Introduction to TEX
What is \TeX

To \TeX{} or not to \TeX{}, that is the question.

\TeX{} is the typesetting language developed by Donald E. Knuth.

\LaTeX{} is a macro package, originally written by Leslie Lamport.

One way to think about the relation between the two is

\LaTeX{} is your typographic designer, and \TeX{} is the typesetter \[8\]

Most people tend to use \LaTeX{} today. And to confuse things most people use the terms interchangeably.
So, . . . why would I want to use it?

Nothing, Nothing does a better job typesetting mathematics than \TeX.

Using \LaTeX gives you the ability to create with little effort a table of contents, a list of figures, a list of tables, an index references and bibliography that keep themselves up-to date.
So what doesn’t it do?

Write your thesis for you.
Your first \LaTeX{} document

The following are reserved characters in \LaTeX{}

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

and are used only in \LaTeX{} commands.

A comment on quotes. \TeX{} is smart as it knows the difference between a left and right quote.

For example you type ‘‘this’’ to get “this”.

The basic form for \LaTeX{} commands is

\texttt{\textbackslash command[options]{argument}}

The basic form for a \LaTeX{} environment is

\texttt{\textbackslash begin\{environ\}[options]} ...
\texttt{\textbackslash end\{environ\}}
Basic \LaTeX Document

\% lines that start with a \% are comments
\documentclass[options]{class}
\% Load any packages with the command
\% \usepackage[options]{packagename}
\% Everything between \documentclass
\% and \begin{document} is called the
\% preamble. This is where you define
\% macros and load packages.
\begin{document}
\% This is the body of your document.
\end{document}

For the documentclass argument you usually have article or report.

For options to the document class you can set things like the page size or the base font size (the default is 10pt).
How do I write a $\text{T}_{\text{E}}\text{X}$ document?

Any text editor such as emacs, pico, or vim will do.

Use a specialized editor that is designed with $\text{T}_{\text{E}}\text{X}$ in mind such as Winedt, Kyle, or emacs.
Seeing what things look like. (part 1)

As mentioned earlier \TeX{} is a typesetting language. Our \TeX{} file is just a big collection of typesetting commands, we \TeX{} or \LaTeX{} our source file to generate the output. \TeX{}’s output that you view or print is a dvi file.

In order to generate the dvi file you \LaTeX{} your source file. How that is done depends on a lot, the OS you are using, what editor you are using \textit{etc}.

- If you are using Winedt or Kyle, there is a button you click on to \LaTeX{} the file you are currently editing.

- If you are under UNIX you can type \texttt{latex filename.tex} from the command line.
What should happen is you will see a window print out a bunch of stuff that you don’t really need to worry about. But more times than not you will get a cryptic error message and prompt asking you what to do. Depending on the error you need to decide if you want to stop, skip this one error and keep going until the next one or keep going irregardless.

After \LaTeX \footnote{Typeset by Foil\TeX} finishes it run, go back to the source file and fix things and re-\TeX it. Get used to dong this a \textit{lot}.

Helpful hint: Sometimes you get errors/warnings that are a bit too cryptic and it is useful ignore all the errors and view the dvi file in order to see what went wrong.
Seeing what things look like. (part 2)

Again this is very dependent on a number of things.

• If you are using Winedt or Kyle, there is a button you click on to open the dvi file of the file you are currently editing.

• If you are under UNIX you can type kdvi filename.dvi or xdvi filename.dvi from the command line.
Converting to other formats

Quite often you want to put a \TeX\ file on the web. As most people have no idea what a dvi file is, you will want to convert it into something else.

- If you are using Winedt or Kyle, there is a button you click on to run pdflatex on your file.

- If you are under UNIX pdflatex filename

  Probably the easiest way is to use pdflatex. Again this is very dependent on a number of things.

