A LaTeX Tutorial

Jim Alexander*

September 5, 2002

1 Some History

- TeX developed by Donald Knuth in 1977
- Pronounced /tɛk/ because it comes from the Greek word $\tau\epsilon\chi\nu\eta$ – 'art'
- Leslie Lamport developed LaTeX in the early 80's on top of TeX to make it easier to use
- LaTeX fragmented into several dialects over the years; LaTeX 2ε attempts unify these dialects as well as providing some new functionality. This document only covers the 2ε version of the language.

2 Setup

The default LaTeX commands provided on Eniac and some other systems here at Penn are version 2.09, not 2ε. To get the newer LaTeX commands, you will need to modify your path so that the directory `/pkg/teTeX/bin` appears prior to `/pkg/bin`, `/usr/local/bin` and `/usr/bin/X11`. What you have to do to make this happen depends on the identity of your shell and the current state of your shell initialization files. If you need help doing this, please ask for it.

The CIS graduate student machine gradient appears to now have version 2ε, so changing your path shouldn't be necessary.

If you would like to set up LaTeX for your own machine running a UNIX-like operating system, look for binary distributions of `tetex` – most Linux distributions have it on their CDs. Installing it from sources is nontrivial. For Windows, look for a distribution called TeX Live (I do not run Windows, though, so can't really help you with this).

There are various front-ends available to make interacting with LaTeX more like a word processor (a very nice one for UNIX is called LyX). This tutorial has nothing more to say about such systems, however: it is intended to get you started with writing in LaTeX directly.

3 The System

- The content of your document is placed in a file ending in the extension '.tex'. The command `latex` is run on the resulting file with the file name as an argument. Several output files are produced (refer to Figure 1):
  - `mypaper.tex`
    your input file

*Inspired (some bits swiped verbatim) from a similar introduction given at UC Santa Cruz by Kevin Karplus, Phokion Kolaitis and Richard Hughey.

- `mypaper.dvi`
  device independent file – this is the compiled version of your input. It contains the position and style of everything in your document and is converted with another program into the language of the final target file type, e.g. the printer language Postscript.
- `mypaper.log`
  detailed processing information and error messages from the LaTeX compiler
- mypaper.aux
  contains cross referencing information for labeled items and bibliography references

- The *device independent* or *dvi* file is converted to whatever final output format you want with other programs. The converters that you want to know about are:
  - `xdvi`
    a command for displaying the content of a dvi file in the X windowing system
  - `dvips`
    a command for converting the dvi file to postscript
    * `dvips -Pprintername mypaper.dvi`
      to send your file to the printer with name `printername`
    * `dvips mypaper.dvi -o output.ps`
      put the postscript output into the file `output.ps` (if the filename argument is omitted, the output will be written to `mypaper.ps`, i.e. the same base filename as your original input file with the '.ps' extension). The result may be sent to a printer with the `lpr` command or may be viewed at an X terminal with the `ghostview` command.

- A new alternative to the `latex` command, `pdflatex`, is available to produce typeset output in Adobe's PDF format directly, bypassing the conversion to dvi and postscript. If you are writing a new document and the final product you want is PDF format anyway, then you should consider just using `pdflatex` from the beginning – just substitute `pdflatex` wherever `latex` occurs in the rest of this document. There are a few gotchas related to using `pdflatex` with graphics, however; these are mentioned briefly in Appendix A.

- The `bibtex` command takes a `bib` file and formats bibliography entries in the style you specify in your input file as well as resolving citations. More on this in section 7.

4 LaTeX Commands: The Basics

- The upper- and lowercase letters, the digits, and the following punctuation characters:

  ```
  . : ; , ? ! ` ' ( ) [ ] - / * @
  ```

  can appear in your input file and are typeset (more or less) verbatim.

- Sentences are entered by just typing them in your favorite editor. Whitespace characters (spaces, tabs and newlines) are ignored - LaTeX will decide how to space your text appropriately. Paragraph breaks are represented by a line consisting of nothing but whitespace (typically just an empty line).

- These characters are used by LaTeX commands:

