LaTeX tutorial

Cem Karan
Aylin Çalışkan

July 23, 2010

Contents

Preface

This is an extremely brief tutorial on LaTeX, intended to get you up and running quickly on Eniac. If you are currently reading the PDF version of this file, *be sure* to download the LaTeX version of this file and read it as well. The best way to learn a new language is to read through examples of it, and then to experiment with it, and this tutorial is written with the intention and expectation that you *will* download and read the LaTeX version of the file in addition to the PDF version. Quite a bit of what you read in the PDF file will be unintelligible without the LaTeX file! The LaTeX version is called `LaTeX_Tutorial.tex`. As an aside, the file suffix for LaTeX is usually `.tex`.

1 Resources

The following are excellent resources on LaTeX.

- LaTeX: A Document Preparation System (2nd Edition) This is one of two books that everyone always refers to when they want to learn or do anything with LaTeX. Note that it is the short one; the other, The LaTeX Companion (Tools and Techniques for Computer Typesetting) is much more comprehensive, but also more than you will likely ever want or need.

- LaTeX–A document preparation system

  Not to be confused with the book of the same name, this is the main site for LaTeX. Ignore the LaTeX3 information; the only version available anywhere is LaTeX 2ϵ. Note that the language itself has not changed, so that information is still valid.

- TeX User's Group - TeX is the language that underlies LaTeX. This site is dedicate to both LaTeX and TeX, and has extensive documentation on various packages that you can use, as well as tutorials that you can look through.

- CTAN - The Comprehensive TeX Archive Network. This is where you will go if you are looking for more packages to use. Packages are exactly what they sound like; things that you include via the `\usepackage{}` command. See the start of the `LaTeX_Tutorial.tex` file for a whole list of packages that this file has included. Note that if you include it but don't use it, you're not costing yourself anything. The easiest way to handle packages is to leave all the ones that you use regularly in some LaTeX file, and copy them over. For example, grab the `LaTeX_Tutorial.tex` itself, get rid of the parts you don't need, and use the rest. That's what it's there for! :)

2 Running LaTeX

This tutorial assumes that you are going to run under Eniac, via the command line. There are packages available for Windows and OS X, but you will need to install and learn how to use them yourself; there isn't enough space in this tutorial to explain how to install and use packages on other systems.

Take the LaTeX version of this file and put it somewhere convenient that you have write access to that is reachable from the Eniac. Note that LaTeX generates a lot of intermediate files, so if you want to be able to quickly get rid of everything, it is probably best to create an new, empty directory somewhere, and run everything from there.

Log into Eniac, change directories to somewhere convenient, and extract the tarball that contains this LaTeX file and all of its associated files. Note that the tarball has had all of the temporary files removed; if you plan on using version control, now is a good time to note what files you have, and to compare it to the files you will have after LaTeX is done running. Whatever gets generated by LaTeX can be safely disposed of because LaTeX will regenerate it later on. For the time being, I'm going to assume that you have a directory named `LaTeX_Tutorial` that you are operating out of. If you just untar the tarball, it should create that directory for you.

1. Change directories to the `LaTeX_Tutorial` directory.

2. Run `latex LaTeX_Tutorial.tex` from the command line. This will generate many files, including a `.dvi` file. `.dvi` files are device independent files. They are like the great-granddaddy of PDF, PS, EPS, etc. Once a LaTeX file has been compiled into a `.dvi` file, other tools can convert it into any number of other formats, including what we're going to do, which is PDF.

3. Run `dvipdf LaTeX_Tutorial.dvi`. This will generate the file `LaTeX_Tutorial.pdf`. If you're curious as to what other output forms you can render to, take a look in `/user/bin` for programs starting with `dvi`. There are also a couple of LaTeX programs that you might want to look at as well. They start with the name `latex`.

A note about references, table of contents, indices, etc. In section 7 I'll tell you about the `\label{}` and `\ref{}` commands. These create links within a document, including automatically generated section numbering. The problem is how these references are generated; it requires two passes of LaTeX to get them right. If you see that everywhere you've used the `\ref{}` command you have `??` instead of actual numbers & sections, or if you're missing your table of contents & index, just run LaTeX again. It should fix all the references then. Also, if you type in enough text that some items move to a different page, or if you reorder your sections, you will need to run the `latex` command twice.

