LaTeX Tips and Tricks

Claire M. Connelly
`cmc@math.hmc.edu`

Fall 2005

1 What is LaTeX?

LaTeX is a tool that allows you to concentrate on your writing while taking advantage of the TeX typesetting system to produce high-quality typeset documents.

LaTeX's benefits include

1. Standardized document classes
2. Structural frameworks for organizing documents
3. Automatic numbering and cross-referencing of structural elements
4. "Floating" figures and tables
5. High-level programming interface for accessing TeX's typesetting capabilities
6. Access to LaTeX extensions through loading "packages"

2 Structured Writing

Like HTML,[1] LaTeX is a markup language rather than a WYSIWYG[2] system. You write plain text files that use special *commands* and *environments* that govern the appearance and function of parts of your text in your final typeset document.

2.1 Document Classes

The general appearance of your document is determined by your choice of *document class*. Document classes also load LaTeX packages to provide additional functionality.

LaTeX provides a number of basic classes, including article, letter, report, and book. There are also a large number of other document classes available, including amsart and amsbook, created by the American Mathematical Society and providing some additional mathematically useful structures and commands; foils, prosper, and seminar, which allow you to create "slides" for presentations; the math department's thesis

1. HyperText Markup Language
2. What You See Is What You Get.

Command	Notes
`\part`	book & report only
`\chapter`	book & report only
`\section`	
`\subsection`	
`\subsubsection`	
`\paragraph`	
`\subparagraph`	

Table 1: Structural Commands in LaTeX.

class, for formatting senior theses; and many journal- or company-specific classes that format your document to match the “house style” of a particular periodical or publisher.

2.2 Packages

LaTeX packages, or *style files*, define additional commands and environments, or change the way that previously defined commands and environments work. By loading packages, you can change the fonts used in your document, write your document in a non-English language with a non-ASCII font encoding, include graphics, format program listings, add custom headers and footers to your document, and much more.

A typical TeX installation includes hundreds of style files, and hundreds more are available from the Comprehensive TeX Archive Network (CTAN), at <http://www.ctan.org/>.

2.3 Structural Commands

LaTeX provides a set of structural commands for defining sections of your document, as shown in Table 1.

Sections are numbered automatically by LaTeX during typesetting. If you change your mind and decide that a subsection should be promoted to a section, or moved to the end of your document, the sections will be renumbered so that the numbers are consistent.

Sections can also be `\label`ed with a tag such as

```
\section{Our Complicated Equations}%
\label{sec:complicated-eqs}
```

and referred to with a `\ref` or `\pageref` command, as in

```
In Section~\ref{sec:complicated-eqs}, we pointed out...
```

or

```
On page~\pageref{fig:gordian-knot}, we illustrated...
```

LaTeX substitutes the correct section number when typesetting your document.

The same commands can be used with numbered environments such as `equation`, `theorem`, and so forth.

2.4 Commands

LaTeX uses commands for changes that are very limited in scope (a few words) or are unlimited in scope (the rest of a document). For example, the commands

```
\textbf{bold}
\emph{italic (emphasized)}
\textsf{sans serif}
```

produce the following output in a typeset document:

> **bold** *italic (emphasized)* sans serif

These are "commands with arguments"—the command itself starts with a backslash (\\), and its *argument* appears inside braces `{ }`). Some commands may also have *optional arguments*, which are typed inside brackets (`[ ]`).

There are also commands that take no arguments, such as `\noindent`, `\raggedright`, and `\pagebreak`.

You can define your own commands, as discussed in Section 2.6.

2.5 Environments

LaTeX provides a number of *environments* that affect the appearance of text, and are generally used for more structurally significant purposes. For example, the commands listed above are typeset inside a `verbatim` environment typed inside a `quote` environment. Their results were typeset inside a `quote` environment.

Environments use special commands to start and close—`\begin` and `\end`, followed by the name of the environment in braces, as in

```
\begin{quote}
``This is disgusting---I can't eat this.  That arugala is so
bitter\ldots{} It's like my algebra teacher on bread.''
\flushright -- Julia Roberts in \emph{Full Frontal}
\end{quote}
```

producing

> "This is disgusting—I can't eat this. That arugala is so bitter... It's like my algebra teacher on bread."
>
> – Julia Roberts in *Full Frontal*

Some environments may take additional arguments in braces (required) or brackets (optional).

Note that the order in which environments nest is extremely important. If you type an environment inside another environment, the inner environment must be `\end`ed *before* the second environment is closed. It's also vitally important that you have an `\end` line for each `\begin` line, or LaTeX will complain.

