The Command Line Crash Course

Controlling Your Computer From The Terminal

Zed A. Shaw

Dec 2011

Contents

Preface

I wrote this book really quickly as a way to bootstrap students for my other books. Many students don't know how to use the basics of the command line interface, and it was getting in the way of their learning. This book is designed to be something they can complete in about a day to a week and then get enough skill at the command line to graduate to other books.

This book isn't a book about master wizardry system administration. It's just a quick introduction to get newbies going.

Introduction: Shut Up And Shell

This book 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 book, 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 of 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 baby little 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.

0.1 How To Use This Book

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

1. Get yourself a small paper notebook and a pen.
2. Start at the beginning of the book and do each exercise exactly as you're told.
3. 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.
4. 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@learncodethehard 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.

0.2 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:

1. Tell yourself you *will* do it. Don't try to find tricks or easy ways out of it, just sit down and do it.
2. 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.
3. 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.
4. Before you go to bed, drill just the cards you got wrong for about 5 minutes, then go to sleep.

There's 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.

0.3 License

0.4 Thanks

Thanks to Lauren Buchsbaum for editing this book and providing me with feedback. Also thanks to the many students who read the book and provided feedback.

Chapter 1

The Setup

In this book you will be instructed to do three things:

1. Do some things in your shell (command line, Terminal, PowerShell).
2. Learn about what you just did.
3. 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 book.

1.1 Do This

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

1.1.1 Mac OSX

For Mac OSX you'll need to do this:

1. Hold down COMMAND and hit the spacebar.
2. In the top right the blue "search bar" will pop up.
3. Type: terminal
4. Click on the Terminal application that looks kind of like a black box.
5. This will open Terminal.
6. 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.

1.1.2 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".

1.1.3 Windows

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

1. Click Start.
2. In "Search programs and files" type: powershell
3. Hit Enter.

If you don't have Windows 7, you should *seriously* consider upgrading. If you still insist on not upgrading then you can try installing it from the download center. You are on your own, though, since I don't have Windows XP, but hopefully the PowerShell experience is the same.

1.2 You Learned This

You learned how to get your terminal open so you can do the rest of this book.

Note 1 *Avoid The Hackers and Their zsh*

If you have that really smart friend who already knows Linux, ignore them when they tell you to use something other than bash. I'm teaching you bash. That's it. They will claim that zsh will give you 30 more IQ points and win you millions in the stock market. Ignore them. 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.

1.3 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 book by doing some memorization. Just trust me, this will make things silky smooth later on.

1.3.1 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 you do this book for just 15 minutes or so.

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!

chmod change permission modifiers

chown change ownership

1.3.2 Windows

If you're using Windows then here's your list of commands:

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

robocopy robust copy

mv move a file or directory

more page through a file

type print the whole file

forfiles run a command on lots of files

dir -r find files

select-string find things inside files

help read a manual page

helpctr find what man page is appropriate

echo print some arguments

set export/set a new environment variable

exit exit the shell

runas DANGER! become super user root DANGER!

attrib change permission modifiers

iCACLS change ownership

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.

Chapter 2

Paths, Folders, Directories (pwd)

2.1 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:

1. You do *not* type in the `$` (unix) or `>` (Windows). That's just me showing you my session so you can see what I got.
2. You type in the stuff after `$ or >`, then hit enter. So if I have `$ pwd` you type just `pwd` and hit enter.
3. 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:

Linux/Mac OSX Exercise 2

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

Windows Exercise 2

```
PS C:\Users\zed> pwd

Path
----
C:\Users\zed

PS C:\Users\zed>
```

2.2 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 pattern of:

1. There's a prompt.

Note 2 *The Windows vs. Unix Prompt*

In this book I need to save space so that you can focus on the important details of the commands. To do this, I'm going to strip out the first part of the prompt (the `PS C:\Users\zed` above) and leave just the little > part. This means your prompt won't look exactly the same, but don't worry about that.

Remember that from now on I'll only have the > to tell you that's the prompt.

I'm doing the same thing for the Unix prompts, but Unix prompts are so varied that most people get used to `$` meaning "just the prompt".