A final note about running LaTeX. Given enough time and hacking, your LaTeX file can get large, and you can end up with a lot of auxiliary files, especially if you use anything like GraphViz or other tools to automatically generate images. Create a `Makefile`, and use `make`. This will keep the headaches to a minimum, and it makes it easy to clean your files when you just want to get rid of all the dribbled bits that LaTeX leaves behind. It will also allow you to run the `latex` command twice as mentioned above.

3 Basics

LaTeX uses a model similar to HTML + CSS; you mark up your document, delimiting blocks of text with special markup, which are then interpreted by LaTeX along with some style files to determine how to render your document. There are a number of built-in styles, and you can create your own. Since this is supposed to be a brief tutorial, we won't go into how to write your own style files. The important take-home point is that what you type in your document doesn't look anything like what will be finally outputted. Get used to writing some, and rendering some.

You can break up your document into chapters, sections, subsections, subsubsections. Note that you will only have access to the chapter command if your document is a report. If it is an article, you'll have access to everything except the chapter command. As you might guess, chapters contain sections, which contain subsections, etc. The commands always fall into two types; one which generates a chapter or section number, and one which doesn't. The difference is the * character; unnumbered sections have it, numbered ones don't. In addition, each of the commands accepts an argument that is the name you want to give the sections. As an example, I'm going to create several of these, which will generate a whole bunch of dummy sections. Read `LaTeX_Tutorial.tex` to see how these work; reading the PDF file will make no sense what-so-ever for these examples (everything in §4).

4 This is the start of a numbered section

4.1 This is the start of a numbered subsection

4.1.1 This is the start of a numbered subsubsection

This is the start of an unnumbered section

This is the start of an unnumbered subsection

This is the start of an unnumbered subsubsection

Sections do not have an `\end{}` command; instead, a section will continue until the next sectioning command.

Paragraphs are delimited by leaving a blank line between two blocks of text. If you don't understand what I mean, *start reading the* `LaTeX_Tutorial.tex` *file* ***now***.

Notice that unnumbered sections (ones that have a '*' character) will not show up in the table of contents. If you look at the unnumbered sections above, and look for them in the table of contents, you'll see what I mean.

5 Environments

Unlike many programming languages, LaTeX is not context-free. You can tell it to switch modes of behavior, which change what different characters and commands mean at a moment's notice. These modes are called

environments. Some environments are built-in, while others are supplied by packages that you include via the `\usepackage` command. When you change to a new environment, you're getting a whole new set of rules, options, etc., that you can play with. As a simple example, you can write extremely complicated mathematical equations using LaTeX, but to do so, you must be in the math environment. Environments that come in from other packages are usually well behaved and use the `\begin{}` and `\end{}` markers. You put the name of the environment in the braces (always making sure to match a `\begin{}` with an `\end{}`!) to switch to that environment's mode.

A few useful built-in environments are below.

5.1 List making environments

There are 3 basic ways of making lists: `itemize`, `enumerate`, and `description`. The syntax is very simple; the only thing you can adjust is the `\item` command's optional argument. Below are some examples you can look at. Note that you'll only understand what is going on if you read the `LaTeX_Tutorial.tex` file.

- blah blah blah

+ More blah blah blah

-
  - blah
  - more blah

1. blah blah blah

+ More blah blah blah

2. (a) blah
   (b) more blah

Description blah blah blah

Another description More blah blah blah

stuff 1. blah
2. more blah

5.2 Verbatim environment

There are two ways of providing verbatim text (text that LaTeX will display without trying to interpret any of the characters inside). The first is via `\verb`. This is generally used inline, and has the odd syntax that only the first and last character have to match. That is, you can use `\verb=Foobar=`, `\verb*Foobar*`, `\verb!Foobar!`, or any other non-alpha-numeric character you want to use. The only requirement is that you cannot use that character inside the verbatim part. That is, `\verb===` won't work. This also means that `\verb{=}` won't work, but `\verb{={` will.