2.5.1 The `document` Environment and the Preamble

The most important environment is the `document` environment, which encloses the *body* of your document. The code before the `\begin {document}` line is called the *preamble*, and includes the all-powerful `\documentclass` command, which loads a particular document class (see Section 2.1); optional `\usepackage` commands, which load in additional LaTeX packages (see Section 2.2); and other setup commands, such as user-defined commands and environments, counter settings, and so forth.

I generally also include the commands defining the title, author, and date in my preambles, but other people include them just after `\begin {document}`, before the `\maketitle` command, which creates the title block of your document.

2.5.2 Math Environments

One of the major hallmarks of TeX is its ability to typeset mathematical equations.

The two primary ways of doing so are with the use of *inline* and *display math environments*. These environments are used so often that there are shorthands provided for typing them. Inline math environments, such as $a^2 + b^2 = c^2$, can be typed as

```
\begin{math}
a^{2} + b^{2} = c^{2}
\end{math}
```

or

```
$a^{2} + b^{2} = c^{2}$.
```

Display math environments set your equation apart from your running text. They're generally used for more complicated expressions, such as

$$f(x) = \int \left(\frac{x^2 + x^3}{1} \right) dx$$

which can be typed as

```
\begin{displaymath}
f(x) = \int \left( \frac{x^2 + x^3}{1} \right)dx
\end{displaymath}
```

or

```
\[
f(x) = \int \left( \frac{x^2 + x^3}{1} \right)dx
\]
```

Generally, you'll want to use the `$` delimited form for inline math, and the `\[ \]` form for display math environments. [Besides being easy to type, these forms are *robust*, which means that they can be used in *moving arguments*, elements that TeX may need to typeset in more than one place (such as a table of contents) or adjust (such as footnotes).]

The `equation` **Environment** You'll probably want to use the `equation` environment for any formula you plan to refer to. LaTeX not only typesets the contents of an `equation` environment in display mode, it also numbers it, as in

$$f(x) = \int \left(\frac{x^2 + x^3}{1} \right) dx \tag{1}$$

written as

```
\begin{equation}
\label{eq:myequation}
f(x) = \int \left( \frac{x^2 + x^3}{1} \right)dx
\end{equation}
```

Note that you can refer to this formula as Equation 1 with

`\ref{eq:myequation}`.

2.6 Customization

The main advantage of using commands and environments is that they allow you to organize your writing. A useful side-effect is that you can change your mind about the way an element is typeset, and change all the appearances of that element in document by editing one piece of code. For example, in this document the names of environments have been set in "typewriter text", using a command I created called `\env`, which is defined as

```
\newcommand{\env}[1]{\texttt{#1}\xspace}
```

All I have to do to make the names of all the environments in the document appear in sans-serif type instead is to change that one line to

```
\newcommand{\env}[1]{\textsf{#1}\xspace}
```

3 Typesetting

So you've got a LaTeX source document. How do you get a typeset document that you can print or put on the web?

Typesetting a document is referred to as "TeXing", "compiling", or "typesetting". Generally, you want to create a PostScript file (for printing) or a PDF file (for printing or placing on the web). There are multiple ways to do both tasks.

3.1 Getting to Paper

Starting with a LaTeX document, `foo.tex`, you can create a PostScript file by running the following commands:

```
unix% latex foo
unix% dvips -o foo.ps foo
unix% lpr foo.ps
```

(On some systems, `dvips` automatically prints your document to the default printer. You need to specify the `-o` flag to get a PostScript file on such systems.)

3.2 PDF for the Web

Starting with `foo.tex`, you can create a PostScript file by typing

```
unix% pdflatex foo
```

or with the following sequence of commands:

```
unix% latex foo
unix% dvips -Ppdf foo -o foo.ps
unix% ps2pdf foo.ps foo.pdf
```

(The last step can be replaced by running the PostScript file through Adobe Acrobat Distiller.)

3.3 General Comments

LaTeX does its numbering (and some other functions) by writing information to an *auxiliary file*. It then reads that information in on the next pass, and uses it to typeset references. Thus you have to run `latex` or `pdflatex` at least twice whenever you make a change that affects the numbering of elements or the flow of text across pages. It's generally good practice to run LaTeX three times, or until it stops warning you about possible changes.

3.4 Additional Programs

There are some additional functions, such as indexing and bibliographies, that use external programs to read auxiliary files and produce LaTeX code for inclusion on later runs. We won't cover those programs in this document.

