Command Line Crash Course

This appendix is a super fast course in using the command line. It is intended to be done rapidly in about a day or two and not meant to teach you advanced shell usage.

Introduction: Shut Up and Shell

This appendix is a crash course in using the command line to make your computer perform tasks. As a crash course, it's not as detailed or extensive as my other books. It is simply designed to get you barely capable enough to start using your computer like a real programmer does. When you're done with this appendix, you will be able to give most of the basic commands that every shell user touches every day. You'll understand the basics of directories and a few other concepts.

The only piece of advice I am going to give you is this:

Shut up and type all this in.

Sorry to be mean, but that's what you have to do. If you have an irrational fear of the command line, the only way to conquer an irrational fear is to just shut up and fight through it.

You are not going to destroy your computer. You are not going to be thrown into some jail at the bottom of Microsoft's Redmond campus. Your friends won't laugh at you for being a nerd. Simply ignore any stupid weird reasons you have for fearing the command line.

Why? Because if you want to learn to code, then you must learn this. Programming languages are advanced ways to control your computer with language. The command line is the little baby brother of programming languages. Learning the command line teaches you to control the computer using language. Once you get past that, you can then move on to writing code and feeling like you actually own the hunk of metal you just bought.

How to Use This Appendix

The best way to use this appendix is to do the following:

- Get yourself a small paper notebook and a pen.
- Start at the beginning of the appendix and do each exercise exactly as you're told.

- When you read something that doesn't make sense or that you don't understand, *write it down in your notebook*. Leave a little space so you can write an answer.
- After you finish an exercise, go back through your notebook and review the questions you have. Try to answer them by searching online and asking friends who might know the answer. Email me at help@learncodethehardway.org and I'll help you too.

Just keep going through this process of doing an exercise, writing down questions you have, then going back through and answering the questions you can. By the time you're done, you'll actually know a lot more than you think about using the command line.

You Will Be Memorizing Things

I'm warning you ahead of time that I'm going to make you memorize things right away. This is the quickest way to get you capable at something, but for some people, memorization is painful. Just fight through it and do it anyway. Memorization is an important skill in learning things, so you should get over your fear of it.

Here's how you memorize things:

- Tell yourself you *will* do it. Don't try to find tricks or easy ways out of it; just sit down and do it.
- Write what you want to memorize on some index cards. Put one half of what you need to learn on one side, then another half on the other side.
- Every day for about 15–30 minutes, drill yourself on the index cards, trying to recall each one. Put any cards you don't get right into a different pile; just drill those cards until you get bored, then try the whole deck and see if you improve.
- Before you go to bed, drill just the cards you got wrong for about 5 minutes, then go to sleep.

There are other techniques, like you can write what you need to learn on a sheet of paper, laminate it, then stick it to the wall of your shower. While you're bathing, drill the knowledge without looking, and when you get stuck glance at it to refresh your memory.

If you do this every day, you should be able to memorize most things I tell you to memorize in about a week to a month. Once you do, nearly everything else becomes easier and intuitive, which is the purpose of memorization. It's not to teach you abstract concepts, but rather to ingrain the basics so that they are intuitive and you don't have to think about them. Once you've memorized these basics, they stop being speed bumps, preventing you from learning more advanced abstract concepts.

Exercise 1: The Setup

In this appendix, you will be instructed to do three things:

- Do some things in your shell (command line, Terminal, PowerShell).
- Learn about what you just did.
- Do more on your own.

For this first exercise, you'll be expected to get your Terminal open and working so that you can do the rest of the appendix.

Do This

Get your Terminal, shell, or PowerShell working so you can access it quickly and know that it works.

Mac OSX

For Mac OSX, you'll need to do this:

- Hold down the command key and hit the spacebar.
- In the top right the blue "search bar" will pop up.
- Type "terminal."
- Click on the Terminal application that looks kind of like a black box.
- This will open Terminal.
- You can now go to your dock and CTRL-click to pull up the menu, then select `Options->Keep in dock`.

Now you have your Terminal open, and it's in your dock so you can get to it.

Linux

I'm assuming that if you have Linux, then you already know how to get at your Terminal. Look through the menu for your window manager for anything named "Shell" or "Terminal."

Windows

On Windows we're going to use GitBash. People used to work with a program called `cmd.exe`, but it's not nearly as usable as GitBash. If you have Windows 7 or later, do this:

- Click Start.
- In "Search programs and files," type "powershell."
- Hit Enter.

