Ph3 LaTeX Week 1: Obtaining and installing the software, compiling your first document

Eric D. Black

August 18, 2021

1 Introduction

LaTeX is a document preparation system widely used in academia, especially in math and science. Its main competitor is Microsoft Word, with which you may already be familiar. The two programs take very different approaches to typesetting a scientific document. Microsoft Word follows the paradigm of WYSIWYG, or What You See Is What You Get, the idea being that you type in your text, figures, and formulae in a window that visually represents what the printed output is going to look like. You are basically generating content and typesetting at the same time. Sounds great, but it doesn't always live up to its promise. Latex is a *markup language*, much like html. Operating a markup language is very much like programming. You write, typically in a simple text editor, a *source code* that contains commands that tell the *compiler* what you want your final document to look like, and then run the compiler to generate your output. Of the two the WYSIWYG program is by far the easiest to use if you have no prior knowledge or training. That is its main selling point, and for writing a simple document such as a business letter it is usually fine. However, when you want to include mathematical formulae or figures in your document, Word will often fall down and fail you, hard. All the time you saved by not having to learn to code in the beginning now gets eaten up and more as you try to get your document to format correctly.

Latex, on the other hand, does require some coding knowledge, but the little bit of time you spend at the beginning learning it will be paid back many times over when your final document just comes out right on the first (or maybe the second) try. Also, you only have to learn how to code once. Learning latex is not hard, and we are going to spend a little bit of our time in ph3 learning its basics. By the end of the term you should know enough to typeset a journal article or start a dissertation in it.

One of the great benefits of using a markup language is that it separates generating content from typesetting, leaving you free to concentrate on *what you are going to say* rather than what it is going to look like. Some people

will even go so far as to write out their content longhand and then, only after crafting their paper in an undistracted environment, will they typeset it using latex. I've done both, and while the quality of my writing is usually better when I write with pen and paper, for expediency's sake I usually create my document at the source code level, generating content as I type. That's what I'm doing now.

A note about pronunciation: The name of the program, latex, is pronounced "lay-tek." Some people call it "lah-tek," and that's ok. What's not ok is pronouncing it the same way as the non-vulcanized rubber product used to make surgical gloves, balloons, and other things. The spelling is the same, but the meaning and pronunciation are completely different. The term latex refers to a set of packages that run under the parent program, TeX (pronounced "tek"), a professional typesetter's program. TeX allows you to control all aspects of the printed output. Latex is designed for scientists and mathematicians writing papers and handles those details for you, allowing you to concentrate on the content of your document rather than its appearance.

2 Obtaining LaTeX

For this class we will be using a web-based compiler known as Overleaf.

Exercise 1: Go to `https://www.overleaf.com/edu/caltech` and set up your account. Overleaf comes in full professional or free "lite" versions. The full professional version is free to members of the Caltech community, and you will use your standard SSO username and password to access it. Like Mathematica or any other site license, it is a perk included in your tuition!

If you want to install it on your own computer you can do that too. Latex is available for linux, MacOS, and Windows. It is free and is covered under a license similar to the gnu public license, so there is no fee for downloading, installing, or using it. It was developed by Donald Knuth, a professor of computer science at Stanford, and Leslie Lamport, who at the time worked for SRI International, formerly known as the Stanford Research Institute. Knuth is a Caltech alum.

You can download the program to your computer from several sites, the most authoritative one being the Comprehensive Tex Archive Network, at `https://ctan.org`. I usually use either a linux distribution on a server maintained by the PMA division or TeXShop Pro (`https://pages.uoregon.edu/koch/texshop/obtaining.html`) on my local MacOS machine, but I am preparing this document in Overleaf. I like TeXShop Pro's editor better, but Overleaf has two main advantages. First, it works on any platform, and second, it allows collaborative editing of documents. This second feature is likely to be of much use to you in the future, so Overleaf is what we will learn on.