The second way is via the `verbatim` environment. This is extremely useful for when you want to put in a code listing, or something else where you don't want LaTeX to mess with the formatting. When you open the environment with `\begin{verbatim}`, the *only* way to close the environment is via `\end{verbatim}`. Anything else will ignored. So:

```
Here are a bunch of random characters:
!@#$%^&*()_+=-[]{}\|'";:/?.>,<
    This line started with some spaces.  \LaTeX\ preserved them (and it ignored the \LaTeX\ command!)
    Note that Latex won't wrap lines for you automatically.  If you're not careful, you'll
    run all the way off the side of the page.
There are 4 tabs to the left of this line
```

One tricky part of the `verbatim` environment is how it handles tabs. Most programs see tabs as either being 4 or 8 spaces; LaTeX defines them as having 0 width. So the line above, `There are 4 tabs to the left of this line` actually does have 4 tabs; LaTeX has just reduced them to 0. The safest way to include code is to make sure that you've replaced all tabs with spaces. Most text editors will let you choose if you want to emulate tabs with spaces, so this shouldn't be a problem.

5.3 Tabular environment

Below is an example of the `tabular` environment. This is the environment you'll be wanting if you want to make tables. The syntax is quite simple.

The vertical pipe characters within the `{|c|c|}` create vertical lines. The letter `c` tells LaTeX that you want whatever is in the columns to be centered. If you used the letters `l` or `r` instead, then that column would be left or right justified instead. The command `\hline` puts in the horizontal line. You can remove any `|` or `\hline` you see, and the table will be fine (just without that particular line).

The one trick to remember is how LaTeX determines the shape of the table; for each `c`, `l`, or `r` it sees, it expects there to be that many columns of data. The data is separated by a `&` character. If your table's columns don't match the number of `c`, `l`, or `r` characters, then you'll get an error. A line is ended with `\\`. Again, forgetting that makes a mess.

Some random text	And some more random text
Some more text	and I'm tired of random text, I want to go home.

5.4 Math environment

Unfortunately, the math environment is not a well-behaved environment. There are at least 5 different ways of writing math equations. Here are the 3 most useful:

5.4.1 $ environment

You can write equations inline like this: $a = n_{\epsilon}^{43}$ by putting the equation you want between $ signs. The equation will not be numbered. If the equation is large, it will make a mess of your lines, so this should be for small, simple equations only.

5.4.2 `equation` and `equation*` environments

If you want to break your equation out, then the `equation` and `equation*` environments are your best choice. The former will have an equation number, while the later will not. Here is an example of each:

$$n = \frac{\infty^6}{\sqrt{\frac{\infty}{6}}} \tag{1}$$

$$n = \frac{\infty^6}{\sqrt{\frac{\infty}{6}}}$$

The advantage of the numbered equations is the the builtin `\ref{}` command can refer to the equation by its number, while the unnumbered equation will be referred to by it's enclosing section. For example, the first equation is Equation 1, while the second is Equation 5.4.2.

5.4.3 `align` and `align*` environments

`align` and `align*` are also math environments, except that they have the ability to format their equations into nicely formatted columns and tables. The alignment method is simple; each line must have the same number of & characters, and those characters are used to align the equations with one another. `\\` is used at the end of each line signal that the next line is a different equation. Here is an example. I'm not going to show you the `align*` environment, it is analogous to the `equation*` environment in how it isn't numbered.

$$4 = 2 + 2 \tag{2}$$

$$E = mc^2 \tag{3}$$

$$\text{A\_long\_variable\_name} = \epsilon * \Delta \tag{4}$$

5.4.4 Some math commands

Note that there are a *lot* of math commands; it is best to look them up on the web. Fig 1 has some of the symbols you can use. Read the `LaTeX_Tutorial.tex` file to see how you use them. Remember that the $ is just there to put LaTeX in math mode; if you're already in math mode, you don't need $.

