INTRODUCTION TO \LaTeX

JEREMY WEST

1. Introduction

\LaTeX is a markup language for a typesetting system. A markup language is a collection of commands, often called macros, which describe the formatting and organization of a document. Other examples of well-known markup languages are HTML (HyperText Markup Language) and XML (eXtensible Markup Language).

This is an introduction to the \LaTeX markup language. I will discuss the tools you will need but I will not cover setup, installation, or use of those tools. I will also not attempt to provide even a cursory list of symbols. There are many excellent lists, for example, www.ctan.org/tex-archive/info/symbols/comprehensive/symbols-a4.pdf.

My intent is rather to introduce the concepts behind \LaTeX. Once the fundamentals are mastered, the numerous online resources documenting various commands and environments, as well as the almost limitless number of example documents should suffice to solve specific problems. I will share a few tips and tricks and useful environments that come up frequently and that were not easily found online.

1.1. History. Between 1979 and 1989, Donald Knuth, professor of Computer Science at Stanford University, developed the \TeX typesetting system to typeset his several volume series: The Art of Computer Programming. One of the greatest features of \TeX was that it was intended to produce high-quality digital typography at any font size with consistent results on any computer. However, \TeX was pretty difficult to use.

During the early 1980’s, Leslie Lamport developed \LaTeX. \LaTeX is a macro language that sits on top of \TeX in other words, \LaTeX gives authors an easier to use markup language while still providing much of the power of \TeX. And of course, \LaTeX preserves the beautiful quality documents that are produced by \TeX. The current version of \LaTeX is \LaTeX 2ε.

Because, as Donald Knuth put it, \TeX is “intended for the creation of beautiful books and especially for books that contain a lot of mathematics,” the American Mathematical Society has created a collection of document styles, packages, and symbols for \LaTeX to support the creation of mathematical documents. These are collectively referred to as \AMS-L\TeX.

1.2. \LaTeX Documents. A \LaTeX document is simply a text file, usually saved with the extension .tex, that contains content and markup. The document consists of two sections, the preamble and the body. The preamble references packages and external files, declares commands, specifies styles and numbering, etc. while the body contains the content of the paper, with special commands to specify organization and formatting.
2. Components of a \LaTeX System

Even though a \LaTeX document is a text file, there are a couple pieces of software you will need to create and work with \LaTeX.

2.1. A \LaTeX Editor. Because \LaTeX documents are simply text files, they can be created using any text editor available, for instance, Notepad. However, most users prefer to use an editor designed for \LaTeX. For Windows, I recommend WinEdt. For Mac, TeXShop is probably the most common. While TeXShop is easy to use, it doesn’t have as many features of WinEdt. Frankly, I haven’t found an editor for Mac that compares to WinEdt.

2.2. A \LaTeX Distribution. Once you have created your \LaTeX file, you will need to turn it into something viewable. For this, you need a \LaTeX distribution. The distribution contains the symbol and document type definitions and the programs that convert your document into something viewable. For Windows, MikTex is the most common distribution. For Mac, TeXLive is commonly used. You can find installation packages for both of these online.

There is a central place for all things TeX, although you will find any number of TeX resources online due to its open source nature. The Comprehensive TeX Archive Network (www.ctan.org) has examples and tutorials as well as a repository of all the standard TeX packages.

3. The \LaTeX Hello World

Before going much further, let’s create your first \LaTeX document. Type the following text into your \LaTeX editor:

\begin{document}
Hello World
\end{document}

If you already know how to typeset (or compile) this, do so. You have just created your first \LaTeX document.

Before moving on, let me comment on the terms typeset and compile. These terms are often used interchangeably to mean the process of converting a text file containing \LaTeX markup, called a \LaTeX source file, into a viewable \LaTeX document. However, typesetting can also refer to the entire process of generating the markup and creating the final document. So to avoid confusion, I use compilation to refer to the creation of a \LaTeX document from \LaTeX source.

At a later time we will look more into the process of compiling a \LaTeX document to understand what is actually happening.

