LaTeX: A Guide for Philosophers

Charlie Tanksley
tanksley@virginia.edu

July 14, 2010

Contents

1 Introduction

LaTeX is a fantastic tool for philosophers of all stripes—though it excels at typesetting the complicated formulas of logicians, it has distinct advantages for all philosophers. (I discuss some of these here.) If you are here, I assume you want to learn how to use LaTeX. You can find vast amounts of information on LaTeX online. But this is not entirely helpful. Much of it is written *for* scientists; most of it is written *by* people who are very capable technologically. While there are certainly philosophers who are technically inclined, comfortable with using the command line and/or writing code, etc., most of us are not. This is a guide to using LaTeX

for those who either are not technologically inclined *or* who do not have the time to poke around and figure *everything* out on their own.

LaTeX can seem overwhelming and complex. It needn't be. There are three basic steps to producing your first LaTeX document. If you want to stop there, you will reap (practically) all the benefits of LaTeX. If you want to go further, it seems there is no end to the amount you can learn. My objective in these pages is to get you to a workable understanding of LaTeX with a minimum effort (on your part). After reading these pages, you should know how to write *all* your papers using LaTeX. If you want to jump ahead, click on the next page below. If you would like to know a bit more about LaTeX before we begin, read on.

Writing a paper in LaTeX is more complicated than writing a paper in a word processor in two ways. First, you have to be more explicit using LaTeX than you would with a word processor. Second, there are a few steps between typing the words on your keyboard and producing a finished document in LaTeX, not so with a word processor.

LaTeX is, in the first instance, a markup language. So when you 'write a paper in LaTeX', what you are really doing is creating a plain text file (with a .tex extension—so paper.tex, for example) filled with your content *and* some explicit instructions. The basic instructions are not that complicated, and your text editor will help you write them, but you do have to be explicit about inserting them.

The instructions you write in that plain text file are used in the steps between typing the paper and producing a finished product. Once you have a file, you run it through LaTeX. This is the step of compiling your paper, turning it from a plain text file into a PDF. (Note, there is the option to turn your paper into a .dvi file. I am certain there is *some* reason to want to do this. But I know that I have never had such a reason, and I bet you will be like me in this regard, so I will ignore that here.) All the little bits of code you explicitly insert into your paper are instructions that LaTeX uses to translate the .tex file into a .pdf file.

Writing a paper in a word processor is kind of like a packaged cupcake. All the formatting work is done for you. Writing a paper in LaTeX is kind of like baking a cake: you mix up raw ingredients and you don't have anything very special—you have a sort of ugly plain text file full of code that, were you to print, would be confusing to someone who didn't know LaTeX. But just as you have to put a cake in the oven to produce delicious goodness, so too must you put your .tex file through a process to get out the typesetting-goodness that is a LaTeX document.

If you do not already have a TeX distribution on your system, click here to continue; if you *do* have a TeX distribution, click here. (If you don't know if you have a TeX distribution, look at the applications or programs on your computer. If you have a Mac and see TeXShop, or if you have a PC and see TeXworks or TeXnicCenter, then you have a TeX distribution

installed.)

2 Obtaining LaTeX

First, you must obtain a TeX (that is, LaTeX) distribution. This is not particularly difficult.

2.1 Installing on a Mac

If you have a Mac, the best option to download and install MacTeX. If you download and install the main package there, you will install a number of different items. Most importantly, you will install the most recent TeX distribution. These are the tools you need to take a document from a marked-up plain text file to the lovely document you think of as being 'written in LaTeX'. Without this bit, you cannot do anything with LaTeX.

The MacTeX package is great because of all the tools it includes. You *could* write all your LaTeX files (your .tex files) in a text editor like TextEdit, and then compile them by typing in commands in Terminal.app. But my guess is that if you are reading this document, you are not going to feel comfortable doing all that. Moreover, though a .tex file is just a plain text file, having an editor dedicated to LaTeX (or at least one that has lots of LaTeX related features built in) is priceless.

So what does the MacTeX package include? For my money, the big three items for philosophers will be TeXShop, BibDesk, and Excalibur. I'll talk about the first two in detail later, but for now I will note that TeXShop is where you will actually compose your paper—it is sort of like your new version of Word or Pages. BibDesk is your new bibliography management tool. And Excalibur is a spell checker.

Download MacTeX. Follow the usual installation instructions. Proceed to the next page.

2.2 Installing on a PC (running Windows)

2.2.1 On your own computer

Disclaimer: I do not have a PC. I did install LaTeX on a PC once, but it has been a while and my memories are fuzzy. So some of my PC-specific instructions might be a bit off. Please let me know if I lead you astray so I can fix any bad advice. LaTeX is platform neutral. So, *for the most part*, the things I say here will apply for both Macs and PCs. I will offer a few

Mac-specific hints, but I bet there are PC analogues for them. Again, if you want to make suggestions, I would be thrilled to have them. Email me at tanksley [at] virginia [dot] edu.

If you have a PC, you will want to download and install MiKTeX. If you download and install the main package there (the 'Basic MiKTeX Installer'), you will install a number of different items. Most importantly, you will install the most recent TeX distribution. These are the tools you need to take a document from a marked-up plain text file to the lovely document you think of as being 'written in LaTeX'. Without this bit, you cannot do anything with LaTeX.

So download MiKTeX and follow the usual installation instructions. (If you need more detailed instructions, take a look at this page.)

You need a good LaTeX-specific text editor to begin your journey. I believe that MikTeX now comes with TeXworks, but I could be mistaken about that. (It used to come with TeXnicCenter, but I am pretty sure it no longer does.) There are other options out there, but these two are pretty straightforward and user friendly, so I suggest you download and install one of them if neither comes with MikTeX.

With your TeX installation in hand, proceed to the next page.

2.2.2 If you cannot install software

If you use a PC at your office or a lab, you might not have an account that lets you install software. If your computer is a PC, you can still install LaTeX, you just have to install it a little differently. Instead of installing MikTeX, you need to get a USB Flash Drive (or thumb drive) and install MikTeX Portable on it. All the files you need will be written to your flash drive. Just plug it in a USB port and you can use TeXworks on any PC you like. (Note that later I will discuss the bibliography management application JabRef. Here is a direct link to a 'portable app' version of that program you can put on your flash drive, too. Note: I got that link from here, I have not tried it out.)

If this is not a good option for you for some reason, you can still use LaTeX. There are an increasing number of capable online LaTeXeditors and compilers you can use for free. Google has made a preview release version of their latexlab available online. I have not done any serious work on it, but it looks pretty good, and people seem to love Google Docs, so it at least runs with a good crowd.

With your TeX installation in hand (or with an online version to use), proceed to the next page.

