Intro to Latex

Tutorial Source Credit:

Gilbert Bernstein
September 21, 2008

What is Latex good for?

- LATEX is by far the easiest way to type mathematics into a computer.
- BIBTEX is one of the best choices for managing and typesetting bibliographies.
- LATEX has great facilities for organizing a document into sections and chapters. You can auto-generate a table of contents and internal page numbers for cross-references.
- Unfortunately, if you want a lot of control over the placement of your figures or diagrams, you will find LATEX infuriating. The vast majority of users can make do with the semi-automatic diagram placement rules.

Installing Latex

Sometimes this can be a bit tricky. Google will be your best friend here. On Windows, MikTeX is a good option. On MacOS, MacTeX. Linux has its own unique issues, but the texlive family of packages and texmaker should work well. Alternatively, there are currently two websites which allow you to edit and typeset LATEX in a web browser, https://www.sharelatex.com/ and https://www.writelatex.com/. These are a great way to get started quickly, but typesetting on them runs very slowly. The computers in the CS labs and math lab all have LATEX pre-installed on them.

Hello, World!

Traditionally, the first program you write when learning a new programming language outputs the phrase “Hello, World!” and then quits. As you will learn, LATEX has a few similarities to programming languages, so it is only fitting that your first LATEX document be “Hello, World!” Unlike many programming languages, there is very little cruft between you and your output:

HelloWorld.tex

Document Type	\documentclass{article}
Preamble (empty here)	
Beginning of Document	\begin{document}
Document Type	Hello, world!
End of Document	\end{document}

However, you will have to spend some time setting up your “environment”. If you are using a Macintosh, I recommend trying the program TeX Shop. On Windows, things will be a tad more difficult. You will certainly need to get MikTeX, ghostscript, and ghostview (or equivalents). After that, you may want to install some editor to make your life easier. If you do so, I strongly recommend against using a WYSIWYG editor while learning LATEX, since it will largely prevent you from learning. If you use Linux, you should be able to figure out this part on your own.

Once you have your environment set up, create a new document. It will be a plain-text file, like you would get in notepad or a similar such program. TEX documents are typed up in plain-text and then fed through LATEX proper to produce the final typeset document, so you should get used to this sort of “compiling” work-flow. When editing a TEX document you should periodically re-compile to make sure your changes haven’t broken your document. Syntax errors can prevent successful typesetting. For instance, try removing any of the parts of HelloWorld.tex as shown above.

IMPORTANT! Get HelloWorld.tex (shown above) to successfully compile before you proceed.

Basic Text

When proceeding through this tutorial, you should compile the examples and try modifying them. Learn by doing.

Basic Rules

In Latex you can type most anything you want as a block of text, and it will be typeset as you would expect. However, there are some special characters which will not behave as you would expect. This is because they have some other, special meaning.

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

We’ll return to the special characters later and tell you how to typeset them safely, but for now, just avoid them.

```
Isn’t that logo cool? You can typeset LATEX using the backslash. When
a word is prefixed by a backslash LATEXtreats it as a command.
Commands can do many things. For instance, notice that second
backslash at the end of the first LATEX. When we omitted it, we got
funky spacing... Isn’t that logo cool? You can typeset \LaTeX\ using
the backslash. When a word is prefixed by a backslash \LaTeX treats
it as a command. Commands can do many things. For instance, notice
that second backslash at the end of the first \LaTeX. When we omitted
it, we got funky spacing...
```

Whitespace & Paragraphs

At first, it seems like LATEX likes to boss you around with whitespace. This is because LATEX is trying to automatically format and layout your text. The philosophy is that—to the greatest extent possible—you should focus on the content when writing, rather than the form.

Whitespace Rules

LATEX ignores extra whitespace. No matter how many spaces you add, you just get one space. If you want a new paragraph, you need to leave a blank line. Extra blank lines won’t make any difference.

```
\LaTeX\ ignores extra       whitespace. No matter how       many
spaces you add, you just get one space. If you want a   new
paragraph,             you need to leave a blank line.
         Extra blank lines won’t make any difference.
```

This may seem like an odd choice nowadays, but back in the day, requiring a blank line was a very effective way to denote a new paragraph. Lines in text files used to have a maximum length of 80 characters. So, paragraphs would be explicitly broken up into these lines by placing carriage returns/new-lines. Because TEX was written back in the 70s and 80s, it was designed to accommodate such quirks.