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

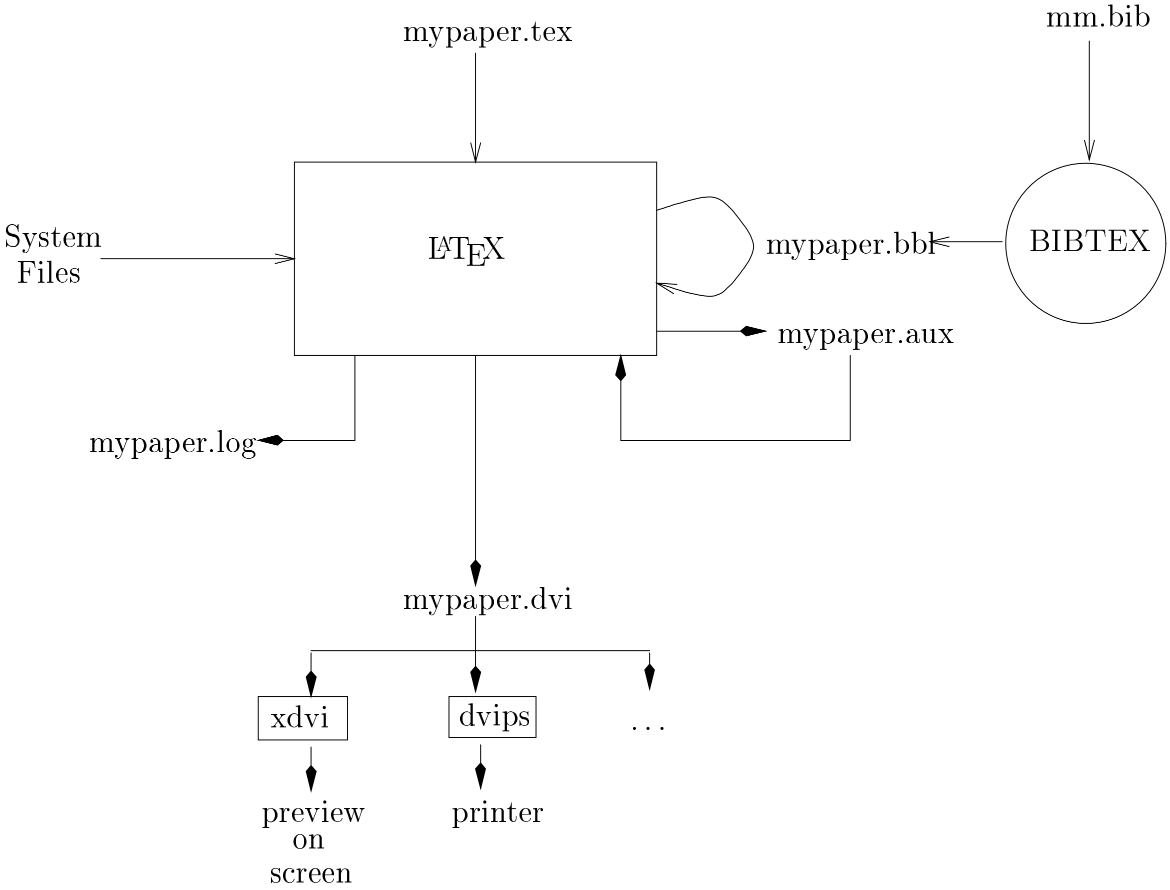

Figure 1: The LaTeX System

If you want them to appear in your text, most of these can be produced by prefixing them with a backslash, e.g. `\$` gets you $.

- Most commands begin with a backslash \ followed by the name of the command. If any arguments are *required*, they are enclosed in curly braces. *Optional* arguments are enclosed in square brackets before any of the required arguments. We'll see examples of the commands that specify the basic structure of your document in the next section.
- LaTeX is more like a programming language than a word processor, and as is advisable with any programming project, you should write you document *incrementally*. That is, save your file and run it through the `latex` command frequently, and probably view the result with `xdvi`. It is far easier to figure out how to fix problems if there are only a few of them.

5 Basic Document Structure

- Your document begins with the command:

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

The standard document classes are `article`, `book`, `report`, `letter`, and `slides`. You will almost always use the `article` class.

Options commonly used include:

- `10pt`, `11pt`, or `12pt`
  Specifies the default type size of your document. The default is `10pt` if not specified otherwise.
- `onecolumn` or `twocolumn`
  Specifies one or two column formatting. One column is the default.
- `notitlepage` or `titlepage`
  Generate a separate page each for the title and the abstract if the `titlepage` option is specified. The `notitlepage` places the title and abstract on the same page. The article class defaults to `notitlepage`.
- `leqno`
  Places equation numbers on the left side (rather than the right).

- The body of your document is enclosed as follows:

```
\begin{document}

text of your document

\end{document}
```

- Everything between the `\documentclass` and `\begin{document}` commands is called the *preamble*. In this section of the document, you place declarations that affect the whole document. Here's one useful such declaration that gives you empty headers and footers (without, e.g., page numbers):

```
\pagestyle{empty}
```

You also load packages in the preamble. Packages contain commands that are not in the base language (e.g. for drawing, including and manipulating graphics files, etc.) or that redefine some of the behaviour of the base commands. You load a package with the `\usepackage` command. Here's an example with a package I use regularly:

```
\usepackage{fullpage}
```

The `fullpage` package uses much smaller margins than the default page formatting (which gives you margins suitable for binding).