2.3 Installing on a Linux machine

Installing a TeX distribution on a Linux platform is straightforward. Just use your package manager to find the texlive package, then install it. This will vary from system to system as the package managers vary. Just be sure to install `texlive`—if there is an option, install `texlive-full`.

Installing texlive via your package manager (either the graphical version or on the command line) will get you a TeX distribution. This is the series of packages and commands that will take you from a markup-laced .tex file to a nice .pdf file. But the texlive package will not include a text editor to help you write said .tex files. You want a good text editor for this—it will make your LaTeX life much easier. Fortunately, there are many options out there.

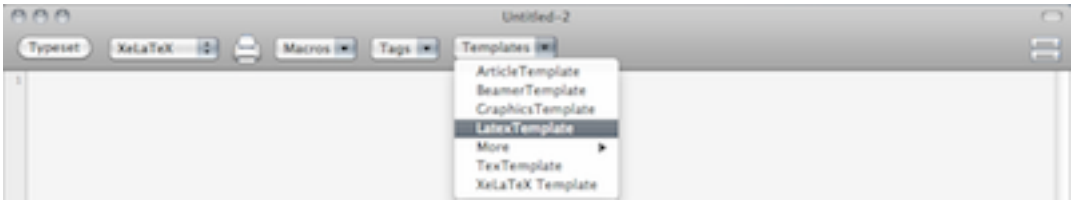
Kevin Kelment (who has given me lots of feedback about the Linux-and-LaTeX options) tells me that Gummi and Kile are very good, very capable LaTeX editors. Download one of them via your package manager, then proceed to the next page.

3 Your first LaTeX document

Your objective is to write a paper. To do that, you need to create a file of the right sort (that contains the right commands). To be more specific, you need to create a plain text file with a .tex file extension. So the file for your first paper might be called 'first_paper.tex'. (Those unfamiliar to the world of computers might be surprised to find out that there are a number of file extensions that a plain text file could take. I always thought they had to be .txt files, but LaTeX files are .tex files and your bibliography file will be a .bib file, both of which are plain text files; there are, in fact, quite a few different flavors of plain text files.)

I'd recommend writing your paper using either TeXShop (if you have a Mac) or TeXnic-Center (if you have a PC). Just start the program and you should get a new file (you may have to select `File -> New`). Whatever program you use, if it is a LaTeX-specific editor, once you save the file, it should have the right extension.

Every paper you write will need to include the same basic commands. It must open with a prologue. And it must end with an `\end{document}` command. Your editor ought to have some templates you can choose from that will do all this for you. Let them do the work when you are getting started (eventually, you can modify the basic template and make your own template; but that is a few steps down the road). When you open up TeXShop for example, there is a button marked 'Templates' at the top. If you click on that, you will get a drop-down menu of template options. Start with the LaTeX template and you will have *most* of the prologue written for you.



Here is what the TeXShop LaTeX template (henceforth, the LaTeX template) looks like:

```
\documentclass[11pt]{amsart}
\usepackage{geometry}
\geometry{letterpaper}
%\geometry{landscape}
%\usepackage[parfill]{parskip}
\usepackage{graphicx}
\usepackage{amssymb}
\usepackage{epstopdf}
\DeclareGraphicsRule{.tif}{png}{.png}%
    {`convert #1 `dirname #1`/`basename #1 .tif`.png}

\title{Brief Article}
\author{The Author}
%\date{}  % Activate to display a given date or no date

\begin{document}
```

This will *almost* work for us. The **\documentclass** command tells LaTeX very general formatting information about your paper. The amsart document class is for math papers. For papers that are a little more philosophy-like, you'll want to use the Article class. And you will probably want your paper to be in 12 point font. So change out the first line to this:

```
\documentclass[12pt]{article}
```

Before you begin writing the content of your paper, you want to do three things. First, change the stuff in the scope of the **\title** command to be the title of your paper. Second, put your name in for 'The Author'. Third, make sure there is a **\maketitle** command right after the **\begin**{document} command. This will be there if you use a template. This command tells LaTeX to write your name and title on your paper.

Now you can begin writing your paper.

3.1 Writing the content of your paper

For the most part, writing the content of a basic paper is no different than writing a paper using a word processor. There are a few things you need to be aware of, however.

1. You must end a paragraph with a blank line (alternatively, you can type two backslashes). A single return will not be read as the start of a new paragraph. You must enter two returns.

As an example, note that this

```
19
20 This is one paragraph.
21 This is another.
22
23
```

will give you a single line of text:

This is one paragraph. This is another.

But this

```
20 This is one paragraph.
21
22 This is another.
```

will give you two separate paragraphs:

This is one paragraph.
This is another.

This is a slightly odd habit to learn, but once you do, it is pretty straightforward. One nice thing is that LaTeX will take care of spacing for you. So you needn't worry about indentation. Nor need you make sure you get two spaces after a period and only one after each word; LaTeX treats multiple spaces as if they were one, and it sorts out the spacing to make your lines look right. Additionally, LaTeX reads three or seven or twenty blank lines as if they were one; you do not have to have *exactly* one blank line between each paragraph, though you must have *at least* one blank line.

2. If you want to *italicize* or **bold** some text, you have to use the proper commands. Any text you put inside the scope of a `\emph{}` command will be italicized; any text in the scope of a `\textbf{}` command will be bold. (If you were to *nest* `\emph{}` commands, however (e.g., `\emph{some \emph{italicized} text}`), the `\emph` command has the effect of toggling italics (so your output would be: '*some* italicized *text*'.) Note that on the toolbar of TeXShop there is a drop down menu called 'Macros'; under that menu you will find a 'Text Styles' heading, then an 'Emphasize' (or 'Typeface' then 'Bold') entry. In TeXnicCenter, there is a little button on the bottom left of the toolbar that you can click to make text bold or italic, just like in Word. I imagine you will find a similar button on most major editors.

3. LaTeX does not know how to handle quotation marks very well. If you type a standard double-quote mark ("), LaTeX will always compile it as closing quotation marks. So at the start of a quotation, single or double, you need to use a backtic—that looks like this "' and is typically on the same key as the tilde ~. To finish a quote, you use a double apostrophe. You never use the double-quote symbol. In order to close a single-quote, you use a single apostrophe.

Fortunately, most LaTeX-specific text editors are sophisticated enough to correct this. So if you type a double-quote you'll end up getting a pair of back ticks followed by a pair of apostrophes, with your cursor right in the middle where you want it. So while you need to be aware of this bit of archaic LaTeX behavior, it should likely not alter the way you work at all.

4. If you want to start a new section, simply type `\section{Title}`, filling in the title of the section. If you want a subsection, type `\subsection{Title}`, filling in the title. LaTeX will take care of the numbering for you.

5. There are some special symbols in LaTeX. If you type a percent sign, for example (%), LaTeX takes that to mean that it should ignore everything that follows on the line. So if my .tex file were to include the following:

```
This is only a test. % Look out for cliches!
```

The compiled document would look like this:

This is only a test.