Controlling Whitespace

As we saw with the symbol, LATEX, we sometimes want or need to give TEX more explicit instructions about whitespace.

We can use the backslash followed by a space, possibly in series to produce extra whitespace characters. We can also use tildes, so that the whitespace can't be used as a line break. We can keep paragraphs from being indented too. Or, we can ask for some horizontal space, if we need it. More frequently useful, however, is vertical space 6 which is great if we want to leave room for hand-drawn diagrams. We can also force line-breaks with double backslash, although this is generally bad form.

```
As we saw with the symbol, \LaTeX, we sometimes want or need to give
\TeX\ more explicit instructions about whitespace. We can use the
backslash followed by a space, possibly in series\ \ \ to produce
extra whitespace characters.
We~can~also~use~tildes,~so~that~the~whitespace~can't~be~used~as~a~lin
e~break.
\noindent We can keep paragraphs from being indented too. Or, we can
ask for some horizontal\hspace{1in}space, if we need it. More
frequently useful, however, is vertical space
\vspace{1in}
\noindent which is great if we want to leave room for hand-drawn
diagrams.\\ We can also force line-breaks with double backslash,
although this is generally bad form.
```

You can append an asterisk to the \vspace{} or \hspace{} commands in order to tell LATEX that you absolutely must have the requested space. If you don't do this, \vspace{3in} occurring at the end of a page may only get half an inch. By contrast \vspace*{3in} will always get 3 inches—on the next page if need be.

Punctuation, Symbols, and Font Style

Quotation Marks

TEX allows you extra control over quotation marks. My friend told me, "You can use the left 'accent' mark to produce a left quotation, whereas the normal apostrophe functions as a right quotation. If you place two of a kind in sequence, you get a double quote. Maybe then you'll be able to see what I'm saying."

```
\TeX\ allows you extra control over quotation marks. 7 My friend told
me, ``You can use the left `accent' mark to produce a left quotation,
whereas the normal apostrophe functions as a right quotation. If you
place two of a kind in sequence, you get a double quote. Maybe then
you'll be able to see what I'm saying.''
```

Dashes and Hyphens

Dashes come in three varieties: hyphens, en-dashes, and emdashes. All three use the short dash or “minus sign” character. One dash gives you a hyphen, for use in forming compound words. Two dashes gives you an en-dash, useful for specifying ranges of page numbers 2–3 and the like. However, if you want to punctuate your interjections—I know I do—then you should use 3 dashes for an em-dash. The phrases ‘en-’ and ‘em-’ dash are taken from the varying width of an ‘n’ and ‘m’ in a typeface/font.

```
Dashes come in three varieties: hyphens, en-dashes, and em-dashes.
All three use the short dash or ``minus sign'' character. One dash
gives you a hyphen, for use in forming compound words. Two dashes
gives you an en-dash, useful for specifying ranges of page numbers
2--3 and the like. However, if you want to punctuate your
interjections---I know I do---then you should use 3 dashes for an
em-dash. The phrases `en-' and `em-' dash are taken from the varying
width of an `n' and `m' in a typeface/font.
```

Special Characters

For most of the special characters,

```
{ } $ % # &
```

you can typeset the character by adding a backslash beforehand. In this sort of an idiom, the backslash is often referred to as an escape sequence, because it is being used to escape from the normal meaning of the following symbol.