α	β	γ	δ	ϵ
ε	ζ	η	θ	ϑ
ι	κ	λ	μ	ν
ξ	π	ϖ	ρ	ϱ
σ	ς	τ	υ	ϕ
φ	χ	ψ	ω	

(a) Lower-case Greek

Γ	Δ	Θ	Λ	Ξ
Π	Σ	Υ	Φ	Ψ
Ω				

(b) Upper-case Greek

$\pm$	$\mp$	$\times$	$\div$	$*$
$\star$	$\circ$	$\bullet$	$\cdot$	$\amalg$
$\cap$	$\cup$	$\uplus$	$\sqcap$	$\sqcup$
$\vee$	$\wedge$	$\setminus$	$\wr$	$\diamond$
$\triangle$	$\bigtriangledown$	$\triangleleft$	$\triangleright$	$\oplus$
$\ominus$	$\otimes$	$\oslash$	$\odot$	$\bigcirc$
$\dagger$	$\ddagger$			

(c) Binary operation symbols

$\leq$	$\geq$	$\equiv$	$\models$	$\prec$
$\succ$	$\sim$	$\perp$	$\preceq$	$\succeq$
$\simeq$	$\mid$	$\ll$	$\gg$	$\asymp$
$\parallel$	$\subset$	$\supset$	$\approx$	$\bowtie$
$\subseteq$	$\supseteq$	$\cong$	$\neq$	$\smile$
$\sqsubseteq$	$\sqsupseteq$	$\doteq$	$\frown$	$\in$
$\ni$	$\notin$	$\propto$	$\vdash$	$\dashv$

(d) Binary relation symbols

$\leftarrow$	$\longleftarrow$	$\Leftarrow$	$\Longleftarrow$
$\rightarrow$	$\longrightarrow$	$\Rightarrow$	$\Longrightarrow$
$\leftrightarrow$	$\longleftrightarrow$	$\Leftrightarrow$	$\Longleftrightarrow$
$\mapsto$	$\longmapsto$	$\hookleftarrow$	$\hookrightarrow$
$\leftharpoonup$	$\leftharpoondown$	$\rightharpoonup$	$\rightharpoondown$
$\rightleftharpoons$	$\rightsquigarrow$	$\uparrow$	$\Uparrow$
$\downarrow$	$\Downarrow$	$\updownarrow$	$\Updownarrow$
$\nearrow$	$\searrow$	$\swarrow$	$\nwarrow$

(e) Arrows

$\aleph$	$\prime$	$\forall$	∞	$\hbar$
$\emptyset$	$\exists$	$\Box$	$\Diamond$	$\imath$
∇	$\neg$	$\jmath$	$\surd$	$\flat$
$\triangle$	ℓ	$\top$	$\natural$	$\clubsuit$
$\wp$	$\bot$	$\sharp$	$\diamondsuit$	$\Re$
$\parallel$	$\backslash$	$\heartsuit$	$\Im$	$\angle$
∂	$\spadesuit$	$\mho$		

(f) Misc. symbols

$\sum$	$\bigcap$	$\bigodot$	$\prod$	$\bigcup$
$\bigotimes$	$\coprod$	$\bigsqcup$	$\bigoplus$	$\int$
$\bigvee$	$\biguplus$	$\oint$	$\bigwedge$	

(g) Variable size symbols

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

(h) Log-like functions

(	)	$\uparrow$
[	]	$\downarrow$
$\{$	$\}$	$\updownarrow$
$\lfloor$	$\rfloor$	$\Uparrow$
$\lceil$	$\rceil$	$\Downarrow$
$\langle$	$\rangle$	$\Updownarrow$
/	$\backslash$	
$\vert$	$\Vert$	

(i) Delimiters

Figure 1: Symbols

Superscripts are made via the `^{}` command, and subscripts are made via `_{}`. Fractions are made using the `\frac{}{}` command, square roots are made via the `\sqrt` command, and dots are made with the `\ldots`, `\cdots`, `\vdots`, and `\ddots` commands.