This means the above sample prologue is full of comments LaTeX will ignore. These are

simply notes to explain to you what the various commands do and what your options are. LaTeX likewise gives special meaning to backslashes, curly braces, and ampersands. So if you want to type any of these symbols, you must *escape* them; that is, you must put a backslash directly in front of them: `\% \{ \}`. But if you backslash-escape a backslash (e.g., `\\`), you will get the symbol for a new line. Instead, if you need to use a backslash you must write `\backslash`.

6. Relatedly, spaces can be slightly tricky when using LaTeX. One of the nice things about LaTeX is that a very advanced algorithm takes care of inter- and intra- word spacing. You needn't put two spaces between sentences; LaTeX will put the best spacing between them given the context. But because of this, LaTeX is somewhat ignorant about periods: it assumes every period ends a sentence. So it will put the same amount of space between the words `Mr . Jones` as it does between the sentences `composition. But Lewis`. To get the spacing right, you need to tell LaTeX to use one space in the first case and just let it use the default behavior in the second. To do this, you would write `Mr.\ Jones`. The backslash tells the LaTeX compiler that you mean there to be only one space between those words.[1]

If you want your spacing to be perfect, you'll always stick a backslash after a period that does not terminate a sentence. I don't run into occasions to do this very often. Sometimes I'll need to do it with an `etc.\ but` or an `et al.\ argue`, but these are pretty rare. The good news (for me at least), is that if you happen to forget to use the backslash space, the spacing difference is typically not noticeable. At least, it isn't for me. I'm sure there are some occasions where I would really want to be perfectly precise, but I usually don't care too terribly much. In fact, I forgot about this point until Greg mentioned it to me in an email—this entire document is written without any such backslash spaces after abbreviations! Nevertheless, I thought Greg was right and I should mention it here for the sake of completeness. Consider yourself notified.

7. You must tell LaTeX about footnotes explicitly. To do this, use the `\footnote{Footnote text.}` command. Do not leave a space between the command and the punctuation it is to follow, else you will get an odd space. Your footnotes will be embedded in your text when you are working on your .tex file. (They'll be lovely footnotes in the .pdf, of course.) For me, this was one of the hardest parts of the transition to LaTeX. It can be difficult to figure out where a footnote begins and ends when it is embedded in the paragraph, and it is difficult to change footnote 23 when the notes in the .tex file lack numbers. TeXShop (and TeXnicCenter, I think) have a feature that helps here: Sync. When you are in a .pdf or a .tex file, right click (or Command + click, or Control + click, depending on your setup) on any part of either file and you will be placed more or less in that same spot in the other file. It is not perfect, but it helps.

[1] Thanks to Gregory Wheeler for suggesting I include this here.

Another suggestion for dealing with footnotes: since LaTeX ignores anything after a percent sign, you can write something like the following to set your footnotes apart in the .tex file:[2]

```
end of sentence.%
%
%
%
\footnote{The footnote text.}
%
%
%
```

8. If you like, you can do your citations like you probably used to: you can write `Lewis (1986)` when you want to cite *On the Plurality of Worlds*, and then create a section called 'Bibliography' where you type in the bibliographic information. A better way is to use BibTeX and BibDesk or JabRef. I'll explain those more later.

9. Finally, note that LaTeX is kind of picky about dashes and hyphens. If you want to type a hyphen, as in 'sub-set', you want to type a single hyphen or minus sign. If you want to type a dash *between page numbers* you want to type two hyphens (so `22--24`). If you want an *em dash*, you need to type three hyphens (so `that is the view---though I`). And if you need a minus sign you type a hyphen within the scope of the math environment (an opening and closing dollar sign mark that environment within a paragraph, so `$-3$` would give you negative three).

3.2 Ending the paper

After you have finished writing the content of your paper, you need to put a `\end{document}` command. Anything you put after this command will be ignored when you compile your paper.

3.3 Compiling your paper

You have written a .tex file at this point. To be honest, it is kind of an ugly mishmash of code and philosophy. It is emphatically not the finished product you are going for. (Incidentally, it does have a beauty all its own: it is a plain text file. This means that in ten years when your

[2] Thanks to Alex Skiles for this suggestion I had never seen.

colleagues can no longer open their old Word documents using whatever word processor is all the rage, you will be able to open your files with ease using any text editor you like.)

To turn your paper from a .tex file to a .pdf file you need to compile it. You could do this via the command line, but TeXShop and TeXnicCenter make this easy for you. In TeXShop, just click the 'Typeset' button in the menu bar and then click on 'LaTeX'. In TeXnicCenter, make sure the button the the toolbar says `LaTeX -> PDF`, then click 'Build' in the toolbar and select 'Build Output'. (I'll discuss some of the other options later.) A little box—the console window—will pop up that prints a record of the typesetting operation (I think in TeXnicCenter, the console window might be at the bottom of the window; in TeXShop it is free floating). If everything goes according to plan, a .pdf document will pop up on your screen.

A word of warning: When you typeset your document, LaTeX has to generate a number of temporary files for its own use. It does not delete them automatically. There is a 'Trash Aux Files' button on the console in TeXShop that allows you to delete these files. Nevertheless, it is a good idea to put your .tex file in a folder or directory before you compile it—that way the cruft is contained and not spread all over your desktop or Documents folder.

When you write a very basic document—one with no references, internal or external—you will only need to compile it a single time. But once you start working with a bibliography or internal references to sections or footnotes, you'll need to compile the document multiple times. This simply means you click on the `Typeset -> LaTeX` button twice. But we can worry about that complication later.

3.4 Error messages

There is a minor hassle with LaTeX that you tend to avoid with word processors: error messages. Sometimes things go wrong in the conversion from .tex to .pdf; when things go wrong, compiling will stop and an error message will appear in the console window. Part of really learning to use LaTeX is learning to read these error messages. And if your use of LaTeX is anything like mine, part of using LaTeX is trying hard to ignore lots of these messages!

Though I am sure there is some way in which this is ill-advised, I always hit <Enter> after I get an error message. Sometimes LaTeX is just letting you know that something isn't quite up to its rather exacting standards. In such a case, it is happy to go on compiling once you dismiss the error message.

But sometimes you aren't so lucky. Sometimes the error keeps repeating. In such cases, you can click the 'Go to Error' button on the console and it will usually take you to the vicinity of the offense (sometimes the errors aren't in a specific location). When I started

using LaTeX, the vast majority of error messages I received were from brackets or commands I had forgotten to close, so that is likely a good first place for you to look.