2. You type a command there. In this case, it's pwd.
3. It printed something.
4. Repeat.

You just learned what *`pwd`* does, which means "print working directory." What's a directory? It's a folder. Folder and directory are 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.

2.3 Do More

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

Chapter 3

What's Your Computer's Name? (hostname)

3.1 Do This

Linux/Mac OSX Exercise 3

```
$ hostname
Zeds-MacBook-Pro.local
$
```

Windows Exercise 3

```
> hostname
zed-PC
>
```

3.2 You Learned This

This is the name of your computer, or at least one of its names. Your name will probably be different than mine, and it could be just about anything.

3.3 Do More

1. Just like the last exercise, type this command 20 times and say it out loud.

Part I

Directories

Chapter 4

Make A Directory (mkdir)

4.1 Do This

Linux/Mac OSX Exercise 4

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

Windows Exercise 4

```
> mkdir temp


    Directory: C:\Users\zed


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/17/2011   9:02 AM            temp


> mkdir temp/stuff


    Directory: C:\Users\zed\temp


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/17/2011   9:02 AM            stuff


> mkdir temp/stuff/things
```

```
    Directory: C:\Users\zed\temp\stuff

Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/17/2011   9:03 AM            things


> mkdir temp/stuff/things/frank/joe/alex/john


    Directory: C:\Users\zed\temp\stuff\things\frank\joe\alex


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/17/2011   9:03 AM            john


>
```

4.2 You Learned This

Now we get into typing more than one command. These are all the different ways you can run mkdir. What's 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 3 *Windows, Slashes And Backslashes*

In this book 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) characters and other Windows users will typically expect those at times.

4.3 Do More

1. 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.
2. Make 20 other directories inside the temp directory in various levels. Go look at them with a graphical file browser.
3. Make a directory with a space in the name by putting quotes around it: `mkdir "I Have Fun"`
4. 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 **`Desktop`** is a good place.

Chapter 5

Change Directory (cd)

5.1 Do This

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

1. You do *not* type in the `$` (unix) or `>` (Windows).
2. You type in the stuff after this, then hit enter. If I have `$ cd temp` you just type `cd temp` and hit enter.
3. The output comes after you hit enter, followed by another `$` or `>` prompt.

Linux/Mac OSX Exercise 5

```
$ 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
~/
$
```

Windows Exercise 5

```
> cd temp
> pwd

Path
----
C:\Users\zed\temp


> cd stuff
> pwd

Path
----
C:\Users\zed\temp\stuff


> cd things
> pwd

Path
----
C:\Users\zed\temp\stuff\things


> cd frank
> pwd

Path
----
C:\Users\zed\temp\stuff\things\frank


> cd joe
> pwd

Path
----
C:\Users\zed\temp\stuff\things\frank\joe


> cd alex
> pwd

Path
----
C:\Users\zed\temp\stuff\things\frank\joe\alex


> cd john
> pwd

Path
----
C:\Users\zed\temp\stuff\things\frank\joe\alex\john


> cd ..
> cd ..
> cd ..
> pwd

Path
----
C:\Users\zed\temp\stuff\things\frank


> cd ../..
> pwd

Path
----
C:\Users\zed\temp\stuff


> cd ..
> cd ..
> cd temp/stuff/things/frank/joe/alex/john
> cd ../../../../../../../
> pwd

Path
----
C:\Users\zed


>
```

5.2 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.

5.3 Do More

A very important part of learning to the use 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, and then going to them with your CLI. This is what you'll do next.

1. cd to the **`joe`** directory with one command.
2. cd back to **`temp`** with one command, but not further above that.
3. Find out how to cd to your "home directory" with one command.
4. cd to your Documents directory, then find it with your GUI file browser (Finder, Windows Explorer, etc.).
5. cd to your Downloads directory, then find it with your file browser.
6. Find another directory with your file browser, then cd to it.
7. 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"`

Chapter 6

List Directory (ls)

6.1 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.