You Learned This

You learned how to get your Terminal open, so you can do the rest of this appendix.

NOTE: If you have that really smart friend who already knows Linux, ignore him when he tells you to use something other than Bash. I'm teaching you Bash. That's it. He will claim that zsh will give you 30 more IQ points and win you millions in the stock market. Ignore him. Your goal is to get capable enough, and at this level, it doesn't matter which shell you use. The next warning is stay off IRC or other places where "hackers" hang out. They think it's funny to hand you commands that can destroy your computer. The command `rm -rf /` is a classic that you *must never type*. Just avoid them. If you need help, make sure you get it from someone you trust and not from random idiots on the internet.

Do More

This exercise has a large "do more" part. The other exercises are not as involved as this one, but I'm having you prime your brain for the rest of the appendix by doing some memorization. Just trust me: this will make things silky smooth later on.

Linux/Mac OSX

Take this list of commands and create index cards with the names on the left on one side and the definitions on the other side. Drill them every day while continuing with the lessons in this appendix.

pwd print working directory

hostname my computer's network name

mkdir make directory

cd change directory

ls list directory

rmdir remove directory

pushd push directory

popd pop directory

cp copy a file or directory

mv move a file or directory

less page through a file

cat print the whole file

xargs execute arguments

find find files

grep find things inside files

man read a manual page

apropos find what man page is appropriate

env look at your environment

echo print some arguments

export export/set a new environment variable

exit exit the shell

sudo DANGER! become super user root DANGER!

Drill, drill, drill! Drill until you can say these phrases right away when you see that word. Then drill the inverse, so that you read the phrase and know what command will do that. You're building your vocabulary by doing this, but don't spend so much time you go nuts and get bored.

Exercise 2: Paths, Folders, Directories (pwd)

In this exercise, you learn how to print your working directory with the pwd command.

Do This

I'm going to teach you how to read these "sessions" that I show you. You don't have to type everything I list here, just some of the parts:

- You do *not* type in the $ (Unix) or `^gt;` (Windows). That's just me showing you my session so you can see what I got.
- You type in the stuff after `$ or >`, then hit Enter. So if I have `$ pwd` you type just pwd and hit Enter.
- You can then see what I have for output followed by another $ or > prompt. That content is the output, and you should see the same output.

Let's do a simple first command so you can get the hang of this:

Exercise 2 Session

```
$ pwd
/Users/zedshaw
$
```

You Learned This

Your prompt will look different from mine. You may have your user name before the $ and the name of your computer. On Windows it will probably look different too. The key is that you see the following pattern:

- There's a prompt.
- You type a command there. In this case, it's pwd.
- It printed something.
- Repeat.

You just learned what pwd does, which means "print working directory." What's a directory? It's a folder. "Folder" and "directory" mean the same thing, and they're used interchangeably. When you open your file browser on your computer to graphically find files, you are walking through folders. Those folders are the exact same things as these "directories" we're going to work with.

Do More

- Type pwd 20 times and each time say, "print working directory."
- Write down the path that this command gives you. Find it with your graphical file browser of choice.
- No, seriously, type it 20 times and say it out loud. Shhh. Just do it.

Exercise 3: If You Get Lost

As you go through these instructions, you may get lost. You may not know where you are or where a file is and have no idea how to continue. To solve this problem, I am going to teach you the commands to type to stop being lost.

Whenever you get lost, it is most likely because you were typing commands and have no idea where you've ended up. What you should do is type pwd to *print your current directory*. This tells you where you are.

The next thing is you need to have a way of getting back to where you are safe, your home. To do this, type `cd ~` and you are back in your home.

This means if you get lost at any time, type:

```
pwd
cd ~
```

The first command pwd tells you where you are. The second command `cd ~` takes you home so you can try again.

Do This

Right now, figure out where you are, and then go home using pwd and `cd ~`. This will make sure you are always in the right place.

You Learned This

How to get back to your home if you ever get lost.

Exercise 4: Make a Directory (mkdir)

In this exercise, you learn how to make a new directory (folder) using the `mkdir` command.

Do This

Remember! *You need to go home first!* Do your pwd then `cd ~` before doing this exercise. Before you do *all* exercises in this appendix, always go home first!