If you type `\emph{` and never close that bracket, LaTeX does not know what the emphasis ranges over. So it returns an error message. *Every* command in the body of your paper has a range, and you will get an error unless you specify the start *and* the end of that range. There are two ways to do this. Inline commands—commands like `\emph{}` and `\footnote{}`—mark their range with brackets. Block commands—such as the command for a block quote or a numbered list—mark their range with `\begin{}` and `\end{}` commands. So a block quote is marked off in the following way:

```
\begin{quote}
The quote text.
\end{quote}
```

If you do not include either of those commands, you will get an error.

There are other ways in which you can make LaTeX throw an error message. This is one place LaTeX is more costly than using a word processor. One nice thing is that LaTeX will always give you an error *message* to let you know what is going wrong. When you are new, you won't always be able to tell what those messages mean, but rest assured that they do mean something, and they typically mean something quite specific. Do a search on the error message and you'll find someone online explaining how to solve your problem (here is a nice place to start your search).

3.5 Next steps

The foregoing is sufficient to get you a basic paper. But most philosophy papers will require more than this. For one thing, you want an easy way to interact with your bibliography. For another, you'll want to double-space your paper to submit it to conferences and journals. And you might want to change the font or play around with other options. The next step I'll cover is the bibliography. But you can click here to learn about customizing your paper.

4 Creating and managing a LaTeX bibliography

There are three steps to effective bibliography management:

1. Creating a .bib file.

2. Insert references into your .tex file.

3. Format your references.

I'll describe each step in detail below.

4.1 Creating a .bib file

All your bibliography information will be stored in a structured file with a .bib extension. When you compile your document, LaTeX will look at that file, pull the right references, and stick all and only the right references in your works cited.

Sounds easy enough: stick all your bibliography information in one file, cite your references as you go, and let the computer sort out the bibliography. And it is (basically) that easy. I include the parenthetical because a .bib file is a highly structured file, and if you don't have the right tools, creating one is a bit of a hassle.

As an example, here is my entry for Lewis's *On the Plurality of Worlds*:

```
@book{LewisOPW,
    Address = {Oxford},
    Author = {David Lewis},
    Date-Added = {2009-10-13 11:33:52 -0400},
    Date-Modified = {2009-10-13 11:33:52 -0400},
    Publisher = {Basil Blackwell},
    Read = {Yes},
    Title = {On the Plurality of Worlds},
    Year = {1986}}
```

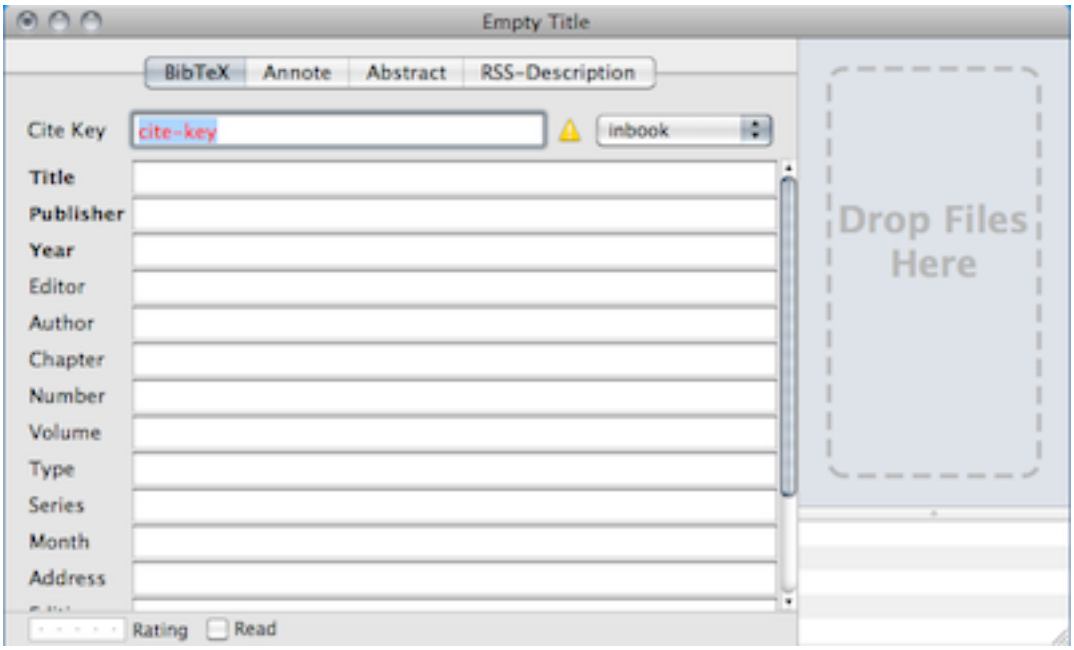
For the most part, it is clear what each line of this file is doing. The first says we are talking about a book, not an article, and then it assigns a little code to the book. My code is the author's last name and then the first letters of each word in the title (or a memorable combination of some of those letters). This is the *cite key* for this reference. It is *very* important. It must be unique to the reference in question. I'll come back to cite keys in the next section.

The rest of the lines indicate where the book was published, who the author is, when I added this file to my bibliography, the book publisher, whether I've read the work, the title of the book, and the year it was published. Like I said, pretty straightforward. But can you

imagine putting every reference from your dissertation, a book, or even a single paper in that format? It seems miserable.

But there is a better way. If you have a Mac, when you downloaded MacTeX, you downloaded a program called BibDesk. BibDesk is going to solve all your problems. (If you have a PC, you are only a little less lucky. JabRef is the Windows counterpart to BibDesk. I've never used JabRef, but I'm guessing it is fairly similar to BibDesk.) BibDesk gives you a very nice graphical user interface for creating a .bib file.

BibDesk is straightforward, so I won't say too much about it. When you click to create a new reference, you get a nice window like this:



All you have to do is pick a *unique* cite key for the work, then fill in the publication information, and BibDesk will format the BibTeX file for you. (Notice the little area over on the right: if you drag a pdf file of a paper here, then when you do a search in your bibliography, BibDesk will search within the article. And if you happen to take notes on the pdf using Skim, BibDesk will search those, too.)

4.1.1 An important note

You *must* create your bibliography file in the right location. If you don't it can't be found when you compile your document, and that defeats the purpose. There are two locations

you can use.

You could put your .bib file in the folder that houses your .tex file. BibTeX (the compiler you run to turn cite keys in your .tex file into the format LaTeX needs to compile them) can find them that way. But if you do this, you obviously cannot use one big bibliography for all your papers (unless you keep everything in one folder). Alternatively, you *could* create a duplicate .bib file for every paper you write, but these would be difficult to keep synchronized.

