Accents and Special Characters

L^AT_EX 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>	ô	<code>\r {o}</code>
ö	<code>\u {o}</code>	õ	<code>\v {o}</code>	ő	<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>	L	<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 AMS \LaTeX package, `amssymb`, provides additional math symbols. For a list, see <http://www.rpi.edu/dept/arc/training/latex/amssymblist.pdf>.

Greek Alphabet

α	<code>\alpha</code>	ι	<code>\iota</code>	ϱ	<code>\varrho</code>
β	<code>\beta</code>	κ	<code>\kappa</code>	σ	<code>\sigma</code>
γ	<code>\gamma</code>	λ	<code>\lambda</code>	ς	<code>\varsigma</code>
δ	<code>\delta</code>	μ	<code>\mu</code>	τ	<code>\tau</code>
ϵ	<code>\epsilon</code>	ν	<code>\nu</code>	υ	<code>\upsilon</code>
ε	<code>\varepsilon</code>	ξ	<code>\xi</code>	ϕ	<code>\phi</code>
ζ	<code>\zeta</code>	o	<code>o¹</code>	φ	<code>\varphi</code>
η	<code>\eta</code>	π	<code>\pi</code>	χ	<code>\chi</code>
θ	<code>\theta</code>	ϖ	<code>\varpi</code>	ψ	<code>\psi</code>
ϑ	<code>\vartheta</code>	ρ	<code>\rho</code>	ω	<code>\omega</code>
Γ	<code>\Gamma</code>	Ξ	<code>\Xi</code>	Φ	<code>\Phi</code>
Δ	<code>\Delta</code>	Π	<code>\Pi</code>	Ψ	<code>\Psi</code>
Θ	<code>\Theta</code>	Σ	<code>\Sigma</code>	Ω	<code>\Omega</code>
Λ	<code>\Lambda</code>	Υ	<code>\Upsilon</code>		

Capitals not shown are produced with English capital letters.

¹An 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>	∇	<code>\nabla</code>	\neg	<code>\neg</code>
\jmath	<code>\jmath</code>	\surd	<code>\surd</code>	\flat	<code>\flat</code>
ℓ	<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>
∂	<code>\partial</code>	\triangle	<code>\triangle</code>	\heartsuit	<code>\heartsuit</code>
∞	<code>\infty</code>	\diamond	<code>\diamond</code>	\spadesuit	<code>\spadesuit</code>
\mho	<code>\mho</code>	\square	<code>\square</code>	\backslash	<code>\backslash</code>
\sum	<code>\sum</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 \LaTeX 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>\lfloor</code>	⌋	<code>\rfloor</code>
[[]]	⌈	<code>\lceil</code>	⌉	<code>\rceil</code>
{	<code>\{</code>	}	<code>\}</code>	⟨	<code>\langle</code>	⟩	<code>\rangle</code>
			<code>\ </code>	↑	<code>\uparrow</code>	↗	<code>\Uparrow</code>
/	/	\	<code>\backslash</code>	↓	<code>\downarrow</code>	↘	<code>\Downarrow</code>
				↕	<code>\updownarrow</code>	↕	<code>\Updownarrow</code>

Relational Operators

\leq	<code>\le</code> or <code>\leq</code>	\geq	<code>\geq</code>	\equiv	<code>\equiv</code>	<code>\equiv</code>	<code>\equiv</code>
\prec	<code>\prec</code>	\succ	<code>\succ</code>	\sim	<code>\sim</code>	\sim	<code>\sim</code>
\preceq	<code>\preceq</code>	\succeq	<code>\succeq</code>	\simeq	<code>\simeq</code>	\simeq	<code>\simeq</code>
\ll	<code>\ll</code>	\gg	<code>\gg</code>	\asymp	<code>\asymp</code>	\asymp	<code>\asymp</code>
\subset	<code>\subset</code>	\supset	<code>\supset</code>	\approx	<code>\approx</code>	\approx	<code>\approx</code>
\subseteq	<code>\subseteq</code>	\supseteq	<code>\supseteq</code>	\cong	<code>\cong</code>	\cong	<code>\cong</code>
\sqsubset	<code>\sqsubset</code>	\sqsupset	<code>\sqsupset</code>	\perp	<code>\perp</code>	\perp	<code>\perp</code>
\sqsubseteq	<code>\sqsubseteq</code>	\sqsupseteq	<code>\sqsupseteq</code>	\bowtie	<code>\bowtie</code>	\bowtie	<code>\bowtie</code>
\in	<code>\in</code>	\ni	<code>\ni</code>	\propto	<code>\propto</code>	\propto	<code>\propto</code>
\vDash	<code>\vDash</code>	\dashv	<code>\dashv</code>	\models	<code>\models</code>	\models	<code>\models</code>
\smile	<code>\smile</code>	\mid	<code>\mid</code>	\doteq	<code>\doteq</code>	\doteq	<code>\doteq</code>
\frown	<code>\frown</code>	\parallel	<code>\parallel</code>				
$\not<$	<code>\not<</code>	$\not>$	<code>\not></code>	\neq	<code>\neq</code>	\neq	<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>	\lhd	<code>\lhd</code>
\triangleleft	<code>\unlhd</code>	\triangleright	<code>\unrhd</code>	\rhd	<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>	\hookleftarrow	<code>\hookleftarrow</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>
\rightrightarrows	<code>\rightrightarrows</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.