4. \LaTeX Basics

Let’s cover the basic anatomy of a \LaTeX document. Once we get this out of the way we can do the interesting stuff.
4.1. **Commands.** Commands in \LaTeX\ are prefixed using a backslash character (\). For example, the command to print the greek character \( \alpha \) is \textbackslash alpha. Some commands print symbols while others change the formatting. For those commands, there are two varieties. Some take an argument. For example, to print a blackboard character \( R \), to represent the reals, I use the command \textbackslash mathbb{R}. This produces \( \mathbb{R} \). The content inside the braces (\{, \}) is called the argument. Some commands have more than one argument. For instance, the command to create a fraction is \textbackslash frac{a}{b} which produces \( \frac{a}{b} \). Notice that the content of the first argument goes on top, and the second argument goes on the bottom. Some commands have optional arguments that are placed in square brackets. For instance, the \textbackslash title command, which is used to create a title for your document takes as its only argument the title of your paper. However, if you want to use a shorter version of the title for the header, you can type

\texttt{\textbackslash title[Short Title]{Really long title that will not fit well}}

Some commands alter the output from that point onward. For instance, the \textbackslash em command sets the text mode to emphasized and produces italicized text such as \textit{this really important message}. If you place \textbackslash em in your document, everything from that point onward will be emphasized. This is typically not what you want. So you use the braces for scoping which controls the area you want to affect. In other words, you would type \texttt{\{\textbackslash em This is emphasized\}} \texttt{this is not} to produce the text

\textit{This is emphasized} \texttt{this is not}

This is somewhat confusing since \textbackslash em{Emphasized text} will emphasize everything after that point, not just the text in braces. Some of this inconsistency is due to the number of different versions and flavors of \TeX. If you really don’t like this, there is usually an alternate command, such as \textbackslash textit that works like you would think: \textbackslash textit{Emphasized} produces \textit{Emphasized}.

The idea of scoping is actually an important one. In many parts of \LaTeX\ braces are used to identify a single unit. For example, the fraction \( \frac{1}{2} \) can be produced using the command \textbackslash frac{1}{2} but you may simply write \textbackslash frac{1}{2} instead. Notice that \LaTeX\ treats each symbol as an individual argument. So how do you make the fraction \( \frac{12}{37} \)? Simple, you use braces to indicate what \LaTeX\ should treat as a single unit. So the command is \textbackslash frac{12}{37}.

The last type of command is what is called an environment. An environment describes how a section should be formatted. For example, the \texttt{center} environment indicates that everything in that section should be centered. Environments have a consistent syntax:

\begin{center}
This text will be centered.
\end{center}

The \texttt{begin} and \texttt{end} commands indicate the beginning and ending of the environment and the text in braces indicates the type of environment.

There are thousands of commands in \LaTeX\ and you can even create your own. Later in this document we will cover formatting and many of the commands you will commonly use.

4.2. **The Preamble.** The preamble is all of the commands that come before the body of a \LaTeX\ document. Every \LaTeX\ document begins with the command
\documentclass{class_name}. Here <class_name> is replaced with the class of the document. There are many document classes available. Here are some that you will probably use:

<table>
<thead>
<tr>
<th>Document Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>book</td>
<td>Used for creating books.</td>
</tr>
<tr>
<td>amsbook</td>
<td>The \texttt{AMS-\LaTeX} version of book.</td>
</tr>
<tr>
<td>article</td>
<td>Used for creating articles.</td>
</tr>
<tr>
<td>amsart</td>
<td>The \texttt{AMS-\LaTeX} version of article.</td>
</tr>
<tr>
<td>thesis</td>
<td>Used for creating a PhD or Masters Thesis.</td>
</tr>
<tr>
<td>beamer</td>
<td>Used for creating beamer slides.</td>
</tr>
</tbody>
</table>

The next command is frequently the \texttt{\usepackage} command. A package in \LaTeX is a file that defines formatting, commands, or other options for your document that you can optionally include. Here are some packages you will frequently use:

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>amsmath</td>
<td>Contains the \texttt{AMS-\LaTeX} math environments.</td>
</tr>
<tr>
<td>amsfonts</td>
<td>The \texttt{AMS-\LaTeX} math fonts</td>
</tr>
<tr>
<td>amssymb</td>
<td>The \texttt{AMS-\LaTeX} math symbols</td>
</tr>
<tr>
<td>amsthm</td>
<td>Support for \texttt{AMS-\LaTeX} theorem environments and styles</td>
</tr>
<tr>
<td>epsfig</td>
<td>Support for \texttt{.eps} figures.</td>
</tr>
<tr>
<td>graphicx</td>
<td>Advanced graphics support.</td>
</tr>
<tr>
<td>makeidx</td>
<td>Support for creating an index.</td>
</tr>
</tbody>
</table>

If you are submitting a paper to a journal or conference, they will often have their own document classes and/or packages to control the formatting of your paper.