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

The three you cannot get this way are \, ^ and ~. You can typeset these three using the following phrases, which we will not explain.

```
$\backslash$ \verb|^| $\sim$
```

You can also get quite a few other special characters using commands:

œ, Œ	\oe, \OE
æ,Æ	\ae, \AE
å,Å	\aa, \AA
ø, Ø	\o, \O
ł,Ł	\l, \L
ß	\ss
¿	?’
¡	!’
†	\dag
‡	\ddag
§	\S
¶	\P
©	\copyright
£	\pounds
Ì	\i
ȷ	\j

Accents

LATEX provides a very convenient mechanism for placing accents on characters, making it as simple to play pi˜nata as it is to add excessive ¨uml¨a¨uts to your band name.

```
 \LaTeX\ provides a very convenient mechanism for placing accents on
characters, making it as simple to play pi\~nata as it is to add
excessive \"uml\"a\"uts to your band name.
```

é	\e’
ō	\=o
o͡o	\t{oo}
è	\’e
ȯ	\.o
ê	\c{c}
ǒ	\^e
o̯	\u{o}
ő	\’o
č	\v{c}
o̲	\b{o}
ñ	\~n
ő	\H{o}

Font Style

If you want to really emphasize your words or phrases, you can enclose them in an emphasis command. Unlike most of the commands you’ve seen up till now, emphasize, takes an argument. The following commands with arguments allow you to control the appearance of your typeface in rather standard ways.

```
If you want to really \emph{emphasize} your words or phrases, you can
enclose them in an emphasis command. Unlike most of the commands
you’ve seen up till now, \emph{emphasize}, takes an \emph{argument}.
The following commands with arguments allow you to control the
appearance of your typeface in rather standard ways.
```

To begin with, there are three basic types of fonts you can select: Roman, with nice strong serifs, Sans-serif, sleek, modern, and elegant, and Monospace, mechanical and reliably spaced. Roman is the default, since serifs make it easier to read large blocks of text—or so I’m told.

```
To begin with, there are three basic types of fonts you can select:
\textrm{Roman, with nice strong serifs}, \textsf{Sans-serif, sleek,
modern, and elegant}, and \texttt{Monospace, mechanical and reliably
spaced}. Roman is the default, since serifs make it easier to read
large blocks of text---or so I’m told.
```

To modify these fonts, we can make them bold, italic, or slanted. We can combine these commands to get bold, slanted text, or other combinations. We can even make upright, mediumweight text with special commands, but there’s no real need since that’s just the default.

```
To modify these fonts, we can make them \textbf{bold},
\textit{italic}, or \textsl{slanted}. We can combine these commands
to get \textbf{\textit{bold, slanted}} text, or other combinations.
We can even make \textup{upright}, \textmd{medium-weight} text with
special commands, but there’s no real need since that’s just the
default.
```

If you want to modify large, possibly multi-paragraph blocks of text, you should use enclosing curly braces, and drop the ‘text’ prefix on the command. When you do this, you’re using the curly braces to

give you a so called local scope, so that the formatting command has no effect outside this scope. If you didn't do this, the rest of your document following the command would be sans-serif. Try it!

```
{\sf If you want to modify large, possibly multi-paragraph blocks of
text, you should use enclosing curly braces, and drop the 'text'
prefix on the command. When you do this, you're using the curly
braces to give you a so called \emph{local scope}, so that the
formatting command has no effect outside this scope. If you didn't do
this, the rest of your document following the command would be
sans-serif. Try it!}
```

Environments

Up till now, we've seen two major kinds of LATEX commands: plain commands, like \LaTeX and commands requiring an argument, like \emph{}. Now, we'll introduce a third kind of command, which uses the \begin{} \end{} idiom. You've seen this idiom before in the \begin{document} and \end{document} commands that begin and end the main text section of a LATEX document.

Quote and Quotation

If we want to include a long quote in a document, it is proper to indent this quotation.

The quote environment is good for short quotes.

Just like that.