`Exercise 4 Session`

```
$ pwd
$ cd ~
```

```
$ mkdir temp
$ mkdir temp/stuff
$ mkdir temp/stuff/things
$ mkdir -p temp/stuff/things/frank/joe/alex/john
$
```

This is the only time I'll list the pwd and `cd ~` commands. They are expected in the exercises *every time*. Do them all the time.

You Learned This

Now we get into typing more than one command. These are all the different ways you can run `mkdir`. What does `mkdir` do? It make directories. Why are you asking that? You should be doing your index cards and getting your commands memorized. If you don't know that "`mkdir` makes directories," then keep working the index cards.

What does it mean to make a directory? You might call directories "folders." They're the same thing. All you did above is create directories inside directories inside of more directories. This is called a "path," and it's a way of saying, "first temp, then stuff, then things and that's where I want it." It's a set of directions to the computer of where you want to put something in the tree of folders (directories) that make up your computer's hard disk.

NOTE: In this appendix, I'm using the / (slash) character for all paths, since they work the same on all computers now. However, Windows users will need to know that you can also use the \ (backslash) character. Other Windows users may expect to see the backslash at all times, but this isn't necessary.

Do More

- The concept of a "path" might confuse you at this point. Don't worry. We'll do a lot more with them and then you'll get it.
- Make 20 other directories inside the temp directory in various levels. Go look at them with a graphical file browser.
- Make a directory with a space in the name by putting quotes around it: `mkdir "I Have Fun"`.
- If the `temp` directory already exists, then you'll get an error. Use `cd` to change to a work directory that you can control and try it there. On Windows, the desktop is a good place.

Exercise 5: Change Directory (cd)

In this exercise, you learn how to change from one directory to another using the `cd` command.

Do This

I'm going to give you the instructions for these sessions one more time:

- You do *not* type in the $ (Unix) or > (Windows).
- You type in the stuff after this, then hit Enter. If I have `$ cd temp`, you just type `cd temp` and hit Enter.
- The output comes after you hit Enter, followed by another $ or > prompt.
- Always go home first! Do pwd and then `cd ~` so you go back to your starting point.

Exercise 5 Session

```
$ cd temp
$ pwd
~/temp
$ cd stuff
$ pwd
~/temp/stuff
$ cd things
$ pwd
~/temp/stuff/things
$ cd frank/
$ pwd
~/temp/stuff/things/frank
$ cd joe/
$ pwd
~/temp/stuff/things/frank/joe
$ cd alex/
$ pwd
~/temp/stuff/things/frank/joe/alex
$ cd john/
$ pwd
~/temp/stuff/things/frank/joe/alex/john
$ cd ..
$ cd ..
$ pwd
~/temp/stuff/things/frank/joe
$ cd ..
$ cd ..
$ pwd
~/temp/stuff/things
$ cd ../../..
$ pwd
~/
$ cd temp/stuff/things/frank/joe/alex/john
$ pwd
~/temp/stuff/things/frank/joe/alex/john
$ cd ../../../../../../../
$ pwd
~/
$
```

You Learned This

You made all these directories in the last exercise, and now you're just moving around inside them with the `cd` command. In my session above, I also use `pwd` to check where I am, so remember not to type the output that `pwd` prints. For example, on line 3, you see `~/temp`, but that's the output of `pwd` from the prompt above it. *Do not type this in*.

You should also see how I use the `..` to move "up" in the tree and path.

Do More

A very important part of learning to use the command line interface (CLI) on a computer with a graphical user interface (GUI) is figuring out how they work together. When I started using computers, there was no "GUI" and you did everything with the DOS prompt (the CLI). Later, when computers became powerful enough that everyone could have graphics, it was simple for me to match CLI directories with GUI windows and folders.

Most people today, however, have no comprehension of the CLI, paths, and directories. In fact, it's very difficult to teach it to them, and the only way to learn about the connection is for you to constantly work with the CLI, until one day it clicks that things you do in the GUI will show up in the CLI.

The way you do this is by spending some time finding directories with your GUI file browser, then going to them with your CLI. This is what you'll do next:

- cd to the `joe` directory with one command.
- cd back to `temp` with one command, but not further above that.
- Find out how to cd to your "home directory" with one command.
- cd to your Documents directory, then find it with your GUI file browser (Finder, Windows Explorer, etc.).
- cd to your Downloads directory, then find it with your file browser.
- Find another directory with your file browser, then cd to it.
- Remember when you put quotes around a directory with spaces in it? You can do that with any command. For example, if you have a directory `I Have Fun`, then you can do `cd "I Have Fun"`.