While there are many other commands usually contained in the preamble, this will get us started until those commands make sense.

4.3. The Document Body. The document body begins with \texttt{\begin{document}} and ends with \texttt{\end{document}}, which is almost always the last command in a \LaTeX document.

5. Text Mode, Math Mode, and Display Mode

There are three basic modes in \LaTeX: text mode, math mode, and display mode. Text mode is for standard text, like this text. Math mode is for creating inline mathematical expressions like $N(A - \lambda I) \neq \{0\}$. Display mode is for creating centered expressions with variable-sized operators such as

$$EX = \int_{\Omega} X dP.$$

Some commands will only work in one mode or another. For instance \texttt{\alpha} only works in math mode or display mode whereas \texttt{\em} only works in text mode. Most anything that works in math mode works in display mode and vice-versa, although the output may be different. For instance, the code \texttt{\lim_{n \to \infty} a_n}
INTRODUCTION TO LATEX

produces \( \lim_{n \to \infty} a_n \) in math mode whereas in display mode it produces

\[ \lim_{n \to \infty} a_n. \]

Notice that the limits are placed below as is common in display mode, but in math mode they are placed inline so that the expression fits better in the text. You can often override this in math mode by using the \texttt{\textbackslash displaystyle} command. For example, typing \texttt{\textbackslash displaystyle\{\lim_{n \to \infty} a_n\}} while in math mode produces \( \lim_{n \to \infty} a_n \). Notice that the limits are below now and that the next line of text is bumped down a little to make room.

Because the only difference between math mode and display mode is whether it is inline or centered on its own line and the positioning of limits (and other examples we will see) I will refer to both of them as math mode and differentiate between the two only when necessary.

6. SUBSCRIPTS AND SUPERSCRIPTS

Creating subscripts and superscripts can be done in math mode only and is done using the underscore (\texttt{_}) and carrot (\texttt{^}) characters. As with arguments, unless you use braces only the next character is treated as the sub-(super-)script. For example, the command \texttt{x_{2k}} produces the output \( x_{2k} \). If you want \( 2k \) to be the subscript, use \texttt{x_{2k}}. The ordering of super- and subscripts does not matter. Therefore, \( x_{k^2} \) and \( x^{2_k} \) both produce the output \( x_{k^2} \). You can create multiple levels of subscripts, but you must use braces. For example \texttt{x_{n_k}} produces the output \( x_{n_k} \).

Subscripts and superscripts are used for many symbols. For example, consider the following symbols which often have limits associated with them:

<table>
<thead>
<tr>
<th>\TeX Code</th>
<th>Produces</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\int_a^b$</td>
<td>(\int_a^b)</td>
</tr>
<tr>
<td>$\sum_{n=0}^{\infty}$</td>
<td>(\sum_{n=0}^{\infty})</td>
</tr>
<tr>
<td>$\prod_{k=1}^n$</td>
<td>(\prod_{k=1}^n)</td>
</tr>
<tr>
<td>$\lim_{x \to x_0}$</td>
<td>(\lim_{x \to x_0})</td>
</tr>
<tr>
<td>$\bigcup_{i=1}^n$</td>
<td>(\bigcup_{i=1}^n)</td>
</tr>
<tr>
<td>$\bigcap_{i=1}^n$</td>
<td>(\bigcap_{i=1}^n)</td>
</tr>
</tbody>
</table>

In producing the table I used math mode (not display mode). If display mode is used, the limits are placed above and blow instead of to the right like normal subscripts and superscripts.

7. MATH ENVIRONMENTS

There are several environments that are useful in display mode for controlling the display and alignment of equations.