However, the quotation environment is more appropriate for long, multi-paragraph quotes, since it will better indent these long quotations. See what I mean?

```
If we want to include a long quote in a document, it 11 is proper to
indent this quotation.
\begin{quote}
The quote environment is good for short quotes.
\end{quote} Just like that.
\begin{quotation}
However, the quotation environment is more appropriate for long,
multi-paragraph quotes, since it will better indent these long
quotations.

See what I mean?
\end{quotation}
```

Lists

LATEX provides three different standard types of list:

1. enumerate for numbered lists
2. itemize for bulleted lists
3. description for named lists, eg. lists of definitions

```
\LaTeX\ provides three different standard types of list:
\begin{enumerate}
 \item \texttt{enumerate} for numbered lists
\item \texttt{itemize} for bulleted lists
\item \texttt{description} for named lists, eg. lists of definitions
\end{enumerate}
```

LATEX provides three different standard types of list:

- enumerate for numbered lists
- itemize for bulleted lists
- description for named lists, eg. lists of definitions

```
\LaTeX\ provides three different standard types of list:
\begin{itemize}
\item \texttt{enumerate} for numbered lists
\item \texttt{itemize} for bulleted lists
\item \texttt{description} for named lists, eg. lists of definitions
 \end{itemize}
```

LATEX provides three different standard types of list:
enumerate for numbered lists
itemize for bulleted lists
description for named lists, eg. lists of definitions

```
\LaTeX\ provides three different standard types of list:
\begin{description}
\item[enumerate] for numbered lists
\item[itemize] for bulleted lists
\item[description] for named lists, eg. lists of definitions
\end{description}
```

Tables

Tables are generated using the tabular environment

r right justified
l left justified
c centered
| vertical line

```
\begin{tabular}{rl}
\texttt{r} & right justified \\
\texttt{l} & left justified \\
\texttt{c} & centered \\
 \texttt{|} & vertical line
\end{tabular}
```

Tables are odd in that they take a second argument to the begin command. This second argument states what the horizontal format of the table is going to look like; what columns there are going to be. The table above describes these choices.

Column breaks are denoted by & ampersands and ends of lines by \\, double backslash. Horizontal lines may be inserted using the \hline command. For instance,

	Therapists	Potent Potables	Famous Titles
row 1	$100	$100	$100
row 2	$200	$20	$200
row 3	$300	$300	$300

row 4	$400	$400	$400
row 5	$500	$500	$500

```
\begin{tabular}{|r||c|c|c|}
\hline
& Therapists & Potent Potables & Famous Titles \\
\hline
\hline
row 1 & \$100 & \$100 & \$100 \\
\hline
 row 2 & \$200 & \$200 & \$200 \\
\hline
row 3 & \$300 & \$300 & \$300 \\
\hline
row 4 & \$400 & \$400 & \$400 \\
\hline
row 5 & \$500 & \$500 & \$500 \\
\hline
\end{tabular}
```

Comments

Sometimes we want to leave notes for ourselves, or we want to remove parts of the document that we might want to put back later. Using comments, we can prevent text from being rendered/typeset. But, if we wait until halfway through the line,

```
Sometimes we want to leave notes for ourselves, or we want to remove
parts of the document that we might want to put back later. Using
comments, we can prevent text from being rendered/typeset. % The
percent symbol creates single line comments. But, if we wait until
halfway through the line, % then we only comment the rest of the
line.

\begin{comment}
But, if we want to take out large sections of text all at once, we
should really use the comment environment. This is much more
efficient than going through and placing a percentage symbol at the
beginning of every
\end{comment}
```

Verbatim and Code

If you want to typeset computer code the easiest way to do this is with the verbatim environment. However, there's one extra step you have to do first. We need to add a declaration to the preamble (That's the part of your .tex document between the \documentclass{} command and the \begin{document} command, as indicated in the HelloWorld.tex example file). Just add the command

```
\usepackage{verbatim}
```

somewhere in the preamble. Then you can produce code like this:

```
// Example.cxx
#include <iostream>
using std::cout;
using std::endl;
int main() {
      cout << "Hello, World!" << endl;
      return 0;
}
```

This is done by enclosing the code in \begin{verbatim} and \end{verbatim}. In fact, this is how I've been typesetting the example TEX source, but you get some self-referential problems if you try to use verbatim to typeset a use of verbatim.

Mathematical Text

Math Mode

LATEX's great strength is it's mathematical typesetting. This is accomplished by entering the so called *Math mode*.

