Introduction to LaTeX, the ECE Homework Template, and Some Basic `git`

Josh Jeppson

December 26, 2021

1 LaTeX in General

LaTeX documents are divided into two sections, the *preamble*, and the *document*. Anything outside of `\begin{document}...\end{document}` is part of the preamble, and anything within is part of the document. The escape character in LaTeX, like it is in C or Python, is \. \\, however, does *not* produce a backslash on your document. This produces a new line (can also be achieved by \newline).

There are two main concepts in LaTeX: *commands* and *environments*. Commands are like functions, which can take one or more arguments (placed in {} or [], depending on whether they are printed parameters or not), and environments are separated by the `begin` and `end` keywords.

1.1 Common Commands

More information can be found at `https://www.overleaf.com/learn/latex/Commands`.

Description	Command	Parameters
Boldface Text	`textbf`	{text to bold}
Italic Text	`textit`	{text to italicise}
Monospace Typeface	`texttt`	{text in monospace}
Math Equation	$$	$Math_{Equation} + Formatted$
Centered Math Equation	\[\]	Math equation
New Line	\\ or `\newline`	
New Page	`newpage`	
Picture	`includegraphics`	[width, height, etc]{path/to/picture.png}
Input Code File	`lstinputlisting`	[language=language]{path/to/file.cpp}

This is not an exhaustive list. Just some of the common ones. A full listing can be found at `https://tug.org/texniques/tn10/latex_cribsheet.pdf`, and a command "decoder" (you draw a symbol, it tells you the closest LaTeX command) can be found at `https://detexify.kirelabs.org/classify.html`.

1.2 Common Environments

More information can be found at `https://www.overleaf.com/learn/latex/Environments`.

Description	Command
Centered Text	`center`
Left-aligned Text	`flushleft`
Right-aligned Text	`flushright`
Bulleted List	`itemize`
Numbered List	`enumerate`
Table	`tabular`
Matrix (no borders)*	`matrix`
Matrix (normal)*	`bmatrix`
Tikz Picture	`tikzpicture`
Aligned Math Equations (numbered)	`align`
Aligned Math Equations (not numbered)	`align*`

*Indicates must be within an `align` (or derivative) environment.

2 The Template

I've tried to include a number of comments in the template, and I've created some custom commands and environments for you to use. These are *specific to the homework template*. Since LaTeX allows you to create custom commands, I have done this in the homework template, and these commands will not work in other documents!

Element	Type	Description
`problem`	environment	Where the problem description goes.
`answer`	environment	Where your answer/solution goes.
`finalAns`	command	The command used to show what your final answer is
`answersection`	environment	A subsection fo a problem.*
`course`	command	The course number
`studentName`	command	Your name
`aNumber`	command	Your A-number
`assnNumber`	command	The assignment number
`laplace`	command	Creates a Laplace symbol
`ilaplace`	command	Creates an inverse Laplace symbol
`fourier`	command	Creates a Fourier transform symbol
`ifourier`	command	Creates an inverse Fourier symbol

*Do not use on the *last* part of a multi-part problem or it will create an extra line.

I've also included a custom `lstlisting` code style (because the default is ugly), and some definitions to make drawing flowcharts easier in `tikz`. If you need help with any of this, come to my office hours.

3 Some Basic `git`

You do not have to use `git` to use the template. However, I highly suggest using it (if not for this then for your coding classes such as ECE 1400/1410 and beyond), since you'll be using it in industry.

`git` is a version control system written by Linus Torvalds. It is the de-facto code version control system in use today and if you write code in your careers (you will), you will most likely use it. `git` tracks file changes in a very space-efficient way and allows you to revert to old versions of files if need be. The quanta by which git uses to keep track of changes is called a "commit". Commits are a *state* of your repository that you can revert back to if necessary. To add file changes to a commit, use `git add [FILE]`. Files in a folder are *by default* not included in a repository and are **only staged for commit after they are *added.*** Once you have made changes, type `git commit -m [Message]`, or just `git commit`, after which a default text editor (generally GNU Nano) pops up and prompts you for a commit message.

You can see a status of changes made and ready for commits by using `git status`, and a list of previous commits using `git log`. If you are using a remote location for your repository (generally GitHub or GitLab and configurable using the `git remote` set of commands), you can *push* your changes to the remote using `git push`, or, if you have not configured a default remote, `git push -u [REMOTE NAME]`. You can add a remote by using `git remote add [REMOTE-NAME] remote@url:generally/an/SSH/URL.git`. The remote name is usually `origin`.

If you messed up and want to revert to an old commit, or want to undo staged changes, there are a few commands you can use. `git restore [FILES]` restores files to the state they were in the last commit. If you wish to restore them to a *specific* commit, you may use `git restore --source=[COMMIT HASH] [FILES]`. You can also use `git reset [COMMIT HASH]` to reset the *entire* repository to the state it was when that commit was made. Commit hashes are found in `git log`.

If you just want to see what the state of the repository was at a certain commit, you can use `git checkout [COMMIT HASH]`. `HEAD` is the latest commit on the current branch that you're working on. Additionally `git checkout [BRANCH NAME]` can switch you to an existing *branch*, and `git checkout -b [BRANCH NAME]` can create a new branch for you and switch you to it.

Obviously, `git` has far more functionality than this. You can create forks (duplicates of a repository linked to that repository), branches (a parallel line of commits that can be `git merge`'d back into the main branch), pull requests (when you want to merge changes in your fork back into an upstream repository), see the `blame` of who exactly edited what on a file, and `bisect` to find exactly where a bug occurred in $O(\log n)$ time. You can `stash` changes you don't want to commit, view the `diff` between two commits, `tag` a certain commit, and much more. Full documentation for `git` can be found at `https://git-scm.com/docs`.

Of course, for just editing the TEX files used for your homework, your `git`

workflow (if you choose to use `git`) will look something like this:

1. Use `makeHW.py` to create a file, fill it with problem descriptions.
2. `git add hwX-LASTNAME.tex`
3. `git commit -a -m "Created homework file for homework X"` (The `-a` option means commit *all* changes)
4. Work on homework
5. `git commit -a -m "Started problem 1"`
6. Work some more
7. `git commit -a -m "Finished problem 1"`
8. Etc...If you use GitHub, you will probably have a `git push` in there occasionally.