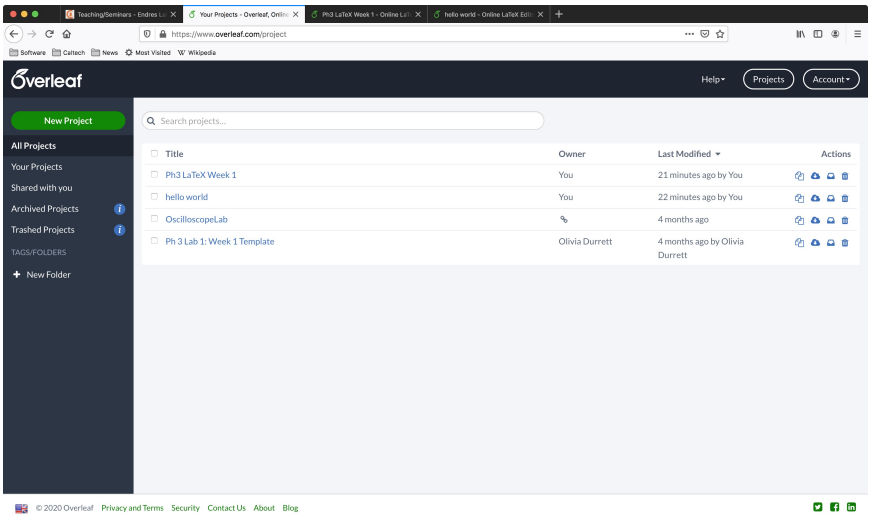

Figure 1: Overleaf opening screen listing all projects.

3 Generating your first document

Once you have your account and log in, you should see a screen that looks something like what is shown in Figure 1, though of course your project list will be different.

Click the big button that says "New Project," and a menu should pop up with several different choices of document type. Here you have a wide variety of templates to choose from, many of which you are likely to find are actually useful. Today, choose the first option, "Blank Project," and when prompted name it "hello world." After this you should be greeted with something like Figure 2.

In your browser window you should now see, as in Figure 2, two large, white frames, along with some peripheral buttons. The frame on the left is your *source code*, and the frame on your right is the *output*, or the document as it will appear when printed. You will notice that overleaf automatically generates several lines of code, pre-populated with things like your username and the title of your project. This code is the minimum necessary to compile and generate an output, and overleaf automatically compiles it for you this first time. Each line lists a command, which is a word preceded by a backslash (\) and followed by an argument in pointy brackets. In some cases the command has one or more options in square brackets as well, *i.e.*

```
\command[option1,option2,etc.]{argument}
```

This is the basic syntax of latex. A backslash alerts the compiler that what is about to follow is a command, rather than text that should appear in the output. So far you don't have any content in your document, so at this point

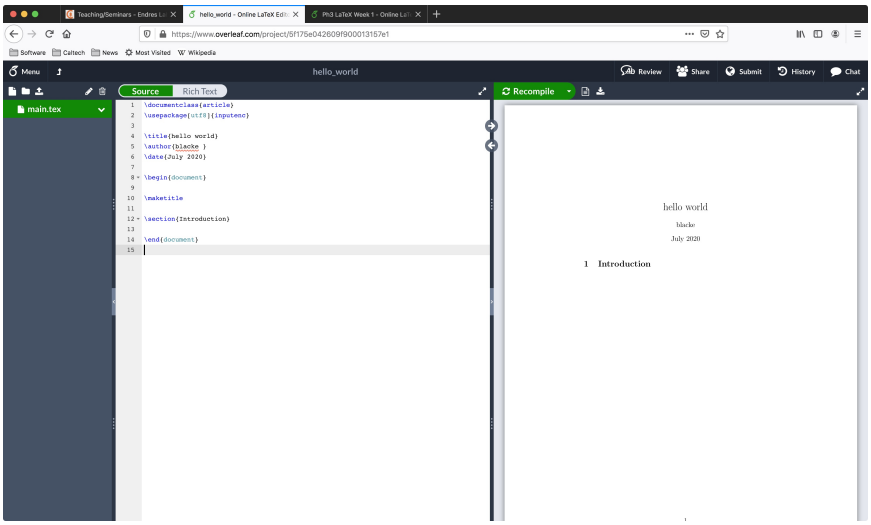

Figure 2: A fresh, new project.

everything in your source code is a command.

Exercise 2: Click on the line underneath the section command, and type in a sentence, *i.e.* something like

```
\section{Introduction}
Now is the time for all good rabbits to come to the aid of
the burrow!
```

Then click the "recompile" button just above your output screen. Now you have some content!

4 What the source code means

On or around Line 8 you will see a command

```
\begin{document}
```

Then at the end you will see a similar command that looks like

```
\end{document}
```

Your actual document is all the stuff between these lines, and these lines specify the beginning and ending of the *document environment*. Commands outside of the document environment tell the compiler important data about the document but don't make it to the printed output directly. Instead they are stored and then passed to commands that get called within the body of the document.

Commands that come before the \begin{document} are called the *preamble*. The first line specifies the *document class*, or what kind of document this will

be. There are several to choose from, the most common being *article, report, book,* and *letter*. There is a *proposal* class as well, which you may find useful later in your career. Use *article* for now. We will cover others later.

The second line instructs the compiler to *use* a *package*. This is similar to the #include directive in C/C++ in that it calls in an external library of special functions. We will explore several of these packages later, but for now overleaf recommends this one, so we will use it.