Exercise 6: List Directory (ls)

In this exercise, you learn how to list the contents of a directory with the `ls` command.

Do This

Before you start, make sure you cd back to the directory above `temp`. If you have no idea where you are, use pwd to figure it out and then move there.

Exercise 6 Session

```
$ cd temp
$ ls
stuff
$ cd stuff
$ ls
things
$ cd things
$ ls
frank
$ cd frank
$ ls
joe
```

```
$ cd joe
$ ls
alex
$ cd alex
$ ls
$ cd john
$ ls
$ cd ..
$ ls
john
$ cd ../../../
$ ls
frank
$ cd ../../
$ ls
stuff
$
```

You Learned This

The `ls` command lists out the contents of the directory you are currently in. You can see me use `cd` to change into different directories and then list what's in them so I know which directory to go to next.

There are a lot of options for the `ls` command, but you'll learn how to get help on those later when we cover the `help` command.

Do More

- *Type every one of these commands in!* You have to actually type these to learn them. Just reading them is *not* good enough. I'll stop yelling now.
- On Unix, try the `ls -lR` command while you're in `temp`.
- On Windows do the same thing with `dir -R`.
- Use `cd` to get to other directories on your computer and then use `ls` to see what's in them.
- Update your notebook with new questions. I know you probably have some, because I'm not covering everything about this command.
- Remember that if you get lost, then use `ls` and pwd to figure out where you are, then go to where you need to be with `cd`.

Exercise 7: Remove Directory (rmdir)

In this exercise, you learn how to remove an empty directory.

Do This

Exercise 7 Session

```
$ cd temp
$ ls
stuff
$ cd stuff/things/frank/joe/alex/john/
$ cd ..
$ rmdir john
$ cd ..
$ rmdir alex
$ cd ..
$ ls
joe
$ rmdir joe
$ cd ..
$ ls
frank
$ rmdir frank
$ cd ..
$ ls
things
$ rmdir things
$ cd ..
$ ls
```

```
stuff
$ rmdir stuff
$ pwd
~/temp
$
```

WARNING! If you try to do rmdir on Mac OSX and it refuses to remove the directory even though you are *positive* it's empty, then there is actually a file in there called .DS_Store. In that case, type `rm -rf <dir>` instead (replace <dir> with the directory name).

You Learned This

I'm now mixing up the commands, so make sure you type them exactly and pay attention. Every time you make a mistake, it's because you aren't paying attention. If you find yourself making many mistakes, then take a break or just quit for the day. You've always got tomorrow to try again.

In this example, you'll learn how to remove a directory. It's easy. You just go to the directory right above it, then type `rmdir <dir>`, replacing `<dir>` with the name of the directory to remove.

Do More

- Make 20 more directories and remove them all.
- Make a single path of directories that is 10 deep and remove them one at a time, just like I did above.
- If you try to remove a directory with contents, you will get an error. I'll show you how to remove these in later exercises.

Exercise 8: Move Around (pushd, popd)

In this exercise, you learn how to save your current location and go to a new location with `pushd`. You then learn how to return to the saved location with `popd`.

Do This

Exercise 8 Session

```
$ cd temp
$ mkdir -p i/like/icecream
$ pushd i/like/icecream
~/temp/i/like/icecream ~/temp
$ popd
~/temp
$ pwd
~/temp
$ pushd i/like
~/temp/i/like ~/temp
$ pwd
~/temp/i/like
$ pushd icecream
~/temp/i/like/icecream ~/temp/i/like ~/temp
$ pwd
~/temp/i/like/icecream
$ popd
~/temp/i/like ~/temp
$ pwd
~/temp/i/like
$ popd
~/temp
$ pushd i/like/icecream
~/temp/i/like/icecream ~/temp
$ pushd
~/temp ~/temp/i/like/icecream
$ pwd
~/temp
$ pushd
~/temp/i/like/icecream ~/temp
$ pwd
~/temp/i/like/icecream
$
```

You Learned This

You're getting into programmer territory with these commands, but they're so handy I have to teach them to you. These commands let you temporarily go to a different directory and then come back, easily switching between the two.

The pushd command takes your current directory and "pushes" it into a list for later; then it *changes* to another directory. It's like saying, "Save where I am, then go here."

The popd command takes the last directory you pushed and "pops" it off, taking you back there.