4 Tips and Tricks

LaTeX is a very complicated and powerful language. As a result, there are many sneaky aspects to it that will cause you problems if you don't know about them. Here are a few.

4.1 Special Characters

TeX and LaTeX have a number of "special" characters that are reserved for use by the language. Using these characters in your writing requires you to do a bit of extra work, as shown in Table 2.

Character		Function	To Typeset
#	octothorp	Macro parameter character	`\#`
$	dollar sign	Start/end inline math mode	`\$`
%	percent sign	Comment character	`\%`
&	ampersand	Column separator	`\&`
_	underscore	Subscripts, as in x_2	`\_`
{, }	braces	Parameters	`\{, \}`
˜	tilde	Nonbreaking space	`\~{}`
ˆ	caret	Superscripts, as in x^2	`\^{}`
\	backslash	Starts commands	`\verb||`

Table 2: Special Characters in LaTeX.

4.2 Comments and Spacing

You can add comments to your source file that won't appear in your typeset document by starting them with a %. Any line that starts with a % will be "commented out", and won't be interpreted. You can also add a % at the end of a line, with or without text, and it will make the end of the line disappear.

For example,

```
% This is a comment line.
This is not a comment line.

This line has a comment at the end%
% This line should be invisible.
of the line.
```

will typeset as

This is not a comment line.

This line has a comment at the endof the line.

Notice the lack of a space in "endof" on the last line of the typeset output. TeX expects a carriage-return character at the end of a line, and interprets that carriage return as an interword space. If you comment out the end of a line, you also comment out the carriage return on that line, and you'll have words run into one another unless you have a space before the %.

TeX collapses multiple spaces into one, and ignores whitespace at the beginning of a line. Thus

```
No spaces.

    Five spaces.

    A tab.
```

typesets as

No spaces.

Five spaces.

A tab.

(The lines are indented because they are at the start of a paragraph. You can suppress paragraph indentation with `\noindent`.)

Paragraphs are delimited by two carriage returns (with or without whitespace between them).

4.3 Quotes and Dashes

Because TeX was designed to do high-quality typesetting, it cares about which quotation mark and dash you're using, and requires you to specify the correct punctuation (although most text editors with special TeX modes will do the substitution for you).

Open and close double quotes—“ and ”—are created by typing ` `` ` and `''`, respectively. The double-quote mark, `"`, is typeset as " (and is useful for abbreviating “inches”, as in 36").

Single-quotes, ‘ and ’, are typed with `` ` `` and `'`.

There are three basic forms of dashes:

1. The hyphen, -, is typed as a single dash, `-`
2. The en dash, –, is typed as two dashes, `--`
3. The em dash, —, is typed as three dashes, `---`

Hyphens are used in hyphenated words, as in “complex-typesetting mechanism”. En dashes are used to indicate ranges, as in “there are 35–50 of them”. Em dashes are used to separate independent phrases, as in “John believed—honestly believed—that he was right.”

Note that you shouldn't type spaces around any of these dashes—they run directly against the words on either side, as in `35--50`.

4.4 Using Graphics with PDFTeX

PDFTeX supports PDF and JPEG as native graphic file formats. EPS is not directly supported—to use EPS figures with PDFTeX, you must first convert your EPS files to PDF.

If you're using a graphics program such as Adobe Illustrator to prepare your figures, just save them as PDF instead of (or in addition to) EPS.

If you don't have access to the tool you used to create your images, but you still need to convert them, you can use the program `epstopdf`.

The old version of `epstopdf` would write to standard output by default, so you had to redirect the output to a file, as in

```
unix% epstopdf foo.eps > foo.pdf
```

Current versions, however, will create a PDF file with the same basename as the original (but with the extension replaced). Thus,

```
unix% epstopdf foo.eps
```

does exactly the same thing as the previous example, and is easier to type, as well.

To convert a whole slew of files, you could use a command such as the following (with the csh):

```
unix% foreach f ( ‘find . -type f -name ’*.eps’‘)
foreach? epstopdf $f
foreach? end
```

For bash, you could do

```
unix$ for f in ‘find . -type f -name ’*.eps’; do epstopdf $f; done
```

These commands work by `\find`ing all files in or below the current directory (`.`) that have names that end in `eps`, and then operate on them.

4.5 Fonts Look Fuzzy in PostScript or PDF Files

When Knuth wrote TEX, typesetting was done by trained typesetters using expensive equipment to cast molten lead into runs of type. Knuth created his own font family, Computer Modern, by writing a tool called METAFONT. METAFONT reads in programs that define various aspects of every character in a font, and generates bitmap representations of those characters at a particular resolution, ready for printing.