Next come three lines specifying the title, author, and date of the document. This is largely self-explanatory, but you may be surprised to see it come in the preamble, before the beginning of the document environment. You can think of these lines as storing values in the variables for title, author, and date, which then get passed to the command \maketitle in the body of the document. That command generates the actual title page in the output, after calling up the values for the author, title, and date variables that were established in the preamble.

Exercise 3: Replace the arguments of the title and author commands with "Ph3 LaTeX Homework 1" and your actual name, as opposed to your project and user name. For the date argument, replace the Overleaf-supplied month and year with the command

```
\today
```

and recompile.

5 Sectioning and spacing

Scientific articles are usually partitioned into sections, so latex makes this easy. As demonstrated in your "hello world" project, sections are started with the \section{Section Name} command. Sub-sections and sub-sub-sections are called out with the commands,

```
\subsection{A subsection}
\subsubsection{A sub-subsection}
```

and their output looks like this.

5.1 A subsection

5.1.1 A sub-subsection

There is no closing command for sections or subsections. They end when the next section or subsection begins. The compiler, of course, automatically numbers your sections for you.

All you have to do to make a new paragraph is hit return in your source code and start a new paragraph like you would in any other document. It doesn't matter if you have one carriage return or many, just as it doesn't matter how many spaces to type in between words or sentences. Latex will adjust the output to give the correct spacing automatically.

6 Math

Now we come to one of latex's strongest points, typesetting mathematical formulae. To do this you first have to tell latex to go into "math mode" by establishing a section of code inside a *math environment*, like so

```
Einstein showed that energy and mass are equivalent with the
relation \begin{math} E = m c^2 \end{math}.
```

Which of course gets typeset like so,

> Einstein showed that energy and mass are equivalent with the relation $E = mc^2$

Latex allows you to save some typing by abbreviating the \begin{math} and \end{math} commands with dollar signs, which is what I usually do for in-line equations. With that shortcut the source code for the example above would be

```
Einstein showed that energy and mass are equivalent with the
relation $ E = m c^2 $.
```

with the result being the same as before.

As the above example demonstrates, superscripts are made with the circumflex or caret symbol ^, which is above the number six on your keyboard (shift-6). Subscripts are made with the underscore symbol _ (shift-minus). Symbols are grouped together using pointy brackets { and }. This grouping is important, as it can mean the difference between $e^A \cos\theta$ and $e^{A\cos\theta}$.

Most math symbols are simply commands executed in the math environment, and their syntax is usually just a backslash followed by the name of the symbol you want. For example, Greek letters are just \alpha \beta \gamma \delta, etc., which show up in a math environment as $\alpha\beta\gamma\delta$. Capital Greek letters, such as $\Gamma\Delta\Theta\Sigma$, are mostly the same, except that their names begin with capital letters, *i.e.* \Gamma \Delta \Theta \Sigma. The only catch is that, where the Greek letter is the same as a Latin letter there is no special symbol for it. Capital alpha, for example, just looks like an A, so there is no separate command \Alpha. You would just type A if you needed a capital alpha.

Commonly-used functions work the same way. Trigonometric functions, for example, can be typeset like

```
\sin ( 2 \pi \omega t + \phi )
```

which comes out $\sin(2\pi\omega t + \phi)$. From here on out I'm going to omit the \begin{math} and \end{math} or $ and $ commands that would come before and after a formula unless they are necessary to make a point. Just remember that you have to start an environment before coding for mathematical expressions, and then end that environment before switching back to text.

A few mathematical symbols are so frequently used that they are abbreviated, which is annoying at first but handy once you get used to it. Some examples of these are the integral symbol $\int$ which is produced by the command \int, the symbol for infinity ∞ generated with \infty, and the syntax for generating a fraction \frac{numerator}{denominator}. For example,

```
\lim_{\tau \rightarrow \infty} \int_0^\tau \frac{1}{t} dt
\rightarrow \infty
```

This comes out to $\lim_{\tau \rightarrow \infty} \int_0^\tau \frac{1}{t} dt \rightarrow \infty$ in the in-line *math* environment, which looks kind of goofy. It looks much better in the standalone *equation* environment.

$$\lim_{\tau \rightarrow \infty} \int_0^\tau \frac{1}{t} dt \rightarrow \infty \tag{1}$$

Latex generally knows which functions are supposed to have subscripts below them, like limits or sums, and typesets those accordingly.