See Eq 5 for an example.

$$n_i = \frac{j^i + k^{s_i}}{a_0 + a_1 + \cdots + a_n} * \sqrt{\pi \times \imath} \tag{5}$$

The symbols in Fig. 1i are unusual in that the size of the symbols depend on what is 'inside' of them. You control what is 'inside' via the `\left` and `\right` commands, which *must* occur in matched pairs. The

way to use them is to put the `\left` or `\right` command immediately before the symbol you're using. See Eq. 6. You can also use a dot, as in `\left.` or `\right.` The dot won't show up in the output, but it allows you to put in the `\left` and `\right` without LaTeX complaining about a missing `\left` or `\right`. This can be handy if you want to make something like Eq. 7

$$\mathcal{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \tag{6}$$

$$\mathcal{B} = \begin{Bmatrix} 1 & 2 & \cdots \\ 4 & 5 & \cdots \\ \vdots & \vdots & \ddots \end{Bmatrix} \tag{7}$$

$$C = \sum_{i=1}^{n} \left(\frac{\Theta_n}{\Lambda^n} \right) \tag{8}$$

6 Figures and graphics

While pure math is great for many things, pictures aren't one of them. To create a picture, you can use the `\includegraphics{}` command as follows:

```
\begin{figure}[htb]
    \centering
    \resizebox{1in}{!}{\includegraphics{FlowChart}}
    \caption{Figure caption}
    \label{FlowChart1}
\end{figure}
```

Which yields the figure in Fig. 2. Note that the `figure` environment is one that I'm not going to explain; the only thing you really need to know is that it forces whatever is inside of it to appear on one page, which is exactly what you want with a picture. If you're interested, search on the web for it, otherwise, just copy the LaTeX above, and replace what you need.

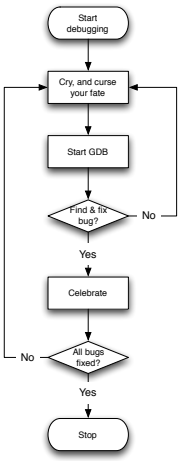
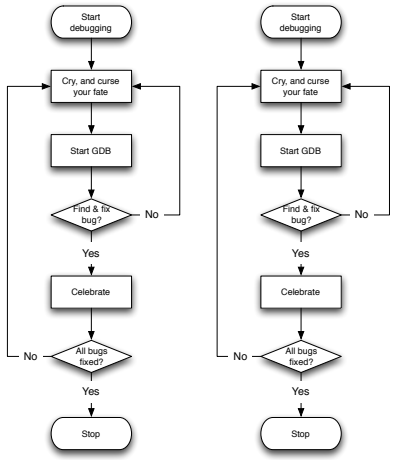


Figure 2: Debugging reality

You can group multiple pictures together using the `\subfloat{}` command as in Fig 3.

```
\begin{figure}[htb]
    \centering
    \subfloat[Left debugging]
    {
        \resizebox{1in}{!}{\includegraphics{FlowChart}}
        \label{FlowChart2a}
    }
    \subfloat[Right debugging]
    {
        \resizebox{1in}{!}{\includegraphics{FlowChart}}
        \label{FlowChart2b}
    }
\caption{Parallel debugging}
\label{FlowChart2}
\end{figure}
```



(a) Left debugging (b) Right debugging

Figure 3: Parallel debugging

The line `\includegraphics{FlowChart}` tells LaTeX to look for the file named `FlowChart` in the same directory as where your LaTeX file is. Note that there isn't a file suffix because at the top of the `LaTeX_Tutorial.tex` file, I've included the command `\DeclareGraphicsExtensions{.eps}` (or possibly `\DeclareGraphicsExtensions{.pdf}`, if I've forgotten to comment out the other line). That command tells LaTeX what the file suffix to search for is.

The `\resizebox{1in}{!}{...}` part tells LaTeX that I want the image to be resized. `\resizebox` uses 3 arguments; the first is the width you want, the second is the height you want, and the third is the thing you want resized (which can be objects other than graphics, but it is unlikely you'll use it for anything else). The `!` in either of the first two arguments means that you want to keep the aspect ratio the same, so your graphic will shrink in size, but it won't get distorted.

The `\label` is described in Section 7. As a general rule of thumb, every `figure` and every `subfloat` should have their own label.

Other arguments are caption text, and getting the figure centered correctly; play with the code to see what is going on.