The recursive relation $f(n) = f(n-1) + f(n-2)$ defines the Fibonacci Numbers when $f(0) = 0$ and $f(1) = 1$.

```
The recursive relation $f(n)=f(n-1)+f(n-2)$ defines
the Fibonacci Numbers when $f(0)=0$ and $f(1)=1$.
```

Sometimes you want to center mathematical formulas, rather than just place them in-line. This can be done by using one of two different notations, both of which produce the same result:

The recursive relation

$$f(n) = f(n-1) + f(n-2)$$

defines the Fibonacci Numbers when $f(0) = 0$ and $f(1) = 1$. .

```
The recursive relation
$$f(n)=f(n-1)+f(n-2)$$
defines the Fibonacci Numbers when $f(0)=0$ and $f(1)=1$.

 The recursive relation
\[
f(n)=f(n-1)+f(n-2)
\]
defines the Fibonacci Numbers when $f(0)=0$ and $f(1)=1$.
```

Math Fonts

In math mode, letters are italicized by default, since they are assumed to represent variable names like x and y. However, sometimes we would like to use alphabetic characters in other ways, or modify their appearance.

In this book, we will use boldface lower-case letters for vectors, ex. v, boldface upper case letters for matrices, ex. **A**, and regular lower-case letters for scalars, ex *c*. Together, $\mathbf{w} = c\mathbf{Av}$.

```
In this book, we will use boldface lower-case letters for vectors,
ex. $\mathbf{v}$, boldface upper-case letters for matrices, ex.
${\mathbf A}$, and regular lower-case letters for scalars, ex $c$.
Together, $\mathbf{w}=c\mathbf{Av}$.
```

You can use a special cursive font, but only for capital letters: $\mathcal{P}(X)$. You can also typeset some capitals in "blackboard" font, although this requires including the amsfonts package. For example, $\mathbb{R}$ is the same cardinality as $\mathcal{P}(\mathbb{N})$.

```
You can use a special cursive font, but only for capital letters:
$\mathcal{P}(X)$. You can also typeset some capitals in
``blackboard'' font, although this requires including the
\texttt{amsfonts} package. For example, $\mathbb{R}$ is the same
cardinality as $\mathcal{P}(\mathbb{N})$.
```

Apostrophes and Primes

In math mode, apostrophes do not create quotations. Instead they create "primes" as in x' , x'' and x'''.

```
In math mode, apostrophes do not create quotations. Instead they
create ``primes'' as in $x', x''$ and $x'''$.
```

Subscript and Superscript

When doing mathematics, we often want to place superscripts and subscripts on our characters. For instance, $x^2 = xx$, and (x_1, x_2, x_3). If we want to put more than one character in the super or subscript, then we need only group together symbols in curly braces: e^{ix} and x_{i+1}.

```
When doing mathematics, we often want to place superscripts and
subscripts on our characters. For instance, $x^2=xx$, and
$(x_1,x_2,x_3)$. If we want to put more than one character in the
super or subscript, then we need only group together symbols in curly
braces: $e^{ix}$ and $x_{i+1}$.
```

Summation, Integrals, and Limits

When we typeset summations, integrals and limits, we would like to place text beneath and/or above. We can do this using superscript and subscript notation.

$$\sum_{i=0}^{n} x_i$$

However, if this is done inline, $\sum_{i=0}^{n} x_i$, then the superscript and subscript are typeset as such, so that the formula doesn't disrupt the line spacing on the page.

```
When we typeset summations, integrals and limits, we would like to
place text beneath and/or above. We can do this using superscript and
subscript notation.
$$\sum_{i=0}^nx_i$$
However, if this is done inline, $\sum_{i=0}^nx_i$, then the
superscript and subscript are typeset as such, so that the formula
doesn't disrupt the line spacing on the page.
```

Symbols

Assorted

For brevity, I will not include a complete list of symbols here. There are many good symbol lists you can find online. Instead, I'll note a couple of symbols of common use or particular interest

∀	\forall	∃	\exists	¬	\neg	⇔	\iff
∪	\cup	∩	\cap	⊆	\subseteq	∅	\emptyset
≤	\leq	≥	\geq	≠	\not=	≠	\neq
⟨	\langle	⟩	\rangle	⌋	\lfllor	⌋	\rfloor
∇	\nabla	∂	\partial	∫	\int	∞	\infty
∑	\sum	×	\imes	∘	\circ	≈	\approx
→	\rightarrow	⇒	\Rightarrow				

Greek Letters

Since mathematical formulas often contain greek letters, TEX provides a simple system. A complete list of greek letters, like a list of mathematical symbols, can be found many places online.