Responding to Errors

When L^AT_EX finds an error, it generates a message such as:

```
! Undefined control sequence.
1.9 \section
      {Introduction}
?
```

This means the `\section` command was misspelled, and the error occurred on line 9 of the input file.

You can respond with:

h for help

x for exit

press the **Return** key to ignore it and hope for the best. L^AT_EX will try to recover from the error and continue processing. Sometimes you can tell from viewing the output what went wrong.

If it stops with a `*` prompt, it often means you have forgotten `\end{document}`. Enter it at the prompt (and fix the file later).

If you mistyped the file name or for some other reason L^AT_EX cannot find a file, it will ask for another filename. If you don't want to enter a new filename, quit the program by typing **Ctrl-d** or **Ctrl-c**. Another handy "Emergency stop sequence" is **Ctrl-z**.

Windows users: *Do not simply close the Command window without responding!*

The window may disappear but L^AT_EX is still running, which will cause confusion when you run it again.

Some Common Errors

- A command was misspelled and L^AT_EX doesn't recognize it.
- You have a special character (e.g., `&`, `$`, `#`, `%`) in the text. Enter these as `\&`, `\$`, `\#` or `\%`.
- Whatever you `\begin{...}`, you must `\end{...}`. This includes braces: `{ ... }`.
- You inserted too many tabs on the line before the `\.`
- A command wasn't given all its arguments and is trying to use the rest of the file for an argument.