More packages will be mentioned in the next section.

6 Beyond the Basics

6.1 Some Niggly Details

- As is conventional in typeset documents, a period coming at the end of a sentence has a bit of extra space. TEX thinks that a period is the end of a sentence unless it follows an uppercase letter. You have to fix cases where this heuristic fails if you want your spacing to be perfect:
  - For abbreviations, put a `\␣` (backslash-space) right after the period.
  - If your sentence really ends in a capital letter, put the `\@` command immediately before the period.

The spacing heuristic (and these workarounds) also work for ?, ! and :.

Ellipsis (...) should be done with the command `\ldots` – just using a sequence of periods will give you really strange spacing.

- If you don't want a new paragraph indentation after you have a line in whitespace in your source file, start the line with `\noindent`.
- Sometime linebreaks will occur in bad places: in the name of table reference like Table 7, for instance. You can tell LaTeX to not split a line at a particular word break by replacing the space with a tilde (`~`). The tilde produces a normal space, but prevents a line break.

  LaTeX also defaults to full justification, and it will hyphenate where necessary to avoid having a line that looks too sparse. You can force LaTeX to not hyphenate a particular word by putting it inside an `\mbox` command, e.g., `\mbox{Pennsylvania}`. The `\mbox` command has many other uses, one of which will be mentioned in conjunction with math mode, below.
- If you want to insert a linebreak, use the `\\` command.
- Sometimes pagebreaks will appear in bad places as well. If you have a line or two at the end of a page, for instance, that you really would rather appear at the top of the next page, you can force a page break with the `\clearpage` command. This will, of course, leave extra whitespace at the bottom of a page. Also, you are strongly advised not to fuss with such spacing issues until your document is otherwise complete, otherwise you might have to do it all again if you insert more text somewhere.
- The open quote mark is made with the backtick ` and the close is made with a single quote '. Double or "scare" quotes are made with two of the individual quote marks in a row: ``` ``scare'' ```. The double quote character " itself is pretty much never used.
- The comment character is `%` - anything following one is ignored. Use `\%` to get a literal %.

6.2 Font Sizes and Styles

In section 5, we saw how to set the base font size for your document. What if you want to deviate from this size? You choose one of the declarations listed below, and you get a size relative to your default size, which you get back with the `\normalsize` declaration.

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

Once you make a size declaration within a given declaration scope, it remains in effect until you change it to something else. The scope of the declaration is normally the whole document, but you can change the scope by placing the declaration and the text you want it to affect inside of curly braces. These two ways of doing it are shown below:

```
\scriptsize
smallish
\normalsize

normal

{\large largish}
```

smallish
normal
largish

Choosing a text style can also be done with a declaration, but a command is also available. I'll only illustrate the command form (since that's what I always use), but I'll list the equivalent declaration if you want to use it:

Command	Style	Declaration
`\textup{Upright (default)}`	Upright (default)	`\upshape`
`\textit{Italic}`	*Italic*	`\itshape`
`\textsl{Slant}`	*Slant*	`\slshape`
`\textsc{Small Caps}`	SMALL CAPS	`\scshape`
`\textmd{Medium (default)}`	Medium (default)	`\mdseries`
`\textbf{Bold}`	**Bold**	`\bfseries`
`\textrm{Roman}`	Roman	`\rmfamily`
`\textsf{Sans serif}`	Sans serif	`\sffamily`
`\texttt{Typewriter}`	`Typewriter`	`\ttfamily`

Most of these commands can be combined in the obvious way to get more complicated styles (but note it is possible that your system might not have fonts to match each every combination you might want).

LaTeX also provides a command `\emph` for when you want to emphasize a piece of text - this translates to italics, by default, for postscipt output.

6.3 Spacing

To insert some horizontal or vertical space, you can use the `\hspace{length}` or the `\vspace{length}` commands, respectively. The `length` may be specified as a decimal number with units in inches `in`, millimeters `mm`, centimeters `cm` or points `pt`. There are also units that depend on the current font an style: `em`, the width taken up by an "M" and `ex`, the space taken up by an "x".

These commands really shouldn't be used to align chunks of text; use a table (discussed below) or a tabbing environment (not covered in this tutorial) if at all possible.

Another useful command is `\hfill`, which is shorthand for `\hspace{\fill}`. This gives you an example of a rubber length: a length that has some natural length, but will take up all of the space in its domain if it can. `\hfill` has a natural length of 0, but will take up all of the extra space in a line if it can. This is one way of right-justifying a line of text:

```
\hfill The edge
```

gives:

The edge

If you have more than one `\hfill` in a line, they will split the space evenly, like so (note the use of `\noindent`):

```
\noindent 1 \hfill 2 \hfill 3
```

1 2 3

Finally, to center something on the page, simply enclose it between `\begin{center}` and `\end{center}`.