Linux/Mac OSX Exercise 6

```
$ 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
$
```

Windows Exercise 6

```
> cd temp
> ls


    Directory: C:\Users\zed\temp


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/17/2011   9:03 AM            stuff


> cd stuff
> ls


    Directory: C:\Users\zed\temp\stuff


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/17/2011   9:03 AM            things


> cd things
> ls


    Directory: C:\Users\zed\temp\stuff\things


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/17/2011   9:03 AM            frank


> cd frank
> ls


    Directory: C:\Users\zed\temp\stuff\things\frank


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/17/2011   9:03 AM            joe


> cd joe
> ls


    Directory: C:\Users\zed\temp\stuff\things\frank\joe
```

```
Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/17/2011   9:03 AM            alex


> cd alex
> ls


    Directory: C:\Users\zed\temp\stuff\things\frank\joe\alex


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/17/2011   9:03 AM            john


> cd john
> ls
> cd ..
> ls


    Directory: C:\Users\zed\temp\stuff\things\frank\joe\alex


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/17/2011   9:03 AM            john


> cd ..
> ls


    Directory: C:\Users\zed\temp\stuff\things\frank\joe


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/17/2011   9:03 AM            alex


> cd ../../..
> ls


    Directory: C:\Users\zed\temp\stuff


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/17/2011   9:03 AM            things


> cd ..
> ls


    Directory: C:\Users\zed\temp


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/17/2011   9:03 AM            stuff


>
```

6.2 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.

6.3 Do More

1. *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.
2. On Unix, try the `ls -lR` command while you're in **`temp`**.
3. On Windows do the same thing with `dir -R`.
4. Use `cd` to get to other directories on your computer then use `ls` to see what's in them.
5. Update your notebook with new questions. I know you probably have some, because I'm not covering everything about this command.
6. 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`.

Chapter 7

Remove Directory (rmdir)

7.1 Do This

Linux/Mac OSX Exercise 7

```
$ 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
$
```

Windows Exercise 7

```
> cd temp
> ls
```

```
    Directory: C:\Users\zed\temp


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/17/2011   9:03 AM            stuff


> cd stuff/things/frank/joe/alex/john/
> cd ..
> rmdir john
> cd ..
> rmdir alex
> cd ..
> rmdir joe
> cd ..
> rmdir frank
> cd ..
> ls


    Directory: C:\Users\zed\temp\stuff


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/17/2011   9:14 AM            things


> rmdir things
> cd ..
> ls


    Directory: C:\Users\zed\temp


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/17/2011   9:14 AM            stuff


> rmdir stuff
> pwd

Path
----
C:\Users\zed\temp


> cd ..
>
```

7.2 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.

7.3 Do More

1. Make 20 more directories and remove them all.
2. Make a single path of directories that is 10 deep and remove them one at a time just like I did above.
3. 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.

Chapter 8

Moving Around (pushd, popd)

8.1 Do This

Linux/Mac OSX Exercise 8

```
$ 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
$
```

Windows Exercise 8

```
> cd temp
> mkdir -p i/like/icecream


    Directory: C:\Users\zed\temp\i\like


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/20/2011  11:05 AM            icecream


> pushd i/like/icecream
> popd
> pwd

Path
----
C:\Users\zed\temp


> pushd i/like
> pwd

Path
----
C:\Users\zed\temp\i\like


> pushd icecream
> pwd

Path
----
C:\Users\zed\temp\i\like\icecream


> popd
> pwd

Path
----
C:\Users\zed\temp\i\like


> popd
>
```

8.2 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.*

8.3 Do More

1. Use these commands to move around directories all over your computer.
2. Remove the **`i/like/icecream`** directories and make your own, then move around in them.
3. Explain to yourself the output that `pushd` and `popd` print out to you. Notice how it works like a stack?
4. 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.

Part II

Files

Chapter 9

Making Empty Files (Touch, New-Item)

9.1 Do This

Linux/Mac OSX Exercise 9

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

Windows Exercise 9

```
> cd temp
> New-Item iamcool.txt -type file
> ls


    Directory: C:\Users\zed\temp


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---        12/17/2011   9:03 AM            iamcool.txt


>
```

9.2 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.

9.3 Do More

1. 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.
2. 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.

