# *Help! I'm stranded on a desert island with nothing but a fully functional computer and a keyboard!*

*A primer to the terminal, pubnixes, and the tildeverse.*

*Adapted from handmadeweb.club for Aidan Quinlan's Handmade Web @ VCUarts GDES, October 2022 by e. couture*

# *Before We Get Started...*

We'll get into the nitty gritty in a second, but first, software!

For **Linux** users: you don't have to do anything, yay!

For **macOS** users: The current default terminal app will work for everything covered here. But, Apple offers many different shells on their machines by default, and you can switch between them to find one that behaves the way you like it!

I use bash, the Bourne Again Shell, which you can switch to using
`chsh -s /bin/bash`

For **Windows** users: Microsoft has since switched off of the UNIX subsystem it was originally built on, and as such their built in terminal, *Command Prompt,* will not work for a lot of this tutorial.

There are several options, including the Linux Subsystem for Windows, but for this tutorial I'm going to have all Windows users download Cmdr.

**A Note!:** Things on my screen will probably look different. Do not fret, if the text is the same, you're in same place.

# *A note on “terminal-ology”*

The words “terminal” and “shell” are used interchangably, I will probably switch between the two. Don’t worry. They’re the same thing.

(for most purposes)

# *First off, what even is the terminal?*

The “terminal” is a text based interface that allows you to navigate and interact with your computer completely with your keyboard.

or “command prompt” or “cmd” or “the command line” or “the command line interface” or “CLI”

Through the terminal, a whole new world of capabilities and customizations open up with your computer! You truly can become a “power user”.

You can navigate through your computer, automate repetitive tasks, speed up day to day tasks, and infinite other things with the terminal, in ways that you couldn’t even dream of with a traditional GUI (graphical user interface).

# *What is this UNIX thing you keep mentioning?*

UNIX was the first widely distributed operating system. Its first iterations came around in the 60s and 70s. Reserch it, it's really cool.

It no longer exists in its original form, at least in any widespread way.

Windows used to be on a UNIX subsystem, it no longer is.

MacOS runs on a heavily modified UNIX subsystem.

Linux and its many distributions are the closest things we get to modern UNIX systems.

# *Let's hop into the terminal!*

Hopefully by this point you all have gotten set up as much as you need to, and we can get going!

Go ahead and open up your terminal, whatever that may be and we'll talk over it.

Go for it! Type some stuff, hit enter! Mess around! See what happens!

# *The terminal for beginners, the very basics:*

A very important concept to grasp:

*All terminal commands are executed relative to where you are located in the filesystem at the time of executing the command*

This sort of “relative location” works the same way that it does with HTML and CSS filesystems.

You could, in theory, type out the entire path to a file each time you want to execute a command on it, but we don’t have to.

Enter, our first command!

# *The terminal for beginners, our first commands!*

First, lets get our bearings. Type `pwd` and hit enter.

`pwd` stands for “print working directory” and it will do just that! It will print the full filepath of wherever you’re located in your computer’s filesystem.

Now, to find out what else is here with us, we’re going to type `ls` and hit enter.

`ls` stands for “list” and it will list all of the files in the current directory where we’re located.

Both of these are incredibly useful commands, memorize these.

# *The terminal for beginners, lets move:*

Now that we know where we are, and what's here with us, let's talk about how to move around! This is where the whole "not having to type the whole file path" thing from earlier comes back into play.

Type `cd [a directory, or path to a directory]` and hit enter. You'll notice some things look a bit different. Let's talk about it!

# *The terminal for beginners, if all else fails:*

The `man` command stands for “manual” and it’s just that.  A user’s manual for the comand you type after it.

However, it’s not really the easiest to read or use. Type `man ls` and see what I mean.

To exit a `man` page just type `q`.

For a significantly more useful manual page, visit SS64. It’s chock full of very useful tools and information for not JUST the UNIX terminal! macOS, command prompt, and Windows Powershell as well!

# *Next steps, lets make some stuff!*

Moving and looking around is awesome, but let's actually make some stuff happen.

First, lets get back to our home directories.

Then, we'll learn a new command! The `mkdir` command allows us to make a new directory at the current locaton.

Nothing changes, but if we use an `ls` we'll see a new directory has appeared in our list! You can also head to your file browser and find it there!

Let's navigate to the new directory we just made.

Now, lets make a new file!

# *Next steps, weird command names*

The command for making a new file is ***`touch [filename]`***. Yeah. I know.

Files don't have to have extensions, but an extensionless file won't really be useful, except within the terminal.

Also, you only ever want to use the ***`touch`*** command with plain text files. These are things like HTML, CSS, JS, .txt, and .json files. Anything more complex encoding that that will likely not work. You can make them, but they won't do anything.

Lets make a new .txt file named "test".

If you open up your file browser and navigate to the folder we made earlier, you should see a new empty text file in there!

You should be able to open it and do whatever you want in there, and save it, as with any other text file.

Another fun command of note, the ***`cat [filename]`*** command will spit out the contents of a file. So if you type something in the text file we just created, and then save it. Try running ***`cat`*** on the text file we just created and see what happens.

# *Now, let's do something practical*

The stuff we're doing right now has probably felt kind of useless. Completely understandable. But. I promise the command line is actually really useful.

Let's look at an actually practical example. The `curl [url]` command allows us to take data or information from one server, and transfer it to another. In this case, our local machine.

The usecase for this that you'll probably come across most frequently is grabbing the text from specific github hosted files.

The `curl` command itself doesn't do anything with the data it recieves by default. It'll just spit it out in the console. Not really useful.

Try it out with this simple javascript library I've used in a few projects. Link here.

We've got two ways forward, let's talk about the simple one first.

If we put a simple angle bracket after our `curl` command with a file name, instead of putting that information right into the terminal it'll do something cooler with it. Try it out and then try a `ls` and `cat` command.

Cool huh?

# *Options! Options! Options!*

Now here's where stuff really takes off.

You may have noticed a large list of stuff with hyphens at the beginning of manual or SS64 pages, welcome to the wide world of options. Also, where the command line starts to get truly incomprehensible.

Options allow you to modify or extend the behavior of the command. These are where some commands get Really Really powerful.

To extend our last example. Let's take a look at the manual page for the `curl` command.

Now, if we use the `-o [filename]` option for the curl command, let's see what happens.

Now, the customary `ls` and `cat` commands...

Even cooler huh?

# *Next steps and more fun stuff*

**Some things to look into!**

- `cron` and the crontab, this is a way to automate tasks really easily.
- The pipe character `|` and nested commands, this is really where commands take off! You can use commands to manipulate and work with the output of other commands infinite other possibilities, it's awesome.
- Look into the original CLI interface for git! It's outrageously powerful and can even interface with github!
- If you don't plan on staying around for the second half of this workshop, look into tilde communities! They're really fun and passionate low-tech communities that in many ways embody the ethos of this class.

**Some useful tools:**

- Check out SS64 more in depth, if there's a complicated or weird syntax for a command, it's likely the manual page for it has a generator or a tool for it.
- `youtube-dl`. I swear by this program, I use it regularly, it's the only youtube to mp3 downloader that I will ever use. It's incredibly powerful and versitile. It also works for a SHITTON of other sites.
- For **MacOS** users: One of the most useful parts of Linux operating systems is the package directories. MacOS does not come with one by default. Install homebrew. It's so useful.

# *And now for a short break.*

*I've been talking for a while and need to stand up. Back in 5 for those who want to get a little more advanced!*

*For those who don't, thank you so much for listening! I appreciate you!*