TERMINAL

- If you are developers chances are you are going to be using command line at some point
- You may be installing npm dependencies as front end developers
- You may be logging to servers via ssh and performing more advanced tasks
- Navigating your file system
- ssh is secure shell

Why use shell/cmd

- ✓ Great control
- ✓ Do things with what we can't do with GUI
- ✓ Speed & Efficiency
- ✓ Access remote servers [deploy project to servers]
- ✓ There are some software's based on non GUI even front end devs face this thing now
- ✓ Getting hired {how to navigate through the file system}
- ✓ there are headless os - no gui

Terminal vs cmd vs shell

Terminal: mac centric, terminal is just an interface provided by the system where we can write our commands. Tool that we use to type and execute commands, used to be hardware now we use terminal emulator software

Command Line: the interface of the terminal, windows often calls their terminal program the command line

Shell: the program that the terminal runs the, os of the terminal

Cmd: windows centric

Gitbash UNIX: based terminal emulators

Many shells available

- Bourne shell (sh)
- korn shell (ksh)
- bourne again shell (bash) …

Common Terminal Commands

Common Terminal Commands

Key Commands & Navigation

Before we look at some common commands, I just want to note a few keyboard commands that are very helpful:

- `Up Arrow` : Will show your last command
- `Down Arrow` : Will show your next command
- `Tab` : Will auto-complete your command
- `Ctrl + L` : Will clear the screen
- `Ctrl + C` : Will cancel a command
- `Ctrl + R` : Will search for a command
- `Ctrl + D` : Will exit the terminal

Manual Command

On Linux and Mac, the `man` command is used to show the **manual** of any command that you can run in the terminal. So if you wanted to know more about the `ls` command, you could run:

```
man ls
```

Unfortunately, if you are on Windows and using Git Bash, the `man` command is not included, however, you can just type the command that you want to know more about and then `--help` and you will get similar info:

```
ls --help
```

You should be able to use the arrow keys or page up and down. When you are ready to exit, just press `q` .

The `whoami` Command

The `whoami` command will show you the current user that you are logged in as.

```
whoami
```

The `date` Command

Another really simple one is the `date` command, which, surprise, will show you the current date and time.

```
date
```

File System Navigation

Commands to navigate your file system are very important. You will be using them all the time. You won't remember every single command that you use, but these are the ones that you should remember.

Command	Description
pwd	Lists the path to the working directory
ls	List directory contents
ls -a	List contents including hidden files (Files that begin with a dot)
ls -l	List contents with more info including permissions (long listing)
ls -r	List contents reverse order
cd	Change directory to home
cd [dirname]	Change directory to specific directory
cd ~	Change to home directory
cd ..	Change to parent directory

Command	Description
cd -	Change to previous directory (which could be different than the parent of course)
find [dirtosearch] -name [filename]	Find location of a program

Of course, you can group flags together. For example, if I want to see more info and view hidden files, I could do `ls -l -a` and even shorten it to `ls -la`.

Opening a Folder or File

If you want to open a file or a folder in the GUI from your terminal, the command is different depending on the OS.

Mac - `open [dirname]` Windows - `start [dirname]` Linux - `xdg-open [dirname]`

You can open folders, files and even URLs

```
open https://traversymedia.com
```

Modifying Files & Directories

Command	Description
mkdir [dirname]	Make directory
touch [filename]	Create file
rm [filename]	Remove file
rm -i [filename]	Remove directory, but ask before
rm -r [dirname]	Remove directory
rm -rf [dirname]	Remove directory with contents
rm ./*	Remove everything in the current folder
cp [filename] [dirname]	Copy file
mv [filename] [dirname]	Move file
mv [dirname] [dirname]	Move directory

Command	Description
mv [filename] [filename]	Rename file or folder
mv [filename] [filename] -v	Rename Verbose - print source/destination directory

We can also do multiple commands at once with the `&&` operator:

```
cd test2 && mkdir test3
```

Right angle bracket >

This symbol tells the system to output results into whatever you specify next. The target is usually a filename. You can use this symbol by itself to create a new file:

```
> [filename]
```

When you are done, hit `ctrl+D`

The `cat` (concatenate) Command

The cat command is a very common command and allows you to create single or multiple files, view content of a file, concatenate files and redirect output in terminal or files.

The most common thing I use it for is to display the contents of a file:

```
cat [filename]
```

You can also view the contents of multiple files:

```
cat [filename] [filename]
```

You can also create a file using the `cat` command:

```
cat > [filename]
```

This will open up a new file and you can start typing. When you are done, you can press `Ctrl + D` to save and exit.

You can also append to a file:

```
cat >> [filename]
```

This will open up the file and you can start typing. When you are done, you can press `Ctrl + D` to save and exit.

You can use it to show line numbers:

```
cat -n [filename]
```

There are other uses as well, but as you can see, the `cat` command is very powerful.