The Euler phi function is, rather obviously denoted φ(n), although one will often see the variation ϕ(n) appear in mathematics texts. When the upper case Greek characters look substantially different from English characters, they can be typeset using a capitalized letter name, as in Δ or Ω.

```
The Euler phi function is, rather obviously denoted $\phi(n)$,
although one will often see the variation $\varphi(n)$ appear in
mathematics texts. When the upper case Greek characters look
substantially different from English characters, they can be typeset
using a capitalized letter name, as in $\Delta$ or $\Omega$.
```

Ellipses

Very often we would like to use ellipses in math to suggest the presence of many more terms than we are willing to write out. Different ellipses commands align the dots with the center or bottom of the text Let x = (x1, x2, . . . , xn) be an n-dimensional vector. Then the number |x1| + |x2| + · · · + |xn| is called the manhattan, or taxi cab length.

```
Let $x=(x_1,x_2,\ldots,x_n)$ be an $n$-dimensional vector. Then the
number $|x_1|+|x_2|+\cdots+|x_n|$ is called the manhattan, or taxi
cab length.
```

Functions

Function Names

Because all of the text in math mode is assumed to represent variables, we need to do something different for function names.

Some common functions are built in:

lim x→0 sin(2πx)

In the general case, you can format math text to make it nonitalicized:

y = modify(x)

However, if you want to put text between symbols, you should use mbox or textrm: x = n or 0. Note that you need to pad the text with space, or it will come out funky: x = nor0.

```
Some common functions are built in: $$\lim_{x\to0}\sin(2\pi x)$$ In
the general case, you can format math text to make it non-italicized:
$$y=\mathrm{modify}(x)$$ However, if you want to put text between
symbols, you should use \texttt{mbox} or \texttt{textrm}: $x=n\mbox{
or }0$. Note that you need to pad the text with space, or it will
come out funky: $x=n\mbox{or}0$.
```

Fractions, Square Roots and Binomials

Special commands with arguments are provided for fractions and square roots:

$$\frac{d\sqrt{x}}{dx}=\frac{1}{2\sqrt{x}}$$

```
$$\frac{d\sqrt{x}}{dx}=\frac{1}{2\sqrt{x}}$$
```

Using the amsmath package, you can also typeset binomials:

$$\binom{n}{k}=\frac{n!}{k!(n-k)!}$$

```
$$\binom{n}{k}=\frac{n!}{k!(n-k)!}$$
```

Parentheses, Brackets, etc

In typing mathematics, we use many different enclosing symbols:

$$[n]=\{1,2,\ldots,n\}$$

```
$$[n]=\{1,2,\ldots,n\}$$
```

$$|\langle x_1, x_2\rangle+\langle 1,1\rangle|=\sqrt{(x_1+1)^2+(x_2+1)^2}$$

```
$$|\langle x_1, x_2\rangle+\langle1,1\rangle|
=\sqrt{(x_1+1)^2+(x_2+1)^2}$$
```

Sometimes they are nested deeply, and we would like to size them appropriately:

$$3.245=3+\left(\frac{1}{4+\left(\frac{1}{12+\left(\frac{1}{4}\right)}\right)}\right)$$

```
$$3.245=3+(\frac{1}{4+(\frac{1}{12+(\frac{1}{4})})})$$
```

Environments

Equations

LATEX provides for automatic numbering of equations for later reference. This is done using the equation environment.

π = 3.14159265358979323846 . . . (1)

If an asterisk is added, then the numbering is suppressed.