Errors About Boxes

Another common error (actually just a warning) refers to an `underfull hbox` (or `vbox`). These errors are harmless and may be safely ignored. They 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. \LaTeX will generate this warning 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 \textbf{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
```

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 or figures	Put <code>\clearpage</code> 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 <code>\</code> or have too many tabs on a line?
It won’t take the caption	Use <code>\caption[short list entry]{Very long caption}</code> to trim size for listoftables/figures entry
You have nested environments	Are they in First-In, Last-Out order?

Resources

General Information on L^AT_EX

Books

L^AT_EX – A Document Preparation System by Leslie Lamport. Addison-Wesley, second edition, Reading, MA, 1994.

A Guide to L^AT_EX by Helmut Kopka and Patrick W. Daly. Addison-Wesley, fourth edition, Boston, MA, 2004.

The L^AT_EX Companion by Michel Goossens, Frank Mittelbach, et al. Addison-Wesley, second edition, Boston, MA, 2004.

On line

For on-line tutorials, example files, and links to other information, see the Rensselaer L^AT_EX Information page: <http://www.rpi.edu/dept/arc/training/latex/>

A longer, more complete Introduction than this document is *The Not So Short Introduction to L^AT_EX 2_ε* by Tobias Oetiker (150 pages):

<http://www.ctan.org/tex-archive/info/lshort/english/lshort.pdf>

A useful on-line command reference is:

<http://computing.ee.ethz.ch/.soft/latex/green/ltx-2.html>

The official L^AT_EX searchable FAQ is at: <http://www.tex.ac.uk/faq>

A good resource for all kinds of information is the T_EX Users Group (TUG) at <http://tug.org>.

You can also search the Comprehensive TeX Archive Network (CTAN):

<http://tug.ctan.org/search.html>

L^AT_EX packages

For a list of available packages with brief descriptions, see the CTAN catalog:

<http://texcatalogue.sarovar.org/brief.html>.

When you find a package you think may help you, before downloading it from CTAN, check first to see if you already have it installed. If not, you will need to download it and then install it. The following, from the FAQ, describes how to install a new package: <http://www.tex.ac.uk/cgi-bin/texfaq2html?label=instpackages>

L^AT_EX Thesis

The Rensselaer thesis web page,

<http://helpdesk.rpi.edu/update.do?artcenterkey=325>, has links to download Rensselaer's thesis class and to several documents that will help you prepare your thesis and create the PDF files necessary for electronic submission.

Mathematical Expressions

For information on using AMSLaTeX, see *The Short Math Guide for L^AT_EX*, at:
<ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf>

A detailed treatment of math, including both standard LaTeX and AMSLaTeX is *Math Mode* by Herbert Voss (150 pages):

<http://www.tex.ac.uk/tex-archive/info/math/voss/mathmode/Mathmode.pdf>

Symbols

A list of AMS Font symbols is at:

<http://www.rpi.edu/dept/arc/training/latex/amssymblist.pdf>

A manageable list of many useful symbols is “The Great, Big List of L^AT_EX Symbols” compiled by David Carlisle, Scott Pakin, and Alexander Holt,

http://www.rpi.edu/dept/arc/training/latex/LaTeX_symbols.pdf

The Comprehensive L^AT_EX Symbols List (Pakin) shows all the possible symbols and the packages required to access them (>100 pages):

<http://tug.ctan.org/tex-archive/info/symbols/comprehensive/symbols-letter.pdf>

Graphics

Rensselaer example files, input and output

Files that illustrate how to import graphics and also how to create eps files from Windows applications:

<http://www.rpi.edu/dept/arc/training/latex/Examples/graphics.tex> (input)

<http://www.rpi.edu/dept/arc/training/latex/Examples/graphics.pdf> (output)

Examples of using both the `graphicx` and the `rotating` packages to include landscape figures and tables:

<http://www.rpi.edu/dept/arc/training/latex/Examples/exrotating.tex> (input)

<http://www.rpi.edu/dept/arc/training/latex/Examples/exrotating.pdf> (output)

General graphics information

Complete documentation for the graphics bundle is in the file `grfguide.pdf`; look for it on your system. Information on the `graphicx` package is in section 4.4.

A helpful discussion of graphics formats, importing graphics, and tools for conversion is *Strategies for including graphics in L^AT_EX documents* by Klaus H^oppner at

<http://www.tug.org/pracjourn/2005-3/hoepfner/hoepfner.pdf>

For an extensive treatment of including graphics, see *Using Imported Graphics in L^AT_EX and pdfL^AT_EX* by Keith Reckdahl (~120 pages):

<http://www.ctan.org/tex-archive/info/epslatex.pdf>

Conversion utilities (see also page 35)

The `epstopdf` utility converts eps files to pdf files. It is very likely part of your TeX/LaTeX distribution.

The `jpeg2ps` utility converts JPEG images to eps files without uncompressing the images. If you don't already have `jpeg2ps` on your system, you can get it from:

<http://www.ctan.org/tex-archive/support/jpeg2ps/>

BibTeX

There is extensive material on BibTeX in the three books mentioned in the first section.

For on-line information on BibTeX, there is a helpful tutorial by Andrew Roberts:

<http://www.andy-roberts.net/misc/latex/latextutorial3.html>

A longer, more complete document covering both L^AT_EX's basic bibliography method and BibTeX is *Tame the BeaST* by Nicolas Markey:

http://tug.ctan.org/tex-archive/info/bibtex/tamethebeast/ttb_en.pdf

An excellent, concise overview of the Natbib package is the file `natnotes.pdf`, and the Natbib manual is in `natbib.pdf`. You can find both these files at:

<http://www.ctan.org/tex-archive/macros/latex/contrib/natbib/>

Installing L^AT_EX

The **TeXLive** distribution is a comprehensive TeX system with binaries for Windows and most flavors of Unix, including GNU/Linux. It also provides the popular MacTeX distribution and ProTeXt (see below). TeXLive is distributed each year on DVD and CD by the TeX Users Group to its members. For more information on TeXLive, see <http://tug.org/texlive/>. Downloading is free, but the download is huge.

Therefore, with the exception of TeXLive for Windows (see below), the best way to get the distribution may be to join TUG (<http://tug.org/join.html>).

Windows

- **TeXLive:** Rensselaer has made TeXLive for Windows available on RCS space. For general information see the Readme file, <http://www.rpi.edu/dept/arc/software/latex-doc/Readme-RPI.html>. To install the software, follow the detailed installation instructions at <http://www.rpi.edu/dept/arc/software/latex-doc/install-windows.pdf>. You should print these instruction for easier use when installing.
- **MikTeX:** This popular distribution can be downloaded free from <http://miktex.org/>. It's not quite as inclusive as TeXLive, but has a nice "package manager" that makes it easy to install new packages. It uses its own previewer, called Yap. An easy way to install MikTeX is to use ProTeXt (<http://tug.org/protext/>), which guides the installation via an explanatory

pdf file with clickable links. ProTeXt is also included with recent TeXLive distributions.

You will need an editor/shell to work with L^AT_EX on Windows. WinShell (<http://winshell.org>) and WinEdt (<http://winedt.com>) are both highly recommended.

UNIX systems

Linux and other UNIX-like systems usually have TeX/LaTeX as an installation option, so you may already have it installed. To check if it's on your system, just type `latex` at a command prompt. If it's not there, you may be able to install it easily using your system administration package management tool. Another excellent option is TeXLive, which is likely to be more up-to-date than the version that came with your system.

Macintosh

The highly recommended **MacTeX** distribution is a complete, easy-to-install TeX distribution for Mac OS X users that requires Mac OS 10.3 or higher. Information and installation instructions are at <http://tug.org/mactex/>. It is also included with recent TeXLive distributions.