Finally, on Unix pushd, if you run it by itself with no arguments, will switch between your current directory and the last one you pushed. It's an easy way to switch between two directories. *This does not work in PowerShell.*

Do More

- Use these commands to move around directories all over your computer.
- Remove the `i/like/icecream` directories and make your own, then move around in them.
- Explain to yourself the output that `pushd` and `popd` will print out for you. Notice how it works like a stack?
- You already know this, but remember that `mkdir -p` will make an entire path even if all the directories don't exist. That's what I did very first for this exercise.

Exercise 9: Make Empty Files (Touch, New-Item)

In this exercise, you learn how to make an empty file using the `touch` (`new-item` on Windows) command.

Do This

Exercise 9 Session

```
$ cd temp
$ touch iamcool.txt
$ ls
iamcool.txt
$
```

You Learned This

You learned how to make an empty file. On Unix, `touch` does this, and it also changes the times on the file. I rarely use it for anything other than making empty files. On Windows, you don't have this command, so you learned how to use the `New-Item` command, which does the same thing but can also make new directories.

Do More

- **Unix.** Make a directory, change to it, and then make a file in it. Then change one level up and run the `rmdir` command in this directory. You *should* get an error. Try to understand why you got this error.
- **Windows.** Do the same thing, but you won't get an error. You'll get a prompt asking if you really want to remove the directory.

Exercise 10: Copy a File (cp)

In this exercise, you learn how to copy a file from one location to another with the `cp` command.

Do This

Exercise 10 Session

```
$ cd temp
$ cp iamcool.txt neat.txt
$ ls
iamcool.txt neat.txt
$ cp neat.txt awesome.txt
$ ls
awesome.txt iamcool.txt neat.txt
$ cp awesome.txt thefourthfile.txt
$ ls
awesome.txt  iamcool.txt  neat.txt  thefourthfile.txt
$ mkdir something
$ cp awesome.txt something/
$ ls
awesome.txt iamcool.txt  neat.txt  something  thefourthfile.txt
$ ls something/
awesome.txt
$ cp -r something newplace
$ ls newplace/
awesome.txt
$
```

You Learned This

Now you can copy files. It's simple to just take a file and copy it to a new one. In this exercise, I also make a new directory and copy a file into that directory.

I'm going to tell you a secret about programmers and system administrators now. They are lazy. I'm lazy. My friends are lazy. That's why we use computers. We like to make computers do boring things for us. In the exercises so far, you have been typing repetitive boring commands so that you can learn them, but usually it's not like this. Usually if you find yourself doing something boring

and repetitive, there's probably a programmer who has figured out how to make it easier. You just don't know about it.

The other thing about programmers is they aren't nearly as clever as you think. If you over think what to type, then you'll probably get it wrong. Instead, try to imagine what the name of a command is to you and try it. Chances are that it's a name or some abbreviation similar to what you thought it was. If you still can't figure it out intuitively, then ask around and search online. Hopefully it's not something really stupid like ROBOCOPY.

Do More

- Use the `cp -r` command to copy more directories with files in them.
- Copy a file to your home directory or desktop.
- Find these files in your graphical user interface and open them in a text editor.
- Notice how sometimes I put a / (slash) at the end of a directory? That makes sure the file is really a directory, so if the directory doesn't exist, I'll get an error.

Exercise 11: Move a File (mv)

In this exercise, you learn how to move a file from one location to another, using the `mv` command.

Do This

Exercise 11 Session

```
$ cd temp
$ mv awesome.txt uncool.txt
$ ls
newplace     uncool.txt
$ mv newplace oldplace
$ ls
oldplace     uncool.txt
$ mv oldplace newplace
$ ls
newplace     uncool.txt
$
```

You Learned This

Moving files or, rather, renaming them. It's easy: give the old name and the new name.

Do More

- Move a file in the `newplace` directory to another directory and then move it back.

Exercise 12: View a File (less, MORE)

To do this exercise, you're going to do some work using the commands you know so far. You'll also need a text editor that can make plain text (.txt) files. Here's what you do:

- Open your text editor and type some stuff into a new file. On OSX, this could be TextWrangler. On Windows, this might be Notepad++. On Linux, this could be gedit. Any editor will work.
- Save that file to your desktop and name it `test.txt`.
- In your shell, use the commands you know to *copy* this file to your `temp` directory that you've been working with.