Chapter 10

Copy A File (cp)

10.1 Do This

Linux/Mac OSX Exercise 10

```
$ 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.tx
$ mkdir something
$ cp awesome.txt something/
$ ls
awesome.txt                 iamcool.txt                 neat.txt                    something
$ ls something/
awesome.txt
$ cp -r something newplace
$ ls newplace/
awesome.txt
$
```

Windows Exercise 10

```
> cd temp
> cp iamcool.txt neat.txt
> ls


    Directory: C:\Users\zed\temp


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
```

```
-a---        12/22/2011   4:49 PM          0 iamcool.txt
-a---        12/22/2011   4:49 PM          0 neat.txt


> cp neat.txt awesome.txt
> ls


    Directory: C:\Users\zed\temp


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---        12/22/2011   4:49 PM          0 awesome.txt
-a---        12/22/2011   4:49 PM          0 iamcool.txt
-a---        12/22/2011   4:49 PM          0 neat.txt


> cp awesome.txt thefourthfile.txt
> ls


    Directory: C:\Users\zed\temp


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---        12/22/2011   4:49 PM          0 awesome.txt
-a---        12/22/2011   4:49 PM          0 iamcool.txt
-a---        12/22/2011   4:49 PM          0 neat.txt
-a---        12/22/2011   4:49 PM          0 thefourthfile.txt


> mkdir something


    Directory: C:\Users\zed\temp


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/22/2011   4:52 PM            something


> cp awesome.txt something/
> ls


    Directory: C:\Users\zed\temp


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/22/2011   4:52 PM            something
-a---        12/22/2011   4:49 PM          0 awesome.txt
-a---        12/22/2011   4:49 PM          0 iamcool.txt
```

```
-a---        12/22/2011   4:49 PM          0 neat.txt
-a---        12/22/2011   4:49 PM          0 thefourthfile.txt


> ls something


    Directory: C:\Users\zed\temp\something


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---        12/22/2011   4:49 PM          0 awesome.txt


> cp -recurse something newplace
> ls newplace


    Directory: C:\Users\zed\temp\newplace


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---        12/22/2011   4:49 PM          0 awesome.txt

>
```

10.2 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 overthink 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.

10.3 Do More

1. Use the `cp -r` command to copy more directories with files in them.
2. Copy a file to your home directory or desktop.
3. Find these files in your graphical user interface and open them in a text editor.
4. 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.

Chapter 11

Moving A File (mv)

11.1 Do This

Linux/Mac OSX Exercise 11

```
$ 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
$
```

Windows Exercise 11

```
> cd temp
> mv awesome.txt uncool.txt
> ls


    Directory: C:\Users\zed\temp


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/22/2011   4:52 PM            newplace
d----        12/22/2011   4:52 PM            something
-a---        12/22/2011   4:49 PM          0 iamcool.txt
-a---        12/22/2011   4:49 PM          0 neat.txt
-a---        12/22/2011   4:49 PM          0 thefourthfile.txt
-a---        12/22/2011   4:49 PM          0 uncool.txt


> mv newplace oldplace
> ls


    Directory: C:\Users\zed\temp


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/22/2011   4:52 PM            oldplace
d----        12/22/2011   4:52 PM            something
-a---        12/22/2011   4:49 PM          0 iamcool.txt
-a---        12/22/2011   4:49 PM          0 neat.txt
-a---        12/22/2011   4:49 PM          0 thefourthfile.txt
-a---        12/22/2011   4:49 PM          0 uncool.txt


> mv oldplace newplace
> ls newplace


    Directory: C:\Users\zed\temp\newplace


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---        12/22/2011   4:49 PM          0 awesome.txt


> ls


    Directory: C:\Users\zed\temp


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/22/2011   4:52 PM            newplace
d----        12/22/2011   4:52 PM            something
-a---        12/22/2011   4:49 PM          0 iamcool.txt
-a---        12/22/2011   4:49 PM          0 neat.txt
-a---        12/22/2011   4:49 PM          0 thefourthfile.txt
-a---        12/22/2011   4:49 PM          0 uncool.txt


>
```

11.2 You Learned This

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

11.3 Do More

1. Move a file in the **`newplace`** directory to another directory then move it back.

Chapter 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:

1. 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.
2. Save that file to your Desktop and name it **`ex12.txt`**.
3. 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.

12.1 Do This

Linux/Mac OSX Exercise 12

```
$ less ex12.txt
[displays file here]
$
```

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

Windows Exercise 12

```
> more ex12.txt
[displays file here]
>
```

Note 4 *Abbreviating "displays file here"*

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.