There are many math environments to number and organization equations in display mode. Rather than trying to cover them all here, I will instead refer you to the math section of the wikibook on LaTeX, which is an excellent resource.

http://en.wikibooks.org/wiki/LaTeX/Mathematics
8. Sections, Theorems, and Numbering

8.1. Sections and Dividing Your Document. Sections are created in a document using the \section command. For example, to create a section called introduction, use

\section{Introduction}

If I want to create a section which is not numbered, say for instance a preface, I would use

\section*{Preface}

Everything after the section command and up until the next section command is considered part of the section. Subsections and subsubsections are created similarly. If you want to add an appendix, use the \appendix command. Once you have used the \appendix command, you can create sections and subsections, only know they will be labeled using letters instead of numbers and will be considered part of the appendix.

In books and theses (the document classes for those) you can also create parts using \part and chapters using \chapter. These can also be unnumbered.

If you want to print a table of contents at some point in your document, just use the command \tableofcontents which prints a completed table of contents at that point. The formatting of parts, chapters, sections, subsections, subsubsections, and the table of contents is determined by the document class and any packages you have included.

8.2. Creating and Numbering Theorem Environments. There are several numbered items in a math document. For example, exercises, theorems, lemmata, corollaries, etc. are typically numbered. Sometimes examples may or may not be numbered, but usually you want them to be set off from the text somehow.

If you have the \amsthm package included, you may use the \theoremstyle and \newtheorem commands to create, style, and number these environments. The \theoremstyle command has a single argument which determines what style will be used on all theorem environments created using \newtheorem from that call until the next \theoremstyle. Common styles are plain, definition, and remark. The \theoremstyle command is a little more involved. Consider the following example:

\theoremstyle{plain}
\newtheorem{theorem}{Theorem}[section]
\newtheorem{corollary}{corollary}[theorem]{Corollary}
\newtheorem{lemma}{lemma}[theorem]{Lemma}
\newtheorem{proposition}{proposition}[theorem]{Proposition}

\theoremstyle{definition}
\newtheorem*{example}{Example}
\newtheorem*{definition}{Definition}

\theoremstyle{remark}
\newtheorem*{remark}{Remark}

The first command indicates that the environments theorem, corollary, lemma, and proposition should be formatted using the plain style. The command
\newtheorem{theorem}{Theorem}[section] creates a new environment, called \texttt{theorem}. The title of this environment is “Theorem”. The last part indicates that theorems should be numbered within a section. To state a theorem in my paper, I would write the following:
\begin{theorem}
There are infinitely many prime numbers.
\end{theorem}
This produces the result

\textbf{Theorem 8.1.} \textit{There are infinitely many prime numbers.}

Notice the number of the theorem: 8.1. The 8 is because the theorem is contained in section 8. The 1 is because it is the first theorem in the section. In fact, this is exactly what the \texttt{[section]} part does. It indicates that theorems should be numbered within sections. The \texttt{newtheorem} command has created a \texttt{counter} named theorem. Every time I create a theorem, the value of the counter is used to generate a number and then the counter is incremented. Including the option \texttt{[section]} on the end tells \LaTeX{} that the \texttt{theorem} counter should be reset to 1 every time the section counter is incremented. Of course, from what you know about sections, the section counter is incremented each time I use the \texttt{section} command.

Notice that command defining corollaries is similar to theorem except that I have placed \texttt{[corollary]} instead of \texttt{[section]} and I put it before the title “\texttt{Corollary}”. When the option is placed before the title it has a different meaning. Instead of indicating that corollaries should be numbered within theorems, this means that corollaries should use the same counter. So instead of creating a new counter called \texttt{corollary}, each time I use the \texttt{corollary} command the value of the \texttt{theorem} counter is used and incremented. So, adding the following \LaTeX{} code:
\begin{corollary}
Given any $n \in \mathbb{N}$ there exists a prime number $p \in \mathbb{N}$ such that $p > n$.
\end{corollary}
produces the following result:

\textbf{Corollary 8.2.} \textit{Given any $n \in \mathbb{N}$ there exists a prime number $p \in \mathbb{N}$ such that $p > n$.}

For all intents and purposes, a corollary is the same as a theorem except the titles are different.