Once you've done that, complete this exercise.

Do This

Exercise 12 Session

```
$ less test.txt
[displays file here]
$
```

That's it. To get out of `less`, just type `q` (as in quit).

Exercise 12 Windows Session

```
> more test.txt
[displays file here]
>
```

NOTE: In the above output, I'm showing `[displays file here]` to "abbreviate" what that program shows. I'll do this when I mean to say, "Showing you the output of this program is too complex, so just insert what you see on your computer here and pretend I did show it to you." Your screen will not actually show this.

You Learned This

This is one way to look at the contents of a file. It's useful, because if the file has many lines, it will "page" so that only one screenful at a time is visible. In the Do More section, you'll play with this some more.

Do More

- Open your text file again and repeatedly copy-paste the text so that it's about 50–100 lines long.
- Copy it to your `temp` directory again so you can look at it.
- Now do the exercise again, but this time page through it. On Unix, you use the spacebar and `w` (the letter w) to go down and up. Arrow keys also work. On Windows, just hit the spacebar to page through.
- Look at some of the empty files you created too.
- The `cp` command will overwrite files that already exist so be careful copying files around.

Exercise 13: Stream a File (cat)

You're going to do some more setup for this one so you get used to making files in one program and then accessing them from the command line. With the same text editor from the last exercise, create another file named `test2.txt`, but this time save it directly to your `temp` directory.

Do This

Exercise 13 Session

```
$ less test2.txt
[displays file here]
$ cat test2.txt
I am a fun guy.
Don't you know why?
Because I make poems,
```

```
that make babies cry.
$ cat test.txt
Hi there this is cool.
$
```

Remember that when I say `[displays file here]`, I'm abbreviating the output of that command so I don't have to show you exactly everything.

You Learned This

Do you like my poem? Totally going to win a Nobel. Anyway, you already know the first command, and I'm just having you check that your file is there. Then you `cat` the file to the screen. This command just spews the whole file to the screen with no paging or stopping. To demonstrate that, I have you do this to the `test.txt`, which should just spew a bunch of lines from that exercise.

Do More

- Make a few more text files and work with `cat`.
- Unix: Try `cat test.txt test2.txt` and see what it does.
- Windows: Try `cat test.txt,test2.txt` and see what it does.

Exercise 14: Remove a File (rm)

In this exercise, you learn how to remove (delete) a file using the `rm` command.

Do This

Exercise 14 Session

```
$ cd temp
```

```
$ ls
uncool.txt  iamcool.txt  neat.txt  something  thefourthfile.txt
$ rm uncool.txt
$ ls
iamcool.txt  neat.txt  something  thefourthfile.txt
$ rm iamcool.txt neat.txt thefourthfile.txt
$ ls
something
$ cp -r something newplace
$
$ rm something/awesome.txt
$ rmdir something
$ rm -rf newplace
$ ls
$
```

You Learned This

Here we clean up the files from the last exercise. Remember when I had you try to `rmdir` on a directory with something in it? Well, that failed because you can't remove a directory with files in it. To do that, you have to remove the file or recursively delete all its contents. That's what you did at the end of this.

Do More

- Clean up everything in `temp` from all the exercises so far.
- Write in your notebook to be careful when running recursive remove on files.

Exercise 15: Exit Your Terminal (exit)

Do This

Exercise 23 Session

```
$ exit
```

Exercise 23 Windows Session

```
> exit
```

You Learned This

Your final exercise is how to exit your Terminal. Again, this is very easy, but I'm going to have you do more.

Do More

For your last set of exercises, I'm going to have you use the help system to look up a set of commands you should research and learn how to use on your own.

Here's the list for Unix:

- xargs
- sudo
- chmod
- chown

Find out what these are, play with them, and then add them to your index cards.

Command Line Next Steps

You have completed the crash course. At this point, you should be a barely capable shell user. There's a whole huge list of tricks and key sequences you don't know yet, and I'm going to give you a few final places to go research more.

Unix Bash References

The shell you've been using is called Bash. It's not the greatest shell but it's everywhere and has a lot of features, so it's a good start. Here's a short list of links about Bash you should go read:

Bash Cheat Sheet http://cli.learncodethehardway.org/bash_cheat_sheet.pdf created by Raphael (http://freeworld.posterous.com/65140847) and CC licensed.

Reference Manual http://www.gnu.org/software/bash/manual/bashref.html