12.2 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.

12.3 Do More

1. Open your text file again and repeatedly copy-paste the text so that it's about 50-100 lines long.
2. Copy it to your **`temp`** directory again so you can look at it.
3. 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 spacebar to page through.
4. Look at some of the empty files you created too.
5. The *`cp`* command will overwrite files that already exist so be careful copying files around.

Chapter 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 Exercise 12, create another file named **`ex13.txt`** but this time save it directly to your **`temp`** directory.

13.1 Do This

Linux/Mac OSX Exercise 13

```
$ less ex13.txt
[displays file here]
$ cat ex13.txt
I am a fun guy.
Don't you know why?
Because I make poems,
that make babies cry.
$ cat ex12.txt
Hi there this is cool.
$
```

Windows Exercise 13

```
> more ex13.txt
[displays file here]
> cat ex13.txt
I am a fun guy.
Don't you know why?
Because I make poems,
that make babies cry.
> cat ex12.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.

13.2 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 **`ex12.txt`** which should just spew a bunch of lines from that exercise.

13.3 Do More

1. Make a few more text files and work with *`cat`*.
2. Unix: Try `cat ex12.txt ex13.txt` and see what it does.
3. Windows: Try `cat ex12.txt,ex13.txt` and see what it does.

Chapter 14

Removing A File (rm)

14.1 Do This

Linux/Mac OSX Exercise 14

```
$ cd temp
$ ls
uncool.txt                  iamcool.txt                 neat.txt                  something
$ 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
$
```

Windows Exercise 14

```
> cd temp
> ls


    Directory: C:\Users\zed\temp


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/22/2011   4:52 PM            newplace
d----        12/22/2011   4:52 PM            something
-a---        12/22/2011   4:49 PM          0 iamcool.txt
-a---        12/22/2011   4:49 PM          0 neat.txt
-a---        12/22/2011   4:49 PM          0 thefourthfile.txt
```

```
-a---        12/22/2011   4:49 PM          0 uncool.txt


> rm uncool.txt
> ls


    Directory: C:\Users\zed\temp


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/22/2011   4:52 PM            newplace
d----        12/22/2011   4:52 PM            something
-a---        12/22/2011   4:49 PM          0 iamcool.txt
-a---        12/22/2011   4:49 PM          0 neat.txt
-a---        12/22/2011   4:49 PM          0 thefourthfile.txt


> rm iamcool.txt
> rm neat.txt
> rm thefourthfile.txt
> ls


    Directory: C:\Users\zed\temp


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        12/22/2011   4:52 PM            newplace
d----        12/22/2011   4:52 PM            something


> cp -r something newplace
> rm something/awesome.txt
> rmdir something
> rm -r newplace
> ls
>
```

14.2 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 of its contents. That's what you did at the end of this.

14.3 Do More

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

Chapter 15

Pipes And Redirection

15.1 Do This

Linux/Mac OSX Exercise 15

```
$ cat ex12.txt ex13.txt | less
$ cat < ex13.txt
I am a fun guy.
Don't you know why?
Because I make poems,
that make babies cry.
$ less < ex12.txt
$ less < ex12.txt | cat | less
$ cat ex13.txt > ex15.txt
$ cat ex15.txt
I am a fun guy.
Don't you know why?
Because I make poems,
that make babies cry.
$
```

Windows Exercise 15

```
> cd ..
> cd temp
> cat ex12.txt,ex13.txt | more
> echo "I am a new file." > ex15.txt
> cat ex15.txt
I am a new file.
> cat ex15.txt > another.txt
> cat another.txt
I am a new file.
> cat ex15.txt | more
>
```

15.2 You Learned This

Now we get to the cool part of the command line: redirection. The concept is that you can take a command and you can change where its input and output goes. You use the `<` (less-than), `>` (greater-than), and `|` (pipe) symbols to do this. Here's a breakdown:

| The `|` takes the output from the command on the left, and "pipes" it to the command on the right. In line 1 you see me do that.

< The `<` will take and send the input from the file on the right to the program on the left. You see me do that in line 2. *This does not work in PowerShell.*

> The `>` takes the output of the command on the left, then writes it to the file on the right. You see me do that on line 9.

>> The `>>` takes the output of the command on the left, then *appends* it to the file on the right.

There are a few more symbols, but we'll start with just these for now.

15.3 Do More

1. Create some more index cards for memorizing these three symbols. Write the symbol on one side, then what it does on the other side. Drill these just like the other commands.

Chapter 16

Wildcard Matching

16.1 Do This

Linux/Mac OSX Exercise 16