  The only problem is that it doesn’t always work. So, you can use a more general method which uses dvips and ps2pdf. After running \LaTeX\ on filename.tex:

  - If you are using Winedt or Kyle, there is a button you click on to run dvips and another button for ps2pdf
• If you are under UNIX use the commands dvips -Ppdf filename then ps2pdf filename.ps
Text Basics
Paragraphs

\TeX{} uses a blank line as a delimiter between paragraphs.
\textbf{Fonts}

\LaTeX\ is \textit{not} a word processor; you mark things up logically. You don’t change fonts often.

Here is a table of the common commands to change the appearance of text:

\begin{verbatim}
\textup{foo}    foo
\textit{foo}   \itshape{foo}
\textsl{foo}   \itshape{foo}
\textsc{foo}   \itshape{foo}
\textbf{foo}   \bfseries{foo}
\textrm{foo}   \textnormal{foo}
\textsf{foo}   \texttt{foo}
\texttt{foo}   \texttt{foo}
\end{verbatim}

Table 1: Some common font commands
One will probably use 	exttt{\textit{emph}} more than anything. One nice thing about \texttt{\textit{emph}} is that it will do the right thing based on context.

For example \textit{This is emph in emphasized text.}

For example \texttt{\textit{This is \textit{emph\{emph\}} in emphasized text.}}
Lists

\LaTeX has two common types of lists: enumerate and itemize. Both allow nesting up to more levels that most people will need.

Here are a couple of examples:

\begin{itemize}
\item foo
\item bar
\item foobar
\end{itemize}

Produces:

- foo
- bar
- foobar
While

\begin{enumerate}
  \item foo
  \item bar
  \item foobar
\end{enumerate}

Produces

1. foo

2. bar

3. foobar
Tables

Tables are one of those things that can get very hard in a hurry. We will keep life simple and stick with a very basic table and give an example of some of the fancy things that can be done.

The goal is to make a table with 2 rows and 3 columns with the entries in each column to be left, center and right justified respectively. We also want to place a line separating each row and each column.
The code

\begin{table}[hbt]
  \centering
  \caption{A simple table}
  \begin{tabular}{|l|c|r|}
    \hline
    foo & bar & foobar \\ \\
    \hline
    foobar & bar & foo \\ \\
    \hline
  \end{tabular}
  \caption{A simple table}
\end{table}

Produces

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>foo</td>
<td>bar</td>
<td>foobar</td>
</tr>
<tr>
<td>foobar</td>
<td>bar</td>
<td>foo</td>
</tr>
</tbody>
</table>

Table 2: A simple table
Introduction to Math

\LaTeX{} does a great job typesetting math unfortunately you spend a fair amount of time entering math.

The AMS has a number of packages that predefine a large number of commands, environments and symbols. I would recommend you place the following in the preamble of your document:

```
\usepackage{amsmath}
\usepackage{amsthm}
\usepackage{amssymb}
```

You can check out the manuals for the packages at http://www.ams.org/tex/.
Math modes

\TeX\ has two math modes: inline and display. From the names it should be fairly clear inline is used in a line of text while display is used for display.

Each has its own way of doing things \textit{e.g.}, where the limits of large operators, \( \int \), \( \sum \), are placed.
Examples

The easiest thing to do is look at examples.

Here is some $x^2$ inline math.

Here is some $$x^2$$ display math.

Here is some $x^2$ inline math.

Here is some $x^2$ display math.
\[ 2 + \log e^{-1} = \]
$e^{i \pi} + 1 =$

$e^{i \pi} + 1 =$
\[ \int_{0}^{\infty} e^{-\alpha} d\alpha = \]
\[ \int_{0}^{\infty} e^{-\alpha} \, d\alpha = \]
\[ \int_{0}^{\infty} e^{-\alpha} \, d\alpha = \]
Does
\[ \sum_{i=1}^{\infty} \frac{\sqrt{n+1}-\sqrt{n}}{n} \] converge or diverge?