Unfortunately, bitmaps with resolutions suited for printing look terrible on screen. The solution is to use Type 1 PostScript fonts instead of bitmaps. If you're using PDFTEX (or PDFLATEX), you get Type 1 fonts without having to do anything special (but see Section 4.4).

If you're using `dvips` to get PostScript as an intermediate step (using `ps2pdf` or Acrobat Distiller to get PDF), you can force `dvips` to use Type 1 fonts by specifying the `-Ppdf` flag, as in

```
unix% dvips -Ppdf foo.dvi -o foo.ps
```

4.6 Debugging

One of the trickiest things about using LATEX is interpreting LATEX's sometimes cryptic error messages.

In particular, the line numbers that LATEX reports are often not the line numbers where the problem *is*, but the line numbers where LATEX noticed there was a problem.

One useful way of getting a bit more context to help you understand the problem is to put the line

```
\setcounter{errorcontextlines}{1000}
```

in the preamble of your document, which will provide you with a (perhaps excessive) amount of context for an error.

The most common errors are probably

- Using one of the special characters (see Section 4.1)
- Leaving off or mismatching a brace or bracket
- Leaving out or swapping arguments to a command or environment

If you've tried everything and you can't find the source of an error message, try the following procedure:

1. Create a new file, copying your preamble into it
2. Try typesetting it—if you have an error, the problem is in your preamble
3. If it typesets, copy half of your document's body into the new file, and typeset that
4. If you see your error, then continue halving the document until you narrow it down to the problem section
5. If you don't see your error, try the other half

5 Resources

There are lots of great resources available for using TEX and LATEX. Here are a few (there are also links available online at <http://www.math.hmc.edu/computing/support/tex/>).

5.1 Online Documentation

Much of the documentation for TEX and LATEX is available online, as part of the TEX system. teTEX, the TEX system installed on the math lab computers, includes a script called `texdoc` to access this documentation. All you have to do is type `texdoc` followed by a string that you believe is the name of the document you're looking for. For example, `texdoc booktabs` will give you the documentation for the booktabs package that I used to create the tables in this document.

Unfortunately, `texdoc` only works for documentation that is sensibly named. The authors of the graphics package, for instance, called their manual `grfguide`. Still others decided that `manual` was a good name for their manual (after all, it's the only manual in their distribution).

Sometimes you can find documentation using the `locate` command, which lists all the files on your system that match a string that you provide. For example, you could find `grfguide` by trying `locate graphics` and `grep` ping out the results with `texmf` in them, and passing that list to another `grep` for the string `doc`:

```
unix% locate graphics | grep texmf | grep doc
```

5.2 UK-TUG FAQ

The primary list of frequently asked questions in the TeX world is the UK TUG FAQ, available at <http://www.tex.ac.uk/cgi-bin/texfaq2html>. If you're not sure how to do something, or you've got a problem that you're pretty sure isn't being caused by a typo, check here first.

5.3 `comp.text.tex`

If you can't find an answer in the UK-TUG FAQ, then your next step is to check `comp.text.tex`, the Usenet newsgroup devoted to TeX and LaTeX. Chances are, whatever your problem is, someone else already had it, asked about it on `c.t.t`, and got an answer. Thanks to Google, Usenet's past is preserved in an easily searchable format. Go to Google Groups (<http://groups.google.com/>), type in some search terms, and check out the answers. (If you specify `group:comp.text.tex` at the end of your search terms, you'll only see results from `comp.text.tex`.)

6 Books

<http://www.math.hmc.edu/computing/support/tex/> has some brief reviews of a number of significant books about TeX and LaTeX.

My pick for the best introductory/reference book is the third edition of George Grätzer's *Math into LaTeX*.[3] It's the only book I'm aware of that discusses the latest version of AMSLaTeX in depth. It also has excellent reference tables and a thorough index.

Another book I highly recommend is Lyn Dupré's *BUGS in Writing*. Dupré is one of Addison Wesley's senior editors, and has edited many of the most significant books published by Addison Wesley. *BUGS* is an accessible guide to writing clearly and effectively. It's the kind of book you leave in the bathroom so you'll always have something interesting and amusing to read. Learning how to write better is almost a byproduct!

If you get serious about typesetting, and want to start doing some fancy page design or want to be sure you're using the right kind of type, Robert Bringhurst's *The Elements of Typographic Style* will show you the way.

7 Acknowledgments

Thanks to Darryl Yong, who wrote a similar document in November, 2000, from which I borrowed some ideas.

3. Which I edited.