```
$ cd temp
$ ls *.txt
ex12.txt         ex13.txt         ex14.txt         uncool.txt
$ ls ex*.*
ex12.txt         ex13.txt         ex14.txt
$ ls e*
ex12.txt         ex13.txt         ex14.txt
$ ls *t
ex12.txt         ex13.txt         ex14.txt         uncool.txt
$ cat *.txt > bigfile.txt
$ rm *.txt
$ ls
$
```

Windows Exercise 16

```
> cd temp
> ls *.txt


    Directory: C:\Users\zed\temp


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---        12/22/2011   5:23 PM         38 another.txt
-a---        12/22/2011   5:23 PM         38 ex15.txt


> ls ex*.*


    Directory: C:\Users\zed\temp
```

```


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---         12/22/2011   5:23 PM         38 ex15.txt


> ls e*


    Directory: C:\Users\zed\temp


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---         12/22/2011   5:23 PM         38 ex15.txt


> ls *t


    Directory: C:\Users\zed\temp


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---         12/22/2011   5:23 PM         38 another.txt
-a---         12/22/2011   5:23 PM         38 ex15.txt


> cat *.txt
I am a new file.
I am a new file.
> rm *.txt
> ls
>
```

16.2 You Learned This

Sometimes you want to do a command to a set of files all at once. The way you do this is to use the `*` (asterisk) symbol to say "anything." Wherever you put the asterisk, the shell will build a list of all the files that match the non-asterisk part.

In this exercise you list out various files that you've made so far. I have a few extra that were hanging out in my directory, and you might have others. The key is that, by writing `*.txt`, you are saying "anything ending in .txt".

At the end we use `rm *.txt` to remove all of the .txt files in the **`temp`** directory.

16.3 Do More

1. Add the `*` to your deck of flash cards. On the back write: "matches anything in a wildcard like *.txt".

Part III

Searching

Chapter 17

Finding Files (find, DIR -R)

17.1 Do This

This exercise is going to combine three concepts into one single command. I'm going to show you how to find all your text files and page through them.

Linux/Mac OSX Exercise 17

```
$ cd temp
$ find . -name "*.txt" -print
$ cd ..
$ find . -name "*.txt" -print | less
$ cd ..
$ find . -name "*.txt" -print | less
$
```

Windows Exercise 17

```
> dir -r
> cd ..
> dir -r -filter "*.txt"
> cd ..
> dir -r | more
>
```

17.2 You Learned This

You just learned how to run the *find* (*dir -r* on Windows) command to search for all files ending in **.txt** and then look at the results with *less* (*more* on Windows).

Here's a breakdown of exactly what this does for Unix:

1. First I go to the **temp** directory.
2. Then I just do a plain find from there since there's not many **.txt** files. How "find" works is you write in a kind of sentence: "Hey *find*, start here (.) then find files named "*.txt" and print them". Thinking about commands as if you're telling the computer to do something is a good way to remember it.

3. Next I go up one directory and then I do the same command, but this time I pipe the output to *`less`*. This command could run for a while, so be patient.
4. I then do this again, but this one could take a really long time, so feel free to just hit CTRL-c to abort it.

On Windows it's nearly the same:

1. I just do a plain dir from wherever you are since there's not many **`.txt`** files. This command says "do a directory listing of this directory and all the others under it", and is called a "recursive" command.
2. Next I go up one directory and then I do the same command, but this time I pipe the output to *`more`*. This command could run for a while, so be patient.
3. I then do this again, but this one could take a really long time, so feel free to just hit CTRL-c to abort it.

17.3 Do More

1. Unix: Get your *`find`* index card and add this to the description side: "find STARTDIR -name WILDCARD -print". Next time you drill make sure you can say that phrase so you remember how find is formatted.
2. Windows: Do the same as above, but write "dir -r -filter"
3. You can put any directory where the `.` (dot) is. Try another directory to start your search there.
4. Look for all the video files on your computer starting at the home drive and use the `>` to save the list to a file. Remember how you can do `SOMECOMMAND > SOMEFILE.txt` and it will write the output of SOMECOMMAND to the file SOMEFILE.txt?

Chapter 18

Looking Inside Files (grep, select-string)

18.1 Do This

I'm showing you a little trick for typing text into a file very quickly. If you do `cat > somefile.txt` then *cat* will read whatever you type and then write it to that file. *On Windows* it's done with `echo > somefile.txt`. The important thing, though, is that you have to "close" the file by typing CTRL-d.

If you can't figure this out just use a text editor to make the **`newfile.txt`** and **`oldfile.txt`**.

Linux/Mac OSX Exercise 18