By this point you will have noticed that lemmata and propositions are similarly numbered. Next we change the style to \texttt{definition}. We then create an \texttt{example} environment. Notice that we have not specified the name of any counters and that we have included an asterisk (*). The asterisk indicates that we don’t want \texttt{example} to be numbered. We just want a title. For this reason, we don’t need (and cannot in fact) include an option indicating the counter to use or number within. Thus, the code
\begin{example}
The prime number 101 is bigger than the number 90.
\end{example}
produces the result
**Example.** The prime number 101 is bigger than the number 90.

Notice the difference in the style. Theorems are printed in italics and definitions (and examples using our styling) are in roman font. Even though I am not numbering my examples, I may wish to name them. I can do this for most any environment in fact. The code

\begin{example} [Russell’s Paradox]
Let $S$ be the set of all sets which do not contain themselves. Then $S$ cannot exist and cannot in fact be a set.
\end{example}

produces the result

**Example (Russell’s Paradox).** Let $S$ be the set of all sets which do not contain themselves. Then $S$ cannot exist and cannot in fact be a set.

Next we defined the **definition** environment. We have not referenced a counter anywhere, either to number with or number within, but we have not included an asterisk. So then definitions will be numbered using the **definition** counter that will not be reset. Hence, the code

\begin{definition}
A set is *closed* if its complement is open.
\end{definition}

produces the result

**Definition 1.** A set is *closed* if its complement is open.

Notice the number is not dotted, and it did not pick up where theorem and corollary left off but instead started at one.

The last **newtheorem** command allows us to create a **Remark.** Useful remark for the paper

I assume at this point you can guess what the code must look like. Notice that we indicated we did not want remarks numbered. This does not have to be the case however.

Counters are central to numbering and organization in **\LaTeX.** The site


is a great reference for using counters.

9. Creating Commands

Creating command in **\LaTeX** is useful for several reasons. One, it can save typing on things you frequently type. But the biggest reason is to separate formatting from content. For example, suppose you have `\mathbb{R}` all over the place in your document and then you find out that the journal, publisher, or conference you are submitting to prefers `\mathbf{R}`. You would have to go through and change each place you have `\mathbb{R}`. However, if you created a command `\R` then just changing the definition of the command could fix the change.

The easiest way to create a command is using the **newcommand** command. For example, if you place the markup

\newcommand{\R}{\mathbb{R}}
INTRODUCTION TO LATEX

in your preamble, \TeX creates a command \textbackslash R. Now, wherever \TeX finds the command \textbackslash R it replaces it with \texttt{\textbackslash mathbb\{R\}}. Obviously then, you can only use the command \textbackslash R in math mode or display mode.

9.1. **Redefining Commands.** Sometimes, the command you want has already been defined. In which case, using \texttt{\textbackslash newcommand} will give you an error when you compile your document. You have two options in this case: pick a different name for your command, or use the \texttt{\textbackslash renewcommand} command. \texttt{\textbackslash renewcommand} works the same as \texttt{\textbackslash newcommand} but it changes the definition of the function to what you have supplied. You should probably make sure you know what command you are replacing before you do this. But there is an example of a time when you want to use this.

The \texttt{enumerate} environment allows you to create numbered lists. For example,

\begin{verbatim}
\begin{enumerate}
  \item This is the first item.
  \item This is the second item.
  \item This is the third item.
\end{enumerate}
\end{verbatim}

produces the result

(1) This is the first item.
(2) This is the second item.
(3) This is the third item.

That may not be how we want to number our items. The \texttt{enumerate} environment uses the command \texttt{\textbackslash labelenumi} to create the labels for the first level of an enumeration (you can nest \texttt{enumerate} environments). So, if we add the markup (anywhere in the document actually)

\texttt{\textbackslash renewcommand\{\textbackslash labelenumi\}\{\textbackslash alph\{\textbackslash enumi\}\}.}

then we would get the result

a. This is the first item.
b. This is the second item.
c. This is the third item.

Notice that \texttt{enumerate} uses the \TeX command \texttt{\textbackslash alph\{\textbackslash enumi\}} to create the labels. \texttt{enumi} is the counter for the items and \texttt{\textbackslash alph\{\textbackslash enumi\}} outputs the value of the counter as a lower-case alpha-character. You can also use \texttt{\textbackslash roman} and \texttt{\textbackslash alph}.