The `less` Command

The `less` command is used to view the contents of a file. It is similar to the `cat` command, but it allows you to scroll up and down.

```
less [filename]
```

To exit the `less` command, just press `q`.

The `echo` Command

The `echo` command is used to display messages, or to create and write to files. It is similar to the `cat` command, but it is used to display a single line of text.

```
echo "Hello World"
```

You can also use it to create a file:

```
echo "Hello World" > [filename]
```

You can also append to a file:

```
echo "Hello World" >> [filename]
```

The `nano` Command

The `nano` command is a text editor that is installed by default on most Linux distributions, MacOS and you can even use it with Git Bash on Windows. It is very similar to the `vim` editor, but it is much easier to use.

You can open an existing file to edit or create a new file and open it with:

```
nano [filename]
```

When you're ready to exit, just hit `Ctrl + X` and then `Y` to save and `N` to not save.

The `head` and `tail` Commands

The `head` command is used to output the first part of files. By default, it outputs the first 10 lines of each file. You can also specify the number of lines to output.

```
head [filename]
```

You can also specify the number of lines to output:

```
head -n 5 [filename]
```

The `tail` command is used to output the last part of files. By default, it outputs the last 10 lines of each file. You can also specify the number of lines to output.

```
tail [filename]
```

You can also specify the number of lines to output:

```
tail -n 5 [filename]
```

The `grep` Command

The `grep` command is used to search for a text pattern in a file. It is very powerful and can be used to search for a string or regular expression in a file or set of files.

```
grep [searchterm] [filename]
```

You can also search for a string in multiple files:

```
grep [searchterm] [filename] [filename]
```

There are a lot more things that you can do with the `grep` command, but it's a but more advanced.

The `find` command

The `find` command is extremely powerful and is used to find the location of files and directories based on conditions that you specify.

To start off by creating something to work with. Let's create 100 files in the current directory. This is one of those things that I talked about earlier where you can do certain things much faster than you could in the GUI. We already know that the `touch` command will create a file. It can also be used to create multiple files.

```
touch file-{001..100}.txt
```

Now we have 100 .txt files in the current directory. Something that would have taken a lot longer to do in the GUI.

Let's do something very simple and find a specific file. The format looks like this:

```
find [dirname] -name [filename]
```

Let's find the file called `file-001.txt` :

```
find . -name "file-001.txt"
```

This will look in the current directory, which is represented with a dot.

We can look in other directories as well. Let's create a file in our home folder called test.txt

```
touch ~/test.txt
```

To find that file:

```
find ~/ -name "test.txt"
```

We can look for files that match a certain pattern as well. Let's find all files that start with `file-`:

```
find . -name "file-*"
```

We can search for files that are empty:

```
find . -empty
```

Let's append some text to the file `file-002.txt`. We could use the `cat` command, like I showed you earlier, but we can also use the `echo` command:

```
echo "Hello World" >> file-002.txt
```

Now if we find the empty files again, we will see that `file-002.txt` is no longer empty:

```
find . -empty
```

We can remove all of the files that we created with this command:

```
find . -name "file-*" -delete
rm -f file-* # This will also work
```

There is so much more that you can do with the `find` command, but it goes beyond the scope of this tutorial.

Piping

Piping is very powerful. It is a way of redirecting standard output to another destination, such as another file. Let's actually use the find command to find a list of files and then pipe them to a new file.

First, we'll create 10 files:

```
touch file-{001..010}.txt
```

Now, let's pipe the result from our find into a new file named `output.txt`

```
find . -name "file-0*" > output.txt
```

You can see the results now in the new file:

```
cat output.txt
```

Creating a Symlink

A symlink is a special type of file that points to another file. It is a shortcut to the original file. It is useful when you want to access a file in a different location without having to copy it.

We can use the `ln` command to create a symlink:

```
ln -s [filename] [symlinkname]
```

You can remove a symlink with the `rm` command:

```
rm [symlinkname]
```

If you're on Windows and you are not using something like Git Bash, you can use the `mklink` command:

```
mklink [symlinkname] [filename]
```

File Compression

`tar` is a program for concatenating multiple files into one big file called a **tarball** and reversing this process by extracting the files from the tarball.

Command	Description
tar czvf [dirname].tar.gz [dirname]	Create tarball
tar tzvf [dirname]	See what is in the tarball
tar xzvf [dirname].tar.gz	Extract tarball

- -c : Creates Archive
- -x : Extract the archive
- -f : creates archive with given filename
- -t : displays or lists files in archived file
- -u : archives and adds to an existing archive file
- -v : Displays Verbose Information
- -A : Concatenates the archive files
- -z : zip, tells tar command that creates tar file using gzip
- -j : filter archive tar file using tbzip
- -W : Verify a archive file
- -r : update or add file or directory in already existed .tar file

The `history` Command

Used to display the history of commands that you have run.

```
history
```

You can also use the `!` to run a command from the history.

```
!100
```

This will run the command that is in the 100th position in the history.