6.4 Titles

To specify title information, you make two or three declarations with the \title, \author, and (optional) \date commands. The declaration used for this handout are:

```
\title
{
A \LaTeX\ Tutorial
}
\author
{
Jim Alexander\thanks{Inspired (some bits swiped verbatim) from a similar
introduction given at UC Santa Cruz by Kevin Karplus, Phokion Kolaitis and
Richard Hughey.}
}
```

To generate the actual title, you put the command \maketitle as the first command in the body of your document. The declarations can appear anywhere before the \maketitle command; I usually put them in the preamble. Some comments:

- Note that I have not made the \date declaration. If you do not, the result of the \today command is used, which gives you the current date.
- Note also the use of the \thanks command – this generates a footnote in the title suitable for such things as funding acknowledgements.
- Separate multiple authors with the \and command.
- If you want to have multi-line entries for each author, e.g. if you to show their affillation, use \\ to separate the lines.

The text between the commands \begin{abstract} and \end{abstract} typesets an abstract wherever you place the commands (typically right after \maketitle).

6.5 Sectioning

You can see examples of sectioning commands in action throughout this handout – the title of this subsection (6.5) was created with the \subsection command (note the \label command as well; this is covered in the next section):

```
\subsection{Sectioning\label{subsec:sectioning}}
```

Major sections are created analogously with the \section command. The book document class also has a \chapter sectioning command, and there are a few others, more rarely used.

6.6 Labeling and References

If you want to refer to some numbered entity in your text, it is best to use the \label and \ref commands rather than explicitly typing a number. This way, LaTeX can keep track of the numbering automatically – if you insert new numbered material, you don't have to do renumbering by hand. The argument to the \label command can be any string. As you might see from my usage of the labeling command in subsection 6.5, I usually make my labels in two parts, separated by a colon. The first part is the type of the entity being labeled, in this case a subsection (other types I use often are fig for a figure and dat for a piece of data). The second part is some name that evokes the specific labeled entity for me, in this case this is sectioning.

You then reference the labeled entity with the same string you used to label it. For example, I have been using the command \ref{subsec:sectioning} to get the number of the previous section as I have referred to it in my text.

When you first insert a new reference and then run latex on your file, you will probably get a warning that looks like this:

```
LaTeX Warning: Reference `subsec:sectioning' on page 5 undefined on input line
440.

[5] [6] [7] (latex-tut.aux)

LaTeX Warning: There were undefined references.


LaTeX Warning: Label(s) may have changed. Rerun to get cross-references right.
```

Just do as LaTeX says and rerun your compilation command. On the first pass, LaTeX writes out all of your references abstractly to the file that ends in the extention `.aux`. On the second pass, it assigns the actual numbers. This will also happen with bibliographical citations.

6.7 Footnotes

To insert a footnote, simply insert a `\footnote{Text of footnote}` command exactly where you want a footnote to appear[1]. Footnote indices are tracked automatically like the labels described in the previous section.

6.8 Lists

Throughout this document, I have made use of the `itemize` environment to produce bulleted lists. Here's another example:

- Foo
- Bar
- Baz

... which was produced like so:

```
\begin{itemize}
\item Foo
\item Bar
\item Baz
\end{itemize}
```

To make numbered lists, use the `enumerate` environment:

1. First
2. Second
   (a) Sub-first
   (b) Sub-second
3. Third

```
\begin{enumerate}
\item First
\item Second
  \begin{enumerate}
  \item Sub-first
```

[1] Like right here

```
  \item Sub-second
  \end{enumerate}
\item Third
\end{enumerate}
```

Note that as shown in the above example (and used extensively throughout this handout) `itemize` and `enumerate` environments may be nested. Nested environments are indented and the style of the bullet or number is varied[2].

In the base version of the `enumerate` environment, you don't have easy control over the style of the numbers of your list items. There is a enhanced version of `enumerate` that gives you precisely this control. To use it, put `\usepackage{enumerate}` in your document preamble. Then, in square brackets after the `\begin{enumerate}` command, you give an example of the style you want by showing what the first number in your list should look like. The possible styles of numbers are upper and lower case versions of Roman numerals or letters, or Latin numerals, and the digits can be "decorated" however you like. Here's an example:

I. First

II. Second

  -a- Sub-first

  -b- Sub-second

III. Third

  Statement 1 = True

  Statement 2 = False

  Statement 3 = False

```
\begin{enumerate}[I.]
\item First
\item Second
  \begin{enumerate}[-a-]
  \item Sub-first
  \item Sub-second
  \end{enumerate}
\item Third
  \begin{enumerate}[{Statement} 1 =]
  \item True
  \item False
  \item False
  \end{enumerate}
\end{enumerate}
```

6.9 Math Mode

LaTeX's capabilities in the typesetting of mathematical formulas is virtually unmatched. Covering math mode in depth, however, could easily comprise several more tutorials on its own, so I'll only say a few things about the basics. To do serious math typesetting, you will absolutely need a good reference book (see section 8).