Exercise 4: Typeset the following in your hello world document using both the *math* and *equation* environments.

$$\frac{d^2y}{d\theta^2} + y = 0$$

Exercise 5: Typeset the following. Hint: Remember that the name for the symbol $\hbar$ is "hbar," pronounced like "aytch-bar," and that you turn a symbol name into a command by adding a backslash in front of it. Also, you can make two fractions next to each other by simply stringing together two fraction commands in the source code, *i.e.* \frac{}{}\frac{}{}.

$$\alpha = \frac{1}{4\pi\epsilon_0}\frac{e^2}{\hbar c}$$

Exercise 6: Typeset the following. Try this with both \Sigma and \sum, and see which works better.

$$\sum_{n=1}^{\infty} \frac{1}{x^n} = \frac{1}{1-x}$$

7 Comments

You can comment your source code using percent symbols %. Any text that follows a percent sign will not be displayed in the output, up to the next carriage return.

```
%A percent sign at the beginning of a line comments out the
whole line.  It still counts as the same line if the text
wraps by itself.  You have to hit return to start a new line.

On a new line, this text makes it into the output, %but this
does not.
```

Overleaf shows your line numbers in a grey bar to the left of your source-code frame, so you can tell at a glance where your line breaks are in your code.

8 The LaTeX symbol and explicit spacing

You make the fancy LaTeX symbol with the command \LaTeX\. The command starts with a backslash as you'd expect, but then it ends with another backslash, which you may not have seen coming. This second backslash is technically a separate command. It is necessary to get the spacing correct between the LaTeX symbol and the word that follows it. For some reason - I don't know why - spacing is normally suppressed after the logo, so if you leave off the second backslash you get the logo LaTeXwith the next word crammed right up against it like you see here. You force a space after the logo by a backslash followed by an ordinary space.

A backslash followed by a normal space is called a *short space* in typography. It is the normal space that falls between words in a sentence and is distinguished from a *long space*, which is longer and falls between sentences. Latex will normally make a short space after a word and a long space after a period or other punctuation mark. If you have a period inside a sentence, after an abbreviation, for example, latex will automatically insert a long space. To recover short spacing you would place a backslash followed by a space in your source code right after your punctuation mark.

```
Dr. Malcom, Dr. Sadler, if you please...
```

Dr. Malcom, Dr. Sadler, if you please...

```
Dr.\ Malcom, Dr.\ Sadler, if you please...
```

Dr. Malcom, Dr. Sadler, if you please...

This is a subtle point, and you usually don't have to worry about it. However, it reinforces the basic pattern that commands consist of a backslash followed by a relatively-obvious name. The command for a space is just a backslash followed by a space.

9 International language support

Latex is marvellously flexible as far as languages go. You can typeset in just about any language you care to, up to and including Asian languages such as Chinese, Japanese, and Korean, and right-to-left languages like Arabic and Hebrew. Typesetting in languages other than English is beyond the scope of this course, but I want you to be aware of this capability.

For further information on this subject see `https://www.overleaf.com/learn/latex/International_language_support`.

10 Downloading, sharing, and submitting your work

You can download all the files in your project or just the pdf output by clicking on the Menu icon at the top left of your screen and then making the appropriate choice.

Figure 3: Click on this icon in the top left of your screen to download your files.

There are two ways to share your project, and both are accessible through the Share icon in the top right corner of your browser window.

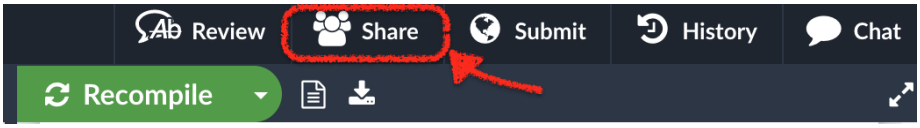

Figure 4: Click on this icon in the top right of your screen to share your project.

The first way is to enter your collaborator's email address (which is the same as their Overleaf username) and then click the Share button. This gives access to that person and only that person, while they are logged in to Overleaf.

The second way is called Link Sharing, and it allows you to generate a url that you can then send to anybody. This is sometimes more convenient than the first way, but it is also less secure since they can pass the link on to anybody else. Or the link can be intercepted in an email by a third party. Don't laugh. People competing for Nobel prizes do strange things. Ask me about Sam Ting some time. Fortunately, when you turn on Link Sharing you get two links, one that allows the wielder of the link to edit your project, and one that is read-only. Most of the time it is the read-only link you want to share.

Exercise 7: Comment your source code with your name and section, then share your project with the ph3 latex TA using Link Sharing. Copy the read-only (view-only) link, and paste it into the text box in this assignment.

References

[1] Lamport, Leslie *LaTeX: A Document Preparation System User's Guide and Reference Manual 2ed.*, Addison-Wesley Publishing Company 1994.

[2] `https://www.overleaf.com/learn/how-to/What_is_Link_Sharing%3F`