```
$ cd temp
$ cat > newfile.txt
This is a new file.
This is a new file.
This is a new file.
$ cat > oldfile.txt
This is a old file.
This is a old file.
This is a old file.
$ grep new *.txt
newfile.txt:This is a new file.
newfile.txt:This is a new file.
newfile.txt:This is a new file.
$ grep old *.txt
oldfile.txt:This is a old file.
oldfile.txt:This is a old file.
oldfile.txt:This is a old file.
$ grep file *.txt
newfile.txt:This is a new file.
newfile.txt:This is a new file.
newfile.txt:This is a new file.
oldfile.txt:This is a old file.
oldfile.txt:This is a old file.
oldfile.txt:This is a old file.
$
```

In Windows a similar trick is to do `echo > somefile.txt` and then you'll be prompted for each line. Give an empty line to stop entering text.

Windows Exercise 18

```
> cd temp
> echo > newfile.txt

cmdlet Write-Output at command pipeline position 1
Supply values for the following parameters:
InputObject[0]: This is a new file.
InputObject[1]: This is a new file.
InputObject[2]: This is a new file.
InputObject[3]:
> echo > oldfile.txt

cmdlet Write-Output at command pipeline position 1
Supply values for the following parameters:
InputObject[0]: This is a old file.
InputObject[1]: This is a old file.
InputObject[2]: This is a old file.
InputObject[3]:
> select-string new *.txt

newfile.txt:1:This is a new file.
newfile.txt:2:This is a new file.
newfile.txt:3:This is a new file.


> select-string old *.txt

oldfile.txt:1:This is a old file.
oldfile.txt:2:This is a old file.
oldfile.txt:3:This is a old file.


> select-string file *.txt

newfile.txt:1:This is a new file.
newfile.txt:2:This is a new file.
newfile.txt:3:This is a new file.
oldfile.txt:1:This is a old file.
oldfile.txt:2:This is a old file.
oldfile.txt:3:This is a old file.


>
```

18.2 You Learned This

You made two files that were almost the same, except one had "new" and the other had "old" in it. Then you searched for different words in those files. You can look for words, but you can also find whole sentences in files if you put them in quotes.

18.3 Do More

1. Use quotes to find "new file" and "old file" and "This is".
2. Take the list of videos you created (or any other list) and use it to find some videos you want to find.
3. Unix: You can use `-i` to ignore case with *grep*. Try `grep -i new *.txt`

Part IV

Help

Chapter 19

Getting Command Help (man, HELP)

19.1 Do This

Linux/Mac OSX Exercise 19

```
$ man find
$ man less
$ man man
$ man grep
$
```

Windows Exercise 19

```
> help dir
> help select-string
> help help
> help cp
>
```

19.2 You Learned This

You can use the *man* command on Unix, and the *help* command in Windows to find information about commands. I've been playing a very dirty trick on you this whole time. I could have just told you to do this and read about each command rather than memorize what they do. But, if I did this, you'd be lost because you wouldn't know the basic commands you have to know, or how directories work, or what a wildcard is, etc etc.

Hopefully you forgive me for not telling you about this awesome tool until now, but now when you forget what a command does, just use the help.

19.3 Do More

1. Use *man* or *help* to look at every one of the commands you have in your list to memorize.

Chapter 20

Finding Help (apropos, HELP)

20.1 Do This

Linux/Mac OSX Exercise 20