An inline mathematical formula is enclosed in dollar signs $...$; a formula enclosed in `\[ ... \]` appears on its own line (this is called a *displayed* formula). The environment `\begin{equation}` ... `\end{equation}`

[2]Note also that I have indented the nested `enumerate` environment, mimicing the output. This enhances the readability of your raw input file.

typesets just like a displayed formula, but also includes an equation number generated from the `equation` counter, which is managed by LaTeX automatically. A simple example:

$$y = mx + b \tag{1}$$

Spaces have no effect on how formulas are typeset; LaTeX will decide for itself how symbols should be grouped. If you want a piece of text inside of a formula to behave as in the normal text mode (called "LR mode," as it happens), put that text inside an `\mbox` command.

Text mode font styles do not work in math mode; there are equivalents for many of them available, though. See below. Math mode symbols are normally typeset in an italic shape, but the spacing is different than for the `\mathit` shown below; use the below font styles for multiletter symbols you want in a different style.

`\mathit{Italic}`	Italic
`\mathrm{Roman}`	Roman
`\mathbf{Bold}`	$\mathbf{Bold}$
`\mathsf{Sans\ serif}`	$\mathsf{Sans\ serif}$
`\mathtt{Typewriter}`	$\mathtt{Typewriter}$
`\mathcal{CALLIGRAPHIC}`	$\mathcal{CALLIGRAPHIC}$

For the remainder of this section, I'll go over some of the more heavily used math mode commands.

- The arithmetic operators $+$, $-$ and $/$ do the expected thing in math mode. `\div` gives the division operator $\div$ and explicit multiplication operations can be symbolized with `\cdot`$\rightarrow \cdot$ or `\times`$\rightarrow \times$.

- The caret `^` superscripts the expression following it to the expression preceding it. The underscore `_` subscripts.

  Superscripting and subscripting may be mixed as shown in x_{4}^{2x+1}, which was made with the command `$x_{4}^{2x+1}$`. The order that you super- or subscript most often doesn't matter.

  It's a good habit to put the sub- or superscripted expression inside curly braces or your expression might get grouped in unexpected ways.

- The Greek alphabet is obtained by command named after the characters' names, e.g. `\pi`$\rightarrow \pi$ and `\delta`$\rightarrow \delta$. Capital Greek letters are obtained by capitalizing the first letter of the name, e.g. `\Omega`$\rightarrow \Omega$.

- Fractional expressions in running text are usually made with the slash `/`, e.g. `1/2`$\rightarrow 1/2$. Complicated fractions (which are usually displayed) can be made with the `\frac` command, which takes two options: the numerator and the denominator. Here's an example (note that it can be used recursively):

  ```
  \[\frac{\alpha + z'}{1 + \frac{n}{17\rho}} \]
  ```

  $$\frac{\alpha + z'}{1 + \frac{n}{17\rho}}$$

  Fractional expressions are used heavily in programming language theory to typeset inference rules. For example, here is the conditional inference rule from chapter 3 of your text (the actual font used in the text is slightly different):

  $$\frac{t_1 \in \mathcal{T} \quad t_2 \in \mathcal{T} \quad t_3 \in \mathcal{T}}{\mathrm{if}\ t_1\ \mathrm{then}\ t_2\ \mathrm{else}\ t_3 \in \mathcal{T}}$$

  This was typeset with the following command:

```
\[ \frac{t_1 \in \mathcal{T} \ \ \ t_2 \in \mathcal{T} \ \ \ t_3 \in \mathcal{T}}
        {\mathrm{if} \ t_1 \ \mathrm{then} \ t_2 \ \mathrm {else} \ t_3
         \in \mathcal {T}} \]
```

The backslash-space command is used to insert space where math mode would normally have none, just to make the overall result look a little neater. If one were typesetting a lot of expression that looked like this, one would write a custom command to avoid needless repetition, as is a good idea in any programming language. Doing this is well beyond the scope of this introduction however – interested readers should refer to discussion of `\newcommand` in a more comprehensive reference on LaTeX (see section 8).