π = 3.14159265358979323846 . . .

```
\LaTeX\ provides for automatic numbering of equations for later
reference. This is done using the
```

```
\texttt{equation} environment.
\begin{equation}
\pi=3.14159265358979323846\ldots
\end{equation}
If an asterisk is added, then the numbering is suppressed.
\begin{equation*}
\pi=3.14159265358979323846\ldots
\end{equation*}
```

Equation Arrays

Equation arrays are perfect for long derivations, or lists of equations. Again, the asterisk suppresses numbering. Columns and Line Breaks are done like tables and arrays.

$$S(x) = \sum_{k=0}^{\infty} x^k$$

```
\begin{eqnarray*}
S(x) & = & \sum_{k=0}^\infty x^k \\
&=& 1+x+x^2+x^3+\cdots\\
xS(x) &=& x+x^2+x^3+\cdots\\
S(x) - xS(x)& = & 1 \\
S(x) &=& \frac{1}{1-x}
\end{eqnarray*}
```

Arrays

Similar to tables in normal text, we have arrays in math mode. Arrays are very helpful for typesetting matrices and piecewise functions:

$$\left|\begin{array}{ccc} i & j & k \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w_3 \end{array}\right| = v \times w$$

```
\[
\left|\begin{array}{ccc}
i & j & k \\
v_1 & v_2 & v_3 \\
w_1 & w_2 & w_3
\end{array}\right|
= v\times w
\]
```

$$\operatorname{sign}(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases}$$

```
\[
\textrm{sign}(x)=\left\{\begin{array}{ll}
1, & x>0 \\
0, & x=0 \\
-1, & x<0
\end{array}\right.
\]
```

Structured Text

In the preceding two sections, you learned LATEX basics and how to typeset math. However, the perspective was on the micro-structure, roughly on the level of one or two paragraphs. This section of the tutorial is primarily about the macro-structure of your LATEX document.

Document Structure

Sections, Subsections, Subsubsections

One of the big advantages of LATEX, is that you can explicitly encode the structure of your document, using the commands

\section{} \subsection{} \subsubsection{}

When one of these commands is invoked a numbered heading is generated with the name given by the command's argument. For instance, if you wanted your document to have a section on spatulas, you just need to place the command \section{Spatulas} right before the text of your scintillating study on spatulas.

LATEX will automatically number the sections, subsections, etc. of your document, unless you suppress the numbering by adding an asterisk to the command: \section*{Numberless} However, we can put all these numbers to good use for cross references.

Cross-references

In LATEX we don't type

see section 2

Instead we type

```
see section \ref{spat sec}
```

and place the command \label{spat sec} right after the \section{Spatulas} command. For instance,

```
 \section{Herman the Hero}
\label{herman}
Herman was just an ordinary man, until one day...
```

As you can see, the contents of the label command are rather arbitrary. We're just declaring some name, some label for later reference.

By using labels rather than static references, we're de-coupling our cross-references from the document layout. Suppose we're about to "go to press" (eg. holy crap, the assignment's due in 10 minutes) and we decide that section 3 should really come before section 2. If we typed in all those cross-references by hand, then they would all be "broken" and we would have to go correct them all by hand. With label and ref, we just cut and paste section 3 in front of section 2 and the entire document gets renumbered automatically when we compile. Cool!

Warning! You will likely have to compile 2 or 3 times before all of the page references resolve. I won't explain why, because that's pretty boring, but you should know that this is the case. We can use this same mechanism to make page references too. Instead of \label, we use the command \pageref

```
 ...which is why they eat cantelope under the full moon every 2
weeks. This ceremony is known as the
\textbf{recantelope}\pageref{recantelope def}. This is a very ...
...in many ways very similar to the recantelope (see page
\ref{recantelope def}).
```

Footnotes

If you want to place footnotes in your document[1] , then all you need to do is place the footnote command inline with your text:

[1]and who wouldn't?

```
\footnote{and who wouldn't?}
```

The footnote command will get replaced by a number referencing the footnote, and the text will be automatically placed at the bottom of the page. LATEX will handle all of the numbering for you

Table of Contents, Title, and Author

Once the document is suitably structured, a table of contents may be automatically generated by placing the command

```
\tableofcontents
```

at the point in the document where you want the table of contents to be.