There is a better option. To use it, you need to create a new folder. (This applies to Macs only. I don't know where you should put your bibliography if you use a PC. Sorry.) If you open up the Finder, it will likely come up to your main user folder. That should bear your user name as its name. (Mine is `charlietanksley`.) This folder contains a Documents folder, a Downloads folder, a Music folder, etc. You want to open the Library folder. Create a folder in that folder called `texmf`. In the texmf folder, create a pair of folders: one called `bibtex` and one called `tex`. We'll set the `tex` folder aside for now. In the bibtex folder, create another pair of folders: one called `bib` and one called `bst`. *Put your bibliography in the bib folder.* My bibliography can be found at the following location: `charlietanksley/Library/texmf/bibtex/bib/mybib.bib`. You should put yours at the same location *modulo* a pair of changes: your user name should replace my user name, and the name of your bibliography should stand in for `mybib.bib`.

If you put your bibliography here, TeXShop knows where to look in the compilation process. It also knows where to look to help you insert references into your document. And that, as we'll see, is a very big help.

4.2 Inserting references

You need to get references into your paper. You do this by calling the proper command with the proper argument. Recall, if you want to make some text bold face, you put it in the scope of a `\textbf{}` command. Similarly, if you want to cite a reference, you put it in the scope of the proper `\cite{}` command. But you don't put the *entire* reference between those curly braces: you only need to put the cite key for the reference. Since each reference has a unique cite key, if you put the cite key in a `\cite{}` command, BibTeX can put the right reference in when you compile the document.

There are two important components of creating a citation:

1. Calling the right citation command.

2. Inserting the right cite key.

I'll deal with each in turn.

4.2.1 The natbib package

I am pretty sure that LaTeX has some default citation resources built in. But I don't think they are that good. And it is *easy* to have exactly the commands you want. So I'll explain how to do that.

To generate the various references we philosophers need, you will want to use the natbib package. The package is rather well documented, so if you need more information, you should certainly check that out. (You can find documentation for *any* package in TeXShop by searching for it by name if you click on `Help` in the menu bar and then `Show help for package`. But you can find the natbib documentation here.)

There are three distinct elments involved in using the natbib package: invoking the package in your prologue, citing sources, and controlling the look of the final bibliography section.

Natbib in the prologue

Calling natbib is quite easy. All you need to do is put the `\usepackage{natbib}` command somewhere in your prologue (i.e., somewhere between `\documentclass{article}` and `\begin{document}`). This will tell LaTeX to process your citations using the natbib style and will tell LaTeX how to make sense of your various citation commands.

While we are in the prologue, I recommend using one more command. I like to use the `\setcitestyle{aysep={}}` command right after the `\usepacakge{natbib}` command in the prologue. This makes it the case that author and year are separated with a blank space rather than a comma. So I would get `Lewis 1986` instead of `Lewis, 1986`.

So, to sum up, I put this in the prologue:

```
\usepackage{natbib}
\setcitestyle{aysep={}}
```

And that takes care of calling the natbib package.

Citation commands

Not all citations are alike. Sometimes you need a page number, sometimes you don't. Sometimes the citation is a parenthetical, sometimes it is part of the subject of the sentence. Sometimes you need the author and year, sometimes you just need the year.

Since there are multiple styles of citations you might need, there are multiple citation commands.

Here is a chart that includes the commands I use most commonly (on the right) and their output (on the left). Recall that 'LewisOPW' is my cite key for *On the Plurality of Worlds*.

Output	Command
(Lewis 1986)	`\citep{LewisOPW}`
Lewis (1986)	`\citet{LewisOPW}`
(1986)	`\citeyearpar{LewisOPW}`

Table 1: natbib commands

There are other commands, but these are the ones I use most often. (I don't aim to be comprehensive here. See pages 7, 8, and 9 of the natbib documentation for extensive and clear tables of all your options.)

I think you can do almost everything you want with these commands. Two things you might want to do: add specific page numbers and cite multiple papers. These are easy additions.

To add page numbers, simply insert the page number or numbers in square brackets ([ ]) between the cite command and the cite key. So this command `\citep[202--203]{LewisOPW}` would produce '(Lewis 1986, 202—203)' (note the double hyphen between the page numbers; you need this to get LaTeX to use the right dash for a range of numbers).

To add a comment before the citation, put that in another set of square brackets before the page number set. To get '(see Lewis 1986, 202)' you would write `\citep[see][202]{LewisOPW}`. And if you want the 'see' but not the page number, you would write `\citep[see][ ]{LewisOPW}`. (Since page numbers are more common than 'see's, LaTeX assumes a single argument in square brackets is a page number. So if you want text before the citation but no page number, you need the empty square brackets.)

To cite multiple sources in one location, simply separate the cite keys with a comma. So `\citep{LewisNWTU,OPW}` will produce '(Lewis 1983, 1986)'. And if you cite multiple papers by one author in a year, LaTeX will stick the 'a' on the first one, a 'b' on the second, etc.

Controlling the final bibliography

One nice feature of using explicit citation commands is that it lets LaTeX automate the creation of your bibliography. If you are writing a long paper or have many sources, this is very helpful. (One of the most time-consuming and frustratingly tedious parts of writing my dissertation in a word processor was making sure every cited reference appeared in the

bibliography and that every reference in the bibliography was cited. I know now that there are citation managers you can add onto most word processors, but I didn't know that then. LaTeX builds this in.)

But since LaTeX is really geared towards scientists, the bibliography you get is not going to look like you want. Even if you use the natbib package. But you can easily customize the bibliography's format.

First, we need to write the bibliography. All you need to do to do this is put the following command wherever you want the bibliography to appear: `\bibliography{your_bibliography_file}`. (Note: *do not* include the file extension here.) When LaTeX sees that command, it will print the bibliography. If you want the bibliography to be printed at the end of the paper, just stick that command one line above the `\end{document}` command.

Now, we need to customize the look of the bibliography. You do this by specifying a style via the `\bibilographystyle{your_style}` command. You can put this anywhere in the document you like, so long as it comes between the `\begin{document}` and `\end{document}` commands. The natbib documentation suggests you put it right after the `\begin{document}` command. I like to put it right above the `\bibliography{your_bibliography_file}` command. But it doesn't matter where you put it.

You have two options for customizing the look of the bibliography via the `\bibliographystyle{}` command. The most straightforward is to use one of the styles that come with the natbib package. Both the `dcu` and `kluwer` styles are similar to a traditional philosophy bibliography. (Page 14 of the natbib documentation lists all the available options, if you want to nose around.) The other option is to use a custom style file.

You can find custom style files in various places on the internet. If you find one you like, stick it in `/User/Library/texmf/bibtex/bst/`. That is one of the folders we created earlier (recall, your .bib file is in `/User/Library/texmf/bibtext/bib/`.) I use a style file that puts my bibliography in the style favored by the *Philosophical Review*. To do this, all you need to do is grab a copy of the style file (the .bst for Phil Review, as he calls it) from Ted Sider's page, stick it in the right folder (described above), and then call it from the `\bibliographystyle{}` command. You can find other style files around, but this one looks nice enough, so I use it.