Does
\[ \sum_{i=1}^{\infty} \frac{\sqrt{n + 1} - \sqrt{n}}{n} \] converge or diverge?
\begin{equation}
\text{If } t > 1 \text{ then } z = \cos \theta + t > 0
\end{equation}

If $t > 1$ then $z = \cos \theta + t > 0$ \hfill (1)
Let $f: \mathbb{R} \rightarrow \mathbb{R}$ be defined by
\begin{equation}
\begin{cases}
0 & \text{for } x \leq 0, \\
1 & \text{for } x > 0.
\end{cases}
\end{equation}

Let $f: \mathbb{R} \rightarrow \mathbb{R}$ be defined by

$$f(x) = \begin{cases}
0 & \text{for } x \leq 0, \\
1 & \text{for } x > 0.
\end{cases}$$

(2)
\begin{equation*}
\begin{split}
\int \log x \, dx &= x \log x - \int x \, d \log x \\
&= \ldots \\
&= x \log x - x
\end{split}
\end{equation*}
Graphics and Color
The topics of graphics in \LaTeX is fairly extensive. As most of us use Maple, Mathematica, Matlab, or xfig to generate encapsulated postscript (EPS) files. I will discuss one way to include those files.

For those of you in graph theory or related areas you may wish to talk at some point with Jamie Radcliffe about MetaPost.

To import eps files I would suggest the graphicx package. The basic syntax is

\begin{verbatim}
% place in the preamble
\usepackage[dvips]{graphicx}
\includegraphics[key=value]{filename.eps}
\end{verbatim}

The most common keys are height and width. Note only one of the two is needed, as the other is automatically scaled.
Here is an example using the figure enviroment.

\begin{figure}[h]
\begin{center}
\includegraphics[height=1.5in]{Mathematica_logo.eps}
\caption{Mathematica Logo---Stellated Icosahedron}
\end{center}
\end{figure}

will generate:

Figure 1: Mathematica Logo—Stellated Icosahedron
With $\LaTeX$ you can even put spiffy color in your documents.

In order to use color in your document you will need the following line in the preamble:

\usepackage[dvips]{color}

To use colors, you can use one of two commands. One uses

\color[model]{spec}

for larger portions of text, while one uses

\textcolor[model]{spec}{text}

for smaller portions of text.

For mode, you have choices of RGB, CMYK, or grey. For example, to get red you would have \textcolor[rgb]{1,0,0}{red}. 
Some colors are so common that they are predefined: red, green, blue, yellow, cyan, magenta, black, and white. To use these you remove the model option, and just use the color name for the spec. For example, \textcolor{green}{green} is generated by \textcolor{blue}{green}

Or if you want to use a custom color over and over, you may want to define it. So to get a \textcolor{purple}{purple color}, you would use

\textcolor{purple}{purple color}

with \definecolor{purple}{cmyk}{0,.41,0,.48} in your preamble.
PSfrag

I am not even going to talk about how to use this package. But I want to make you ware of its existance and what it does. What is does is best seen from an example.

![Figure 2: Without psfrag](image)

![Figure 3: With psfrag.](image)
References and \texttt{BIBTEX}
A nice feature of \LaTeX{} is its ability to do automatic references. In order to use this feature one needs two commands

\label{key}

and

\ref{key}

One uses \texttt{\label} to set the reference and uses \texttt{\ref} to print the number of that object. The key \textbf{must} be unique.
First we must set the label
\section{Math}\label{math}

We will get:

Please see section 3.

When you use

Please see section \ref{math}.

after \emph{multiple} runs of \LaTeX.
**BibTEX**

BibTEX is by far one of the most useful packages for use with LaTeX. With BibTEX you can have all of your references in a couple of files that can be used by each of your documents.

To use BibTEX you will need a bibliography database or a set of them, and one needs to add the following lines to your document where the bibliography is to begin

\begin{verbatim}
\bibliography{file1, file2,...}
\bibliographystyle{style}
\end{verbatim}

where file1, file2, ... are a list of bibliography databases to use and style is plain, alpha or any number of choices.