We can also generate a nicely typeset title and author name, using the commands title, author, and maketitle. Other options like the date, and e-mail address can also be specified, though they won't be here.

```
\title{World Domination}
 \author{Dr. Death}
\maketitle
```

Preamble and Global Options

Preamble Commands

In our Hello World example, the preamble is the part of our document that resides between the initial \documentclass command and the \begin{document} command.

The Preamble →	HelloWorld.tex \documentclass{article} \begin{document} Hello, World! \end{document}

The preamble is where we put global commands to alter the entirety of the generated document, rather than the local text. Perhaps the most important preamble command to know is \usepackage{}. This command allows you to enable extra features for your LATEX

document. A number of packages have come up during this tutorial. Often the safest thing to do is err on the side of using superfluous packages.

```
\usepackage{verbatim}
\usepackage{amsfonts}
\usepackage{amsmath}
```

Custom Commands

We can also use the preamble to define our own commands. Occasionally this will save you time typing in long obnoxious commands. One very common example in math, is to use the "blackboard" letters $\mathbb{R}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{C}$ to denote sets of numbers (the reals, integers, rationals, and complex numbers here—if you're curious). Normally this would require typing in

```
\mathbb{R}
```

every time we wanted the blackboard R. However, we can define a new command with the name \R so that we need only type \R to get a blackboard R. This is done using the \def "meta-command"

```
\def\R{\mathbb{R}}
```

Def says to create a new command with name \R, which, when invoked, will be replaced by the text in between the braces.

However, sometimes we would also like to define commands that take arguments. For instance, suppose that I am writing an introductory Biology text and want to typeset all of my terminology definitions in bold, so that they may be more easily picked out from a large block of text. I could wrap all of these definitions in explicit \textbf{} commands, but this is not a very extensible or flexible approach. For instance, I may later decide that boldface is too brash and distracting from the rest of the text and wish to change all of my definitions to italicized text. I do not want to hunt down every definition to change boldface commands to italicization 27 commands. Instead I can define a terminology command, centralizing, or factoring out this decision from my text.

```
\newcommand{\term}[1]{\textbf{#1}}
```

Let's break this down. The newcommand command takes three arguments. The first is the name of the command to be created, \term in this case, wrapped in curly braces. Then, a number is placed in square braces. This number tells newcommand how many arguments the new command is supposed to take. Finally, the text that replaces the command invocation is specified, just as before. The difference is that we can now use the special sequences #1, #2, etc. to specify where the arguments should be added in.

> Chromosomes are organized structures of DNA and proteins that are found in cells.

```
\term{Chromosomes} are organized structures of DNA and proteins that
are found in cells.
```

Now you need only change the command definition to change how all of the terminology is typeset. However, you can even do one better. One would often like to refer back to definitions with a page number reference. We can roll this into our command:

```
\newcommand{\term}[1]{\textbf{#1}\pageref{#1}}
```

Now, you can invoke \ref{Chromosomes} to produce the page number of the chromosome definition. You could even go a step further and instead define \term to take two arguments: one the word to be type-set and one for the label name.

```
\newcommand{\term}[2]{\textbf{#1}\pageref{#2}}
```

Document Classes

One can change many things about the appearance of a LATEX document merely by changing the documentclass command. For instance, our current Hello World example uses the article document class. Besides article, we can also use book for longer format documents and letter for shorter 28 format documents. Changing this option will then change what structuring commands are available. In book we gain the \chapter command in addition to section and subsection. In letter, we lose sections and subsections altogether, since there is no need to structure such a small document.

The documentclass command also allows us to specify extra parameters as options, using the square brackets. For example,

```
\documentclass[12pt, a4paper, titlepage]{article}
```

Besides the rather obvious control on font size, you can add one of the different paper sizes to produce an appropriately sized document: a4paper, letterpaper, a5paper, b5paper, executivepaper, legalpaper. You can also add one of the commands, titlepage or notitlepage, to alter or supress the default behavior. You may also use one of the two onecolumn, twocolumn. This is only a partial list of possible options