- Summations can be typeset with the `\sum` command. You specify the bounds of the summation with the super- and subscript commands, and LaTeX will move them to the right place. An example:

```
\[ \sum_{i=0}^{n} \frac{1}{i^2} \]
```

$$\sum_{i=0}^{n} \frac{1}{i^2}$$

The same command inline looks like this: $\sum_{i=0}^{n} \frac{1}{i^2}$. If you want the display style of summation in your inline formula, simply include the command `\displaystyle` before the `\sum` command in the formula.

Products, unions, integrals, etc., are done with the `\prod`, `\bigcup`, `\int`, etc., which otherwise have the same command syntax.

6.10 Tables

The standard LaTeX table-making environment, `tabular`, is very powerful. As is the case throughout this tutorial, this section covers the basics.

The body of the table is surrounded by `\begin{tabular}` and `\end{tabular}`. The `\begin{tabular}` command has a required parameter which is a string specifying how the columns of your table should be laid out. The column specifiers you will use most often are:

`l`	a left-aligned column
`r`	a right-aligned column
`c`	a centered column
`|`	a vertical line running the full length of the table

Let's see an example of a simple table without any lines:

```
\begin{tabular}{lc}
foo  & bar \\
junk & stuff
\end{tabular}
```

foo	bar
blah	defenestration
junk	stuff

Note that cells in a given row are separated with ampersands `&` and all of the rows but the last are terminated with `\\`.

If we add lines to the table, we generally want to draw a box around the whole table as well. Here's the same example with many more lines than you would probably want:

```
\begin{tabular}{|l|c|} \hline
foo  & bar \\ \hline
blah & defenestration \\ \hline
junk & stuff \\ \hline
\end{tabular}
```

foo	bar
blah	defenestration
junk	stuff

The vertical lines are part of the column specification string (the | character), including the outside edges of the table. The `\hline` command places horizontal lines wherever you want them. To make the top of the box surrounding the table, you put an `\hline` command before the first row of your table. As you can see, I usually put it on the same line and `\begin{tabular}`. The bottom line goes at the end of the last row of the table. Note that in this case, there is also a `\\` on that last line – this is the only time the `\\` should appear on the last row.

One way you might want to make a table more complicated is to have subcolumns within columns, like in this table from one of my Phonology papers:

Height	Front				Back		
	-Round		+Round		-Round	+Round	
	short	long	short	long	long	short	long
High	i /i/	í /iː/	ü /y/	ű /yː/		u /u/	ú /uː/
Mid		é /eː/	ö /ø/	ő /øː/		o /o/	ó /oː/
Low	e /ɛ/				á /ɑː/	a /ɔ/	

This is accomplished with the `\multicolumn` command, which I am not going to say anything else about beyond providing the source for this table. My LaTeX books had poor examples when it came to learning how this command works, and hopefully you can refer to my example to help you figure it out when you need to. Here's the source:

```
\begin{tabular}{|l||c|c|c|c||c|c|c|} \hline
% the top line of the headings
\multicolumn{1}{|c||}{} &
\multicolumn{4}{c||}{Front} &
\multicolumn{3}{c|}{Back} \\ \cline{2-8}

% the middle line of the headings
\multicolumn{1}{|c||}{Height} &
\multicolumn{2}{c|}{-\textsc{Round}} &
\multicolumn{2}{c||}{+\textsc{Round}} &
\multicolumn{1}{c|}{-\textsc{Round}} &
\multicolumn{2}{c|}{+\textsc{Round}} \\ \cline{2-8}

% the bottom line of the headings
\multicolumn{1}{|c||}{} &
\multicolumn{1}{c|}{short} &
\multicolumn{1}{c|}{long}  &
\multicolumn{1}{c|}{short} &
\multicolumn{1}{c||}{long}  &
\multicolumn{1}{c|}{long} &
\multicolumn{1}{c|}{short} &
\multicolumn{1}{c|}{long}  \\ \hline
```

```
% the body of the table - the details of this aren't important
\textsc{High} & i /i/         & \'{\i} /i{\length}/ & \"{u} /y/ &
  \H{u} /y{\length}/ &                          & u /u/ & \'{u} /u{\length}/ \\
\textsc{Mid}  &               & \'{e} /e{\length}/  & \"{o} /\o/&
  \H{o} /\o{\length}/&                          & o /o/ & \'{o} /o{\length}/ \\
\textsc{Low}  & e /\niepsilon/ &                        &         &
                    & \'{a} /\scripta{\length}/ & a /\openo/ & \\ \hline
\end{tabular}
```

If you want to have a table that consists entirely of equations, use the `array` environment instead of `tabular`. It works pretty much the same as `tabular` except every item is typeset in math mode - this saves you from enclosing every item in your table between dollar signs.