To cite an entry you use \cite{key}, where the key matches that entry in the bib file.
Now for the tricky part: getting \textsc{BibTeX} to play nice. You first need to \textsc{LaTeX} your document a couple of times, then run \textsc{BibTeX} on filename.aux. From a command line this is done by

\texttt{bibtex filename}

Then \textsc{LaTeX} needs to be run again. You may need to repeat this a few times to get everything right.

So, what do these \textsc{BibTeX} files look like, well let’s go look at http://www.ams.org/mrlookup.

After looking up your sources, one now needs to copy that information to a file. The file needs to be saved as filename.bib.
Large Documents and UNL Thesis Class
Large Documents

\LaTeX\ has several commands that help you keep your sanity in writing your thesis.

It is recommended that you have your main document import selected smaller documents. The command to do this is \texttt{\include\{file\}}.

Another useful command is the

\texttt{\includeonly\{file\ list\}}
For example your main file would look something like:

\documentclass{article}
\includeonly{chapt2}
\begin{document}
    \tableofcontents
    \listoffigures
    \listoftables

    \include{chapt1}
    \include{chapt2}
    \include{chapt3}

    \bibliography{file1}
    \bibliographystyle{alpha}
\end{document}
The above will only process the \filename{chapt2} but will read in the aux files for the other chapters. It will use this information for page numbering, etc.

This can get confusing as the page numbers and refs will tend to get out of sync. To re-sync all the information, you just need to comment out the \includeonly line and \LaTeX{} the document a few times.

Something to note is that each include command will start a new page with the information in that file.
The commands

\texttt{\tableofcontents}
\texttt{\listoffigures}
\texttt{\listoftables}

have \LaTeX \ generate a special file with that will be read back in on the next run that will produce the table of contents, a list of figures and list of tables.
As you may or may not know, Allan Donsig and I have written a class file that makes writing a thesis a little bit easier.

The class file takes care of readjusting the margins, page number placement, page numbers on the pages that should have them, title and abstract pages that are formatted correctly.

All one needs to do is to enter various information into various commands and type your thesis.

The package can be downloaded from either Allan’s web page or mine.
Slides

A number of us will give a talk at some point. As you have spent all of this time entering things in \LaTeX do you really want to re-enter it for use in PowerPoint?

As I’m guessing not, you will be glad to know there is a very nice package for making slides. I would suggest foiltex for two reasons

- it’s very straightforward to use
- it’s fairly customizable.
The basic document using foiltex is

\documentclass[]{foils}
\begin{document}

% start a new slide
\foilhead{Heading}

% make new slides as needed automatically.

Text...

% start a new slide.
\foilhead{Heading}

% make new slides as needed automatically

Math...

\end{document}
Macros

Macros are probably one of the coolest and most frustrating things in \TeX. With them you can greatly simplify your document by defining various logical structures in your document to produce a more consistent-looking document, which will also be more easy to maintain. In addition, keeping your own macros in a separate file, you can input them into each of your other documents so all of your documents can use the same macros.
Let’s look at a simple example. Very often we deal with number systems. We want the look and feel of these concepts to be the same in a document. We have two options

1. make a choice now and use it

2. define a macro

The first option has a very large problem with it (for large documents). Say you like to use bold letters for the numbers systems e.g., $\mathbf{R}$ for the reals. But a month or so later your adviser (or you) thinks that things would look better with double struck letters, $\mathbb{R}$. With option 1, you must go back by hand and change each and every bold letter that was used for the reals, integers, etc. With option 2, you just change the definition and that’s it.
Let’s see how to set up a macro for number systems.

\newcommand{\numbersys}{\mathbb{#1}}

% Define each of our numbers systems
% in terms of numbersys.
\newcommand{\complex}{\numbersys{C}}
\newcommand{\reals}{\numbersys{R}}
\newcommand{\rationals}{\numbersys{Q}}
\newcommand{\ints}{\numbersys{I}}
\newcommand{\naturals}{\numbersys{N}}

Using \reals now produces $\mathbb{R}$. 
References