9.2. **Math Operators.** If you put the markup $\texttt{\$log(x)}$ in a document you get $log(x)$. Notice that the $log$ function looks like a bunch of variables. We really want it to look like $\log(x)$. We can do this using the markup \texttt{\textbackslash mbox\{log\}(x)}), but this would get tedious to type over and over. So \TeX has a built-in command \texttt{\log} which gives you the result you want. Similarly, there are commands for sin, cos, tan, exp. But some math operators are not defined by default. Let’s say you want a trace operator, so you want to display $\texttt{tr(A)}$. You can create a command using the markup

\texttt{\textbackslash newcommand\{\texttt{\textbackslash tr}\}\{\texttt{\mbox\{tr\}}\}}

but there is a simpler and better way (better because it documents what you are doing so you don’t get confused later). The markup

\texttt{\textbackslash DeclareMathOperator\{\texttt{\textbackslash tr}\}\{\texttt{tr}\}}
will create the command for you. Now you can use $\texttt{\textbackslash tr}$ in math or display mode. The \texttt{DisplayMathOperator} command also has one other nice feature. Suppose you want to create an operator that works like the union or sum operators, so you want limits on the top and the bottom in display mode. Let’s say we want to create a command that looks like

$$\Theta_{i=1}^{n} A_i.$$ 

Then we could just put \texttt{\DeclareMathOperator*{\theop}{\Theta}} in the preamble and then use the \LaTeX code

\[
\texttt{\theop_{i=1}^{n} A_i.}
\]

The asterisk tells \LaTeX to put limits above and below in display mode. Note that the \texttt{\DeclareMathOperator} command can only be placed in the preamble.

10. Bibliographies

\LaTeX has great support for bibliographies using BiBTeX. Getting a bibliography into your document requires three steps: creating the bibliography database, citing the bibliography, and printing the results.

10.1. The Bibliography Database. A bibliography database is a text file, often called a BiBTeX file, that contains the sources you wish to cite. The nice thing about a bibliography database is that you list all the information you have about a source, without formatting it, for any of the sources you might use in any order. You can also put comments in your database to indicate the contents of the items. Once you have created your bibliography database, BiBTeX takes care of formatting and ordering the bibliography that is included in your document.

A BiBTeX file is just a list of bibliography entries. For example, a BiBTeX file might contain the following:

@book {durrett,
    AUTHOR = {Durrett, Richard},
    TITLE = {Probability: theory and examples},
    EDITION = {Second},
    PUBLISHER = {Duxbury Press},
    ADDRESS = {Belmont, CA},
    YEAR = {1996},
    PAGES = {xiii+503},
    ISBN = {0-534-24318-5},
    MRCLASS = {60-01},
    MRNUMBER = {MR1609153 (98m:60001)},
}

@book {campbell-meyer,
    AUTHOR = {Campbell, S. L. and Meyer, Jr., C. D.},
    TITLE = {Generalized inverses of linear transformations},
    NOTE = {Corrected reprint of the 1979 original},
    PUBLISHER = {Dover Publications Inc.},
    ADDRESS = {New York},
    YEAR = {1991},
}
This database consists of two books. The first part is the tag that will be used to cite the source. The remainder of each entry specifies the information about the source.

For the mathematical sciences, MathSciNet (located at ams.org/mathscinet) is a great resource for building your BibTeX file. Simply search for the source you want to include. When you find the source and have navigated to it, click on the combo box for other formats, choose BibTeX and copy and paste the supplied BibTeX entry into your BibTeX file.

10.2. Citing Sources. To cite a source in your document, use the \cite command. For example, you might use something like

This can be found in \cite{durrett}.

Assuming you have actually included your bibliography in the document, a properly formatted citation will appear in your document in place of \cite{durrett} and the Durrett source will be added to your bibliography.