7 Cross References

To make a reference within a document, you use the `\label{}` and `\ref{}` commands. The `\label{}` command acts like the anchor for a link, while the `\ref{}` command tells LaTeX to make a link to a particular anchor. Note that `\label{}`'s are not exactly like URLs; they will only anchor to the start of a section. If you go through the `LaTeX_Tutorial.tex` file, you will see many uses of both of the commands. Try moving them around a bit, and you'll see where you can put them. You can put any non-special character in for labels, including spaces. The labels can be as long as you wish, so try to make your labels descriptive, it will make it easier to find errors.

8 Miscellaneous

The `\noindent` command right before a paragraph will force the start of the line to be all the way to the left of the page. Look in the `LaTeX_Tutorial.tex` file for examples of how its used.

8.1 Whitespace

There are a number of ways of adding more space to a line, and there are some ways of forcing LaTeX to go backwards. Here are a few space characters:

`\,`	‘ ’
`\;`	‘ ’
‘`\ `’	‘ ’
`~`	‘ ’

`~` is a non-breaking space character. That means that if you have two words next to each other, but separated by `~`, they will be treated as one word for the purposes of line breaking, justification, etc. The practical upshot of this is if you have something like a reference like `Fig.~\ref{some reference}`, you are guaranteed that the `Fig.` part and the generated reference number are always stuck together. This is handy if there is a line break, or a page break as you are guaranteed that you don't end up with the `Fig.` on a different line or page than the reference number.

8.2 Vertical Spacing

You can create a page break via the `\pagebreak` command. Put it on any line by itself, and the page will be broken there.

You can make a line break via the `\linebreak` or `\\` commands, and LaTeX will try to justify the lines. `\newline` simply breaks the line, without trying to justify it.

`\bigskip`, `\medskip`, and `\smallskip` will put space between paragraphs. This can be handy if you want to space a paragraph away from an equation, or some other object.

9 Errors

LaTeX is powerful, but its error messages can range from useful to malignantly misleading. You will get used to error messages on lines that can't possibly have errors on them, only to find that the error occurred somewhere far, far away. The best thing to do is to run LaTeX *often*, and to keep everything under version control. That way, when something breaks, you can do a quick diff between your old and new versions, which should give you an idea as to where the errors could be.

Here are some common problems:

- **Incorrect or missing reference numbers** Try re-running LaTeX. LaTeX uses a two pass algorithm to fill in all the reference details. In the first pass, it builds a database of where the `\labels{}` are, and in the second pass, it actually puts those references into the `\ref{}`s. If that fails, *make absolutely sure* that your references and labels match *exactly*. That usually solves most problems.

- **Complaints about a missing $** You tried to use something that only has meaning inside of the math mode outside of it. The most common cause of this is the '_' character. If you forget to escape it with a \, then it will cause you a headache. Note that the warning is generally on the wrong line, sometimes in the wrong part of your paper all together. The easiest way to fix this is to use your favorite search utility to look for any unescaped '_' characters, and then to try again.

  This can also happen if you accidentally put a `$` inside of another math mode like so:

  ```
  \begin{equation}
  $E = mc^{2}$
  \end{equation}
  ```

  Double check all your math environments to see if you've forgotten anything, or gotten overzealous and added extra environment markers.

- **ANY complaint that occurs beyond the end of the document** Check to make sure that every open brace or bracket is matched with a close brace or bracket. Also make sure that every `\left` is matched with a `\right` and that every `\begin{}` is matched with an `\end{}`. Inevitably, there is something left open that needs to be properly closed off. Also make sure that everything is properly nested; incorrect nesting will cause you headaches as well. Finding it will likely be a hassle because LaTeX won't tell you where your error actually occurred, so it will involved some detective work. Good luck, you're going to need it!

- **I fixed the bug, *why doesn't it work?!?!?*** Throw away all of the intermediate files that LaTeX generated. Those intermediate files are LaTeX's cached state, and if a prior error has mangled that state somehow, LaTeX usually won't be able to recover from it. By throwing away those files, you'll force LaTeX to rebuild from a clean state.