Summary

That is it. If you use natbib, these three citation commands (plus optional arguments to include page numbers and notes), and call a style file you like with the `\bibliographystyle{}` command, you'll end up with a nice, well formatted, complete bibliography with minimal work. That is a good thing.

So, to review, your paper will look something like this:

```
\documentclass[12pt]{article}

% the rest of the prologue

\usepackage{natbib}
\setcitestyle{aysep={}}

\begin{document}

\maketitle

% Body of your paper\

\bibliographystyle{dcu} % or \bibliographystyle{Phil_Review}
   or whatever
\bibliography{your_bibliography_file}

\end{document}
```

4.3 Footnote citations

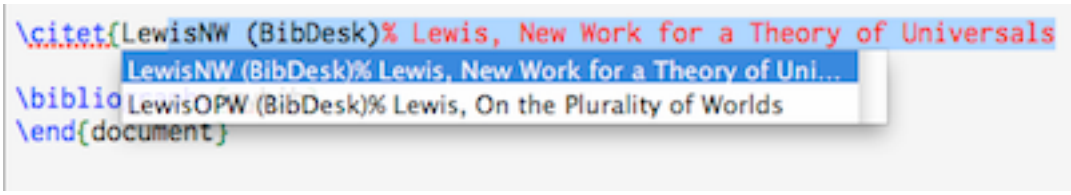
You need to follow a slightly different set of directions if you want to do full citations in footnotes (where, e.g., the first footnote that mentions a text includes the full citation information). I've written a detailed post on my blog (www.charlietanksley.net/philtex/proper-footnote-citations-with-latex/). Since this is a slightly more advanced topic, I'm not going to discuss the details here. But you can read the details on the blog at the address above, or you can download a pdf version at http://www.charlietanksley.net/pdf/latex-footnote-citations.pdf.

4.3.1 TeXShop and BibDesk integration

There is one hitch in the bibliography management system described above. Instead of remembering that you want to cite Lewis's *On the Plurality of Worlds*, you have to remember that you want to cite that book *plus* the unique cite key for the book. This isn't so hard for the books and papers you cite all the time, but it is difficult for papers you cite infrequently.

Coming up with a cite key generation system that works for you will help this a little bit. But it won't really solve the problem. Thankfully, the people who wrote TeXShop have come up with an elegant solution to this problem. (A number of text editors have similar fixes for this problem. You can find a very similar solution for Vim and TextMate. In the near future I hope to play around on a PC and see how TeXWorks handles this issue. But for the time being, I'll only talk about TeXShop.)

In TeXShop, to use this functionality, simply start typing your cite command, then hit the F5 key once you type the open curly bracket. (BibDesk needs to be running for this to work; but if it is not open, the most recent version of TeXShop will open it. If you have trouble getting this to work, try opening BibDesk.) You'll get a little popup window with your bibliography in it that you can navigate with the up and down arrows. Hit when you get to the entry you want, and TeXShop will stick it in your document. If you start typing part of the name, that will narrow down the list of options. So if I type `\citep{Lew` and hit F5, I get the following little list:



This is, obviously, a quite handy feature. Just select your reference, hit , then type your } and you are done.

4.3.2 Printing your citations and bibliography

Once you have written a paper full of `\citet{}` commands and have called the bibliography in a style you like, you need to compile your paper correctly. If you just compile it using LaTeX, you will not get a bibliography. You will get a question mark everywhere you wanted a citation, and you will get a blank section called 'References' at the end of the document.

To get your bibliography and citations to print, you need to compile your document in multiple steps. First, compile it using LaTeX. This creates a list of the cite keys you used (I think that is what it does.) Then, compile it using BibTeX (you'll find this in the same menu as the 'compile with LaTeX' feature). This generates a file with the relevant bibliographic information. Now compile your document with LaTeX *again*. This is the step where LaTeX is told what each cite key corresponds to, and it isn't until this step that you get question

marks replaced with citations. I think that should do it. But I always compile with LaTeX one more time just to be sure all the details get ironed out.

So when you are ready to get a pdf file, you will

1. Run LaTeX.
2. Run BibTeX.
3. Run LaTeX.
4. Run LaTeX.

Then you are done. You have your first full paper.

Though I think a big selling point of LaTeX is the fact that you can produce beautifully typeset documents with a minimum of effort, I'd be lying if I said I just use the basic article document class with no adornments. You can while away many an afternoon getting your papers to look *exactly* like you want. You can argue about whether this is time wasted. But if you are interested in tweaking your document in this way, read on to find out about a few packages and options to alter the look of your document.

5 Further steps

You likely want to customize your paper in some way or other. At the very least, you'll need to double-space it to submit it to a journal or conference. But you also might want to use a different font, use some fancy math or logic symbols, add a custom header or footer, or change the margins of your paper. In this document, I'll tell you enough that you'll be able to do each of those things to some degree or other (LaTeX is complicated; I don't claim to be comprehensive here).

Note: I am going to talk about using various packages in this document. I am pretty sure that most of these are included in a standard TeX distribution. If you don't have one of these packages, either use your package manager (TeX Live Utility for the Mac; I think that MikTeX might actually download packages you need on the fly, i.e., if you specify a package in the prologue that you do not have installed, MikTeX will install it without you doing anything), or search for the package at CTAN and follow the directions to install it.

5.1 Spacing

There are three main options for double-spacing your paper.

The first is to use the setspace package. Anytime you 'use the X package', for any package X, you do so by adding it in your prologue via a `\usepackage{}` command, perhaps with some additional arguments, and perhaps with some additions to the body of the document. To use the setspace package you need to do two things:

1. Add `\usepackage{setspace}` to your prologue.
2. Add `\doublespace` at the start of your document (likely after `\maketitle`).

If you want to single-space the bibliography, you can just add `\singlespace` before the `\makebibliography` command. Anything you want single-spaced can be made single-spaced by adding this command (and just add the `\doublespace` command when you want to switch back).

I like the `setspace` package. One thing it *does not* do is double-space your footnotes. I like that, but some journals do not. To double-space *everything*, you'll want to use the second of the three spacing options.

The second option is to redefine the default spacing between baselines (the bottom of letters like 'b'). You do this by adding the following line to your prologue:

```
\renewcommand{\baselinestretch}{2}
```

If you prefer 1.5 spacing, simply put a 1.5 in for the 2. Since this redefines the baseline for the document, it will change the spacing for footnotes as well as the main text.

The third option is a bit more powerful than the first two. It makes more global changes than just double-spacing things, but these are changes you might like (specifically, it converts footnotes to endnotes, changes the spacing of the document, alters the indentation of block quotes, and uses smaller fonts for the title, headers, etc.). Gregory Wheeler has put together a package specifically for philosophy papers. To use it, just go to his LaTeX for Philosophers page and download the `philosophy.sty` package. If you are on a Mac, put this in the `user/Library/texmf/tex/latex` directory you manually created; if you are on a PC, I gather that you ought to put it in the `latex` folder within MikTeX, where this will likely be `C:\Program Files\MiKTeX 2.5\tex\latex`, though the version number might differ.