6.11 Code

To typeset source code, you usually don't want LaTeX to change the spacing, but rather you want it left exactly as it is. For this, one employs the `verbatim` environment: just enclose anything that you want inserted exactly as you typed in between `\begin{verbatim}` and `\end{verbatim}`. This is exactly how all of the literal chunks of LaTeX are typeset throughout this document. If you want to insert a literal LaTeX command into running text, the command to employ is `\verb`. This is an unusual command: it's first argument is the very next character after the `b`: it can be any character and it is used as a delimiter. It is followed by any sequence of characters except that first character; a reoccurence of that first character delimits the end of the command. Everything between the two occurences of the delimiter character is inserted literally in typewriter style. As you might guess, this command is used everywhere in this document, 3 times so far in this paragraph. The first occurence in the current paragraph looks like this in the source file: `\verb|\begin{verbatim}|`. You would employ this command if you want to insert anything that LaTeX would mistakenly interpret as a real command.

6.12 Floats

One usually places tables and figures inside `figure` and `table` environments, respectively. These are examples of floating environments – objects that cannot be split in the middle like normal text, and instead must "float" to the next available space. Here is the figure environment bracketing Figure 1 from the beginning of the handout:

```
\begin{figure}[ht]
  body of figure here
\caption{The \LaTeX\ System}
\label{fig:texsys}
\end{figure}
```

The `ht` options given tell LaTeX to place the figure exactly where it appears in your text, if possible, and otherwise at the top of the next available page.

LaTeX doesn't always make the greatest choices when it comes to placing floats. Fighting with weird float placement has to be handled on a case-by-case basis and is beyond the scope of this tutorial.

7 BibTeX

The `bibtex` program reads entries from a bibliography database file (`.bib`) and produces typeset citations and bibliography entries in the style specified in your input file.

The database file consists of entries that look like this:

```
@book{Gaz79a,
   author    = {Gazdar, Gerald},
   title     = {Pragmatics : implicature, presupposition and logical form},
   publisher = {Academic Press},
   year      = 1979
}
```

The part at the beginning of the entry that starts with the @-sign, in this case `@book`, specifies the type of the entry. The content of the entry is enclosed in curly braces and subparts are separated by commas. The first member of the entry is the *key*, the name that is used to cite an entry with the `\cite` command in your document file. This may be any string you want so long as it is unique over all of the `.bib` files you are using. These identifiers are case-sensitive[3].

The rest of the entry are called *fields* and consist of the name of the field (which is case insensitive), an equal sign, and the value of that field. The content of the field should be surrounded by either curly braces or by a pair of double-quote (`"`) characters (the braces/quotes are optional if the field has a purely numerical value).

Some tips on making your entries turn out how you want them once they are typeset:

- If the text in your field contains any LaTeX commands, say commands for accented characters, you should enclose the command in curly braces as sometimes these will confuse `bibtex`.
- In a title entry, some bibliography styles will lower the case of all of all of the letters but the first. If there are letters that should never be lower case (e.g. those beginning proper names in English), enclose them in curly braces.
- Names in multiple author works should all be separated by the word `and`.
- Although you can enter names in `author` or `editor` fields as `Given-name Family-name`, I recommend entering them as `Family-name, Given-name` instead since names with multi-word family names must be entered this way anyway (otherwise `bibtex` will treat the first word of the family name as a middle name).

In addition to `book`, entry types you will commonly use include `article`, `incollection`, `inproceedings`, `phdthesis` and `unpublished`. For each type, some fields are required and others are optional. Rather than enumerate these, I think it's easier to emulate examples. Below is an example of each of the field types I mention above. Also included is an example of a `@string` entry – this allows you to define abbreviations for strings that are used in more than one entry in your file.

```
@string{sctc8 = "Proc. Eighth Annual Structure in Complexity Theory Conference"}
@article{Gaz79b,
   author    = {Gazdar, Gerald},
   title     = {A Solution to the Projection Problem},
   journal   = {Syntax and Semantics},
   volume    = 11,
   number    = 1,
   year      = 1979,
   pages     = {57--89}
}
```

[3] I use keys that look like the labels produced by the `alpha` bibliography style: essentially the first 3 letters of the author's name and a two digit year, with a letter appended if that author has more than one publication for that year

```
@incollection{Kra81,
   author    = {Kratzer, Angelika},
   title     = {The Notional Category of Modality},
   booktitle = {Words, Worlds, and Contexts : New Approaches in Word Semantics},
   editor    = {Eikmeyer, Hans-Jurgen and Rieser, Hannes},
   publisher = {W. de Gruyter},
   year      = 1981,
   pages     = {38--74}
}
@inproceedings{HH93,
   author    = {T. Hirst and D. Harel},
   title     = {Taking it to the limit: on infinite variants of {NP}-complete
                problems},
   booktitle = sctc8,
   year      = 1993
}
@phdthesis{Hei82,
   author    = {Heim, Irene Roswitha},
   title     = {The Semantics of Definite and Indefinite Noun Phrases},
   school    = {University of Massachusetts},
   year      = 1982
}
@unpublished{Ell95,
   author    = {Ellison, T. M.},
   title     = {Phonological Derivation in {O}ptimality {T}heory},
   note      = {Unpublished manuscript},
   year      = 1995
}
```