10.3. Printing the Bibliography. Once your database has been created and your sources have been cited in your document using the \cite command, you need to instruct \TeX where to print your bibliography. To do this, use the \bibliography command. For example, if my bibliography database was called math.bib, I would insert the command

\bibliography{math}

wherever I wanted the bibliography to be printed. Notice that I didn’t include the extension (.bib), just the file name. The bibliography database needs to be in the same directory as your document. Note further that \TeX will include only the sources that you cited. So in my document, the durrett source would appear in the bibliography, but not the campbell-meyer source since I never cited it.

Many journals and conferences will have a specific format for their bibliographies. They will supply you with style files that will properly format your bibliography assuming you have correctly created your BibTeX file and cited your sources. Alternatively, you may use the command \bibliographystyle with any of the following bibliography styles: plain, unsrt, abbrv, alpha. The \bibliographystyle command should be placed before the \bibliography command. For example, the code

\bibliographystyle{unsrt}
\bibliography{math}

would created a bibliography from the math.bib BibTeX file and would format it using the unsrt package, which orders the bibliography in the order sources are cited.

11. Indices

Creating an index for your document is very simple using the makeidx package. You should note that some document classes already have support for the index
commands. A simple approach is to try to use the index commands without including the makeidx package. If you get an error saying the commands are not defined, then add the package. The index is created in two steps: tagging the entries, printing the index.

11.1. Tagging Index Entries. To tag an item in your document as something that should appear in the index, use the \index command. For example, I might have something like this in my document:

\begin{definition}
A \textit{prime number} \index{prime} is an integer greater than 1 whose only divisors are 1 and itself.
\end{definition}

An entry will be created for the index under the heading \textit{prime} which will reference the page on which this definition appears. To create subentries or subsubentries in your index, use the ! character. For example, \index{cheese!gouda} would create a \textit{cheese} entry in the index which would have a subentry called \textit{gouda} which would reference this page. A subsubentry could be done using ! twice: \index{cheese!gouda!brie}.

You can create cross-references in your index using the syntax \index{cheese!see{crackers}}. Also, you can style your entries. For example, an entry for \textit{\(\sigma\)-algebra}, should be alphabetized under sigma-algebra, but should display correctly. For this, use the syntax \index{sigma-algebra@$\sigma$-algebra}. If you have a large section, say for example, a proof or a large explanation, that you want to index as one item, you can use the syntax

\index{cheese|(} ...
...content about cheese...
\index{cheese|)}

If the content spans multiple pages, it will be correctly listed, for example cheese, 12-14. If it spans only one page it will show up as cheese, 12.

11.2. Printing the Index. To print the index you need to add two commands. The \texttt{makeindex} command is placed in the preface and simply instructs the \LaTeX engine to track index information. The \texttt{printindex} command should be placed in your document wherever you want the index to appear.

12. Compilation Revisited

Getting numbered references, citations, bibliographies, and indices to appear correctly can sometimes be frustrating and bewildering. This is because the document compilation process for a \LaTeX document happens in several phases. Understanding this process will make it easier to get the correct output for your document.

When you typeset your document, the \LaTeX engine produces your output file (usually .pdf or .div) and an auxiliary file with the extension .aux. The auxiliary file contains information about the symbols in your document. For example, the numbers that correspond to labeled items, the bibliography sources you cite, and the index entries you create. This file is needed to make your citations, bibliography, references, and index appear correctly. If this file does not exist, \LaTeX still creates the output file, but it will likely have question marks in place of references.
and citations and it will not include a bibliography or index. Once you have generated the auxiliary file, you need to run BiBTeX on your document to prepare a bibliography file with extension .bbl and then run MakeIndex on your document to create the .idx file. Once these files exist, you can re-typeset your document to get an output file with all the references and extra content correctly in place.

Some editors will do this for you automatically. Some will not. There are many ways to automate this, for example, with a Makefile. Some editors provide menu options for running these commands, but they require you to do it yourself. Whatever the case, just know that if you create new labels, rearrange number items, add citations, add sources, or add index entries, then you will probably need to recreate the auxiliary file, and possibly the bibliography or index file, as needed. If you just want to read the text you have written, typesetting once will give you something pretty close, you might just have some question marks or missing index or bibliography entries. But before submitting, printing, or distributing your document, you will probably want to typeset it to make sure you have an up-to-date auxiliary file, then run BiBTeX, then run MakeIndex, then typeset again to make sure everything is current.