Once you have downloaded `philosophy.sty` and put it in a directory where LaTeX can find it during compiling, you need to specify that the package be used. This will require a change to your prologue and a change near your bibliography.

Put the following in your prologue:

```
\usepackage{endnotes,philosophy}
```

This tells LaTeX to use two packages: the philosophy package you want to use and the endnotes package that the philosophy package relies on.

To get the endnotes to print, you need to explicitly put a `\noteshere` command where you want the notes. This will likely be just before or just after your bibliography. Add this command in and your paper will be double-spaced. It will also have what you might think is a more traditional look.

5.2 Fonts

At some point you will likely want to use a new font. LaTeX is quite old, and it is made to interact with only a small set of fonts. The easiest way to use a new font is to use XeLaTeX. (For our purposes, we can think of XeLaTeX as a super-set of the LaTeX commands—if you compile your paper using XeLaTeX, you can use all the LaTeX commands and packages, but you can also use some packages that go beyond LaTeX. Relevant here are the packages that let you use fonts easily.) If you use XeLaTeX, your final pdf file can use *any* font you have on your computer.

To use XeLaTeX you need to do three things.

First, you need to insert the following at the very start of your prologue (*even before the* `documentclass{article}` *command*):[3]

```
%!TEX TS-program = XeLaTeX
%!TEX encoding = UTF-8 Unicode
```

Then you need to stick the following in your prologue somewhere:

```
\usepackage{fontspec,xltxtra,xunicode}
\defaultfontfeatures{Mapping=tex-text}
\setromanfont[Mapping=tex-text]{Hoefler Text}
\setsansfont[Scale=MatchLowercase,Mapping=tex-text]{Gill Sans}
\setmonofont[Scale=MatchLowercase]{Andale Mono}
```

[3] This is only if you are using a program like TeXShop, that can determine which compiler to use based on this line. You likely don't need this if you are using a different editor.

If you are using a serif font in your document, just change `Hoefler Text` to any font you want to use—you could even use Times New Roman if you need to. (If you are using a sans-serif, change the `Gill Sans` to whatever you like.)

Now write your paper like normal. When you are ready to compile it, do not compile it with LaTeX. Instead, compile it with XeLaTeX (this will clearly be an option wherever you ordinarily select to compile with LaTeX).

The major text editors have an XeLaTeX document template that will include all these basic bits for you. In fact, I pulled the prologue elements above directly from the XeLaTeX template in TeXShop.

5.3 Margins

The default LaTeX margins seem huge at first. They are larger than 1 inch in order to make the document easier to read. But you might want to change the margins because you think they are too big or because you need specific margins for a dissertation or thesis. Changing the margins is easy. There is (or should be) a line in your prologue that says `\usepackage{geometry}`. This package can take margin sizes as arguments. So to get 1 inch margins all around, just write

```
\usepackage[right=1in,left=1in,top=1in,bottom=1in]{geometry}
```

And if you want larger or smaller margins on either side, just change that number.

5.4 Symbols

One of the beauty of LaTeX is the way it typesets formulae. If you write logical notation, LaTeX is wonderful.

Each symbol is called by a command, and each command has a specific mode. Just as you type a command to make some text italic, you type a command to get LaTeX to print a symbol. For example, the code for the existential quantifier (the backwards 'E') is `\exists`. And the universal quantifier is called by the `\forall` command.

Modes. Most of the commands you want for logic can only be used in *math mode*.

To put a bit of text in math mode, surround it with either (a) dollar signs or (b) backslashes and parentheses. So you would write either `$logic stuff$` or `\(logic stuff\)`. To write 'there is an x' in symbolic notation, I would type `\(\exists (x)\)`. (The space is

necessary. It tells LaTeX that the command is finished. If you typed `\(\existsx\)`, LaTeX would think you meant some other command, the `\existsx` command.) I have always used the dollar signs, but you can use the backslash parentheses method as well.

All text in math mode is italicized, though not all symbols are. So if you don't want your 'x' italicized in the above example, you would want to type `\(\exists\)(x)` (the dollar sign terminates the command, so you do not want an extra space here).

Ordinarily, you write in text mode. There are some symbols you can use in text mode. The most common are the accents on letters (like the little hat on the 'u' in Noûs) and, for me at least, the little section symbol. You get the section symbol by typing `\S` in text mode (so `\S 2` for example—don't forget the space). You get the accents by typing the right command followed by the letters it ranges over in brackets (so `No\^{u}s`, for example). Strictly, the braces are not necessary, but they don't hurt, and I like to use them because they help me see what is going on.

Combining math and text. Sometimes you might want to write out a premise or claim using a combination of symbols and text. Any text within the scope of the math mode will be italicized (and some letters will look kind of funny). So you should make sure the words are outside of the scope of the math mode. The most elegant way to do this is to just wrap the words in a `\mathrm{}` or `\text{}` command. For example, you might write `\(\exists x)\ \mathrm{some\ text}\ \therefore\ y.\)` (more on those extra backslashes in a moment). The (less elegant) alternative is to end the math mode before your text and then start a new math mode before your next symbol. I can see some situations where this might be a good option, but I think the `\mathrm{}` is better in most instances.

Spacing in math mode. Spaces are collapsed in math mode. If you leave a dozen spaces between two characters, they will be eliminated and your characters will appear side by side. This will happen even when you are inside a `\mathrm{}` command. To avoid having all your words and symbols run together, you merely need to stick a backslash at the end of every word you want followed by a space. Hence the extra backslashes in my example in the last paragraph.

There are a number of useful resources for learning or entering these commands. Gregory Wheeler put together a shortish pdf document that lists all the symbols you will probably need. You can find that here. There is a very long, comprehensive guide to symbols at CTAN here.

Perhaps more helpful than those guides is your text editor. If you are using TeXShop, just click `Window -> LaTeX Panel` in the menu bar and you will get a floating window listing all the symbols you might want (*plus* all the environments you might want, like 'itemize', 'quote', etc.). Just click the symbol you want and TeXShop will put it in the document,

though it will not add the dollar signs for math mode-only symbols. I am pretty sure that TeXnicCenter has something similar. But from a first-pass, I don't seen any such option in TeXworks. If there isn't such a helpful menu, Wheeler's short guide is easy enough to use, and you will learn the codes you use often pretty quickly. (There is also a free iPhone app called LaTeX Help that lists all the basic symbols.)

Some symbols are only available to you if you use the right package. I follow TeXShop's lead and always include the `\usepackage{amssymb}` command in my prologue, as this allows me to use all the basic math symbols. And there are some symbols you might want to use that aren't defined in a package. At his LaTeX for for Philosophers page, Gregory Wheeler has a list of symbols of interest to logicians. These aren't part of any package, but he describes what you need to put in the prologue of your paper to be able to use the particular symbol you want.