To place the reference list in your document, if your database file is named `mybib.bib`, put the command `\bibliography{mybib}` where you want the reference list to appear. That is, the argument to the `\bibliography` is the name of the database file with the extension removed. This is usually last, just before the `\end{document}` command. To make citations in the text of your document, simply invoke the `\cite` command with the bibliography database key as an argument. For example, `\cite{Hei82}` cites Heim's Ph.D. thesis as listed above. You can cite multiple sources by separating the keys with commas. If you want one or more items to appear in your reference list without having cited them, invoke the `\nocite` command with the appropriate keys. Only the entries referenced in at least one `\cite` or `\nocite` command will appear in your reference list – this allows you to keep a master bibliography file from which you can select whatever entries you want without having to have all of them. If you have a small `bib` database that you want to include in its entirety, include the special command `\nocite{*}` in your text.

What the printed bibliography looks like depends on the bibliography style you choose. You specify your desired bibliography style with the `\bibliographystyle` command. There are many, many styles available; the standard ones are `plain`, `unsrt`, `alpha`, and `abbrv`. Describing in detail what these styles look like would be difficult and tedious – you should just try them and see. I can describe some differences between the standard styles, though:

`plain`	Entries are sorted alphabetically and labeled with a number
`unsrt`	Like `plain`, but in order of citation
`alpha`	Like `plain`, except labels are strings formed from the name of the author and the year of publication
`abbrv`	Like `plain`, except names, titles, etc. are abbreviated

In each case, the citations in your text will look like the labels in the reference list.

It remains to describe how to use the `bibtex` command to generate your references. Once you have the bibliography commands in your source file (and at least one `\cite` or `\nocite` command if this procedure is going to do anything):

(1) Run the `latex` command as usual. It will complain about there being undefined references for each of your citations, very similar to those described in subsection 6.6, the subsection on labeling.

(2) Run `bibtex` on your paper, specifying the filename of your paper **without** the `.tex` extension, e.g. `bibtex mypaper` (what `bibtex` will actually look for is the `.aux` file automatically generated by `latex` – that you have to leave the file extension off is just an annoying inconsistency). Don't use the name of your bib database as the argument to `bibtex` – it will consult your paper to see which bib files should be referenced.

(3) Run `latex` on your source file **twice** more. The first time incorporates the reference list; the second time resolves the labels.

You will need to redo this procedure each time you add or remove a citation: LaTeX will let you know you need to do so by complaining about undefined references.

8 Reference

- Leslie Lamport's *LaTeX: A Document Preparation System* is really a must-have. For more advanced needs, there is *The LaTeX Companion*, by Goossens, Mittelbach and Samarin (sometimes known as the "big dog" book).
- The commands that comprise the LaTeX system have useful manual pages describing the loads of options several of them have. See especially the man page for `dvips`.
- The Penn Math department maintains a links page on TeX resources. This is a good place to start if you want to find styles, fonts, etc., that don't seem to be in the default set installed on your machine. There is also information on TeX for Macs and PCs. The URL is:

  `http://www.math.upenn.edu/TeX.html`

A A note on PDF

It has become desirable to be able to generate PDF (Adobe's Portable Document Formant) output instead of or in addition to Postscript, e.g., for distribution on the web. There are several ways of doing this, but by far the easiest way of getting good results is to simply use the `pdflatex` command in place of `latex`. This generates PDF from your LaTeX source directly, bypassing the dvi file completely. The only drawback that I am aware of of this method is that all of your figures must be converted to PDF as well, which can be difficult or at least undesirable depending on how the original graphics were generated. Also, the many LaTeX reference books on the market have not yet been updated to reflect the shift to PDF in the publishing world, so a lot of the nice postscript tricks detailed in those books are likely not to work. If you *can* convert your figure to PDF with acceptable quality, then the figure is inserted with the command `\includegraphics{pic}`, where your figure is in the file `pic.pdf` (i.e. the `.pdf` extension must be omitted).

If you are editing an old document that proves resistant to working with `pdflatex`, another path to take to PDF is to use the command `dvips -Ppdf ...` to convert your file to postscipt with embedded fonts, which should produce better results with postscript to PDF converters like Adobe Distiller (available as the command `distiller` on gradient). Normally, the `-P` flag is used to choose a printer to print to, but most modern LaTeX installations have a fake printer made specifically to produce postscript that is intended to become PDF.