5.5 List Environments

The LaTeX list environment is very helpful for writing philosophical papers. Using it will allow you to write nice numbered premise arguments and inset definitions.

There are three kinds of lists: itemized lists, enumerated lists, and description lists. They are similar in structure, but they produce different sorts of lists.

5.5.1 Itemized lists

An itemized list is a bulleted list. It is unordered. To create an itemized list you simply write

```
\begin{itemize}
  \item Your first item.
  \item Your second item.
  \item Etc.
\end{itemize}
```

(Your text editor likely has a command or a button for inserting each sort of list.)

5.5.2 Enumerated lists

An *enumerated* list is a numbered list. This is the sort of list you would use for an argument. To write a list like this, you replace `itemize` above with `enumerate`:

```
\begin{enumerate}
  \item First item.
  \item Second item.
  \item Third item.
\end{enumerate}
```

This will give you the classic:

1. First item.
2. Second item.
3. Third item.

There are a couple of ways you might want to customize such a list. You might want to use different numbers (use (1*), (2*), etc.), you might want to start counting at a higher number, and you might want to use the therefore sign instead of the last number.

Changing numbers. To change the numbers your argument uses, you need to do two things. First, you need to insert the `\usepackage{enumerate}` command somewhere in your prologue. Second, after the `\begin{enumerate}` command put the counter you want to use in square braces. For example, if I want to enumerate with (1*), etc., I would write

```
\begin{enumerate}[(1*).]
  \item First item.
  \item Second item.
  \item Third item.
\end{enumerate}
```

Note that *all and only* the characters you put in those square braces will show up in your list (though it will count up from 1). So if you want parentheses around the number, add them; and if you want a period after the number, add it.

Changing starting number. Changing the starting number is pretty straightforward. You merely need to insert the `\setcounter{enumi}{desired number - 1}` command just after the `\begin{enumerate}` command, setting the number in the curly braces after `{enumi}` to one less than the first number you want in your list. So if I need to pick up my argument with premise five I would write

```
\begin{enumerate}
\setcounter{enumi}{4}
```

```
  \item Fifth item.
  \item Sixth item.
\end{enumerate}
```

Ending with a conclusion. You are using LaTeX. You might as well make use of the symbols at your disposal. So to wrap your argument up with the little three dot 'therefore' sign instead of the next number in the list, just do the following:

```
\begin{enumerate}
  \item First item.
  \item Second item.
  \item[\(\therefore\)] Conclusion.
\end{enumerate}
```

The square brackets directly after an `\item` command override the item counter with whatever you put there. In this case, I have set the counter for the final list item to be the symbol called by the math mode `\therefore` command. To make this work, you have to use it in math mode. (Note that if you needed to, you could change the counter of *any* premise to *anything* you like.)

Often in LaTeX, there is more than one way to achieve your goal. Kevin Klement suggested to me that there is a better way of manipulating your numbered lists. I can't decide if it is a better way or not. I think each has advantages. I'll describe this other method here. This method has one clear advantage: you can easily *resume* the numbering of a list that has been interrupted by text.

This method relies on the `enumitem` package. For a normal numbered list, you just use the same `\begin{enumerate} ... \item ... \end{enumerate}` commands like above. But here is an example to show how the more complicated features work, starting with a bit of the prologue:

```
... start prologue
\usepackage{enumitem}
... end prologue

\begin{enumerate}[start=4, label=(\arabic**).]
  \item Fourth item.
  \item Fifth item.
\end{enumerate}
```

```
Some text explaining something or other.

\begin{enumerate}[resume, label=(\arabic**).]
  \item Sixth item.
  \item Seventh item.
\end{enumerate}
```

The forgoing illustrates three key features of the `enumitem` package for numbered list management. The `\begin{enumerate}` command is followed by an optional list of arguments in square brackets. Those arguments dictate what the list will look like. A `start=#` argument dictates the starting number for the list (note that if you put '4' here the list starts with 'd', unlike the method above. Second, the `label=` argument dictates what the counter looks like. `\arabic*` gives you standard numbers, `alph*` gives you lowercase letters, `\Alph*` gives you uppercase letters, and `\roman*` and `\Roman*` give you lower and uppercase roman numerals, respectively. (Note that even if you are counting with letters, if you want to start counting at 'd' you write `start=4`.) Third, if you want the current list to continue where the last list left off, you can write `resume` instead of `start=#`. This is nice—if you add or remove elements from the earlier list or lists, you don't have to manually update all your other lists. But do note that you need to specify the counter style every time, as in my example.

5.5.3 Description lists

The *description list* is a way of providing an explicit definition. You write a description list using the following commands:

```
\begin{description}
  \item[Thesis] My thesis.
\end{description}
```

The term you are defining goes in the square braces after the `\item` command. (And if you want to add elements to your list, add them with more `\item` commands.) In the current example, the output will be a block indented paragraph that looks like this:

> **Thesis**: My thesis.

The description list is, obviously, a great way to highlight theses you want to make clear and then refer back to.

5.6 Headers

The default LaTeX paper header and footer is kind of bland. It just includes the page number, centered in the footer. Not bad, but not great, either. To customize this, use the fancy header package. This is likely installed by default; if not, use TeX Live Utility or let MikTeX install it for you. You can find the full documentation here, but here are the basics you need to know to use it right away.

First, put the following in your prologue:

```
\usepackage{fancyhdr}
\usepackage{lastpage}
\pagestyle{fancy}
\lhead{}\chead{}\rhead{}
\lfoot{}\cfoot{}\rfoot{}
\renewcommand{\headrulewidth}{0.0pt}
```

Whatever you put in the scope of `\lhead{}` will appear in the header on the left side. `\chead{}` and `\rhead{}` put text in the center and right side of the header, respectively. And the `\lfoot{}`, etc. commands do the same for the footer.

The `lastpage` package is optional. If you use it, then you can put a 'page m of n' kind of page number in the header or footer. To do this, just put `\thepage\ of \pageref{LastPage}` wherever you want that to appear. (Note that you need to compile the paper twice with LaTeX to get this to work right—the first time, LaTeX counts the pages; the second time, it enters the number in the right place.)

Second, you want to add a `\thispagestyle{empty}` command right after your `\maketitle` command. This gives you a blank first page. If you wanted, you could change the 'empty' out for a 'fancy', and your fancy header would show up on page one. But if you have anything in the header, that will show up above your title. And that just looks weird.

5.7 More

There are lots of great packages you can use to customize your LaTeX document. I plan on detailing those I use (and new ones I come across) on my blog devoted to matters technical: PhilTeX. If you have any questions, take a look around over there or email me at tanksley@virginia.edu.