```
$ apropos search
$ apropos find
$ apropos remove
$ apropos directory
$
```

Windows Exercise 20

```
> help *Al*
> help *Dir*
> help *Find*
> help *remove*
```

20.2 You Learned This

Sometimes you forget the name of a command but you know what it does. This command looks through all the help files and finds potentially relevant help for you.

Honestly, I only use this out of desperation. I first go search online and can usually find a better page describing what I want to do. If I *still* can't figure out what the name of that command was then I go through this help list until I see it.

20.3 Do More

1. Use this to find all the commands in your list as well.

Part V

Sessions

Chapter 21

What's In Your Environment (env, echo, Env:)

21.1 Do This

Linux/Mac OSX Exercise 21

```
$ env
TERM_PROGRAM=Apple_Terminal
TERM=xterm
SHELL=/bin/bash
OLDPWD=/Users/zed/temp
USER=zed
COMMAND_MODE=unix2003
PATH=/opt/local/bin:/opt/local/sbin:/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin:/usr/X11/bin
PWD=/Users/zed/
LANG=en_US.UTF-8
PS1=$
SHLVL=1
HOME=/Users/zed
LOGNAME=zed
_=/usr/bin/env
$ env | grep zed
OLDPWD=/Users/zed//temp
USER=zed
PWD=/Users/zed/
HOME=/Users/zed
LOGNAME=zed
$ echo $USER
zed
$ echo $PWD
/Users/zed/
$ export TESTING="1 2 3"
$ echo $TESTING
1 2 3
$ env | grep TESTING
TESTING=1 2 3
$
```

Windows Exercise 21

```
> get-childitem Env:

Name                           Value
----                           -----
ALLUSERSPROFILE                C:\ProgramData
APPDATA                        C:\Users\zed\AppData\Roaming
...


> $env:PROMPT
$P$G
> $env:TEMP
C:\Users\zed\AppData\Local\Temp
> $env:OS
Windows_NT
>
```

21.2 You Learned This

Your shell has these "hidden variables" that can change how other programs work. In this exercise I first print out my environment, just dumping it to the screen. Then I do it again but pipe it through *grep* to find only the variables with my username in them. Finally, I set an environment variable TESTING to "1 2 3".

You may not run into this too often, but sometimes to get something configured you have to change these. A good example is the PATH variable, which determines the search order for other commands. Changing your PATH is going to be one of your exercises.

21.3 Do More

1. I want you to go online and research how you change your PATH for your computer. Try to do it entirely from the CLI.

Chapter 22

Changing Environment Variables (export, Env:)

22.1 Do This

Linux/Mac OSX Exercise 22

```
$ export TESTING="bada bada bing"
$ echo $TESTING
bada bada bing
$ unset TESTING
$ echo $TESTING

$ env | grep TESTING
$
```

Windows Exercise 22

```
> get-childitem Env:

Name                           Value
----                           -----
APPDATA                        C:\Users\zed\AppData\Roaming
COMPUTERNAME                   ZED-PC
...


> $env:TESTING = "bada bada bing"
> $env:TESTING
bada bada bing
> remove-item Env:\TESTING
> get-childitem Env: | select-string TESTING
>
```

22.2 You Learned This

You can also change an environment variable to something else, as well as unset it so it isn't in your environment at all. You can't set an environment variable to nothing ("") to remove it. You have to use a command.

22.3 Do More

1. Take and list out all the environment variables you've found and then go look up what they are online.
2. Read the man page for env again. What else can it do?

Chapter 23

Exiting Your Terminal (exit)

23.1 Do This

Linux/Mac OSX Exercise 23

```
$ exit
```

Windows Exercise 23

```
> exit
```

23.2 You Learned This

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

23.3 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:

1. xargs
2. sudo
3. chmod
4. chown

For Windows look up these things:

1. forfiles
2. runas
3. attrib

4. icacls

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

Part VI

Epilogue

Chapter 24

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.

24.1 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 and CC licensed.

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

24.2 PowerShell References

On Windows there's really only PowerShell. Here's a list of useful links for you related to PowerShell:

Owner's Manual http://technet.microsoft.com/en-us/library/ee221100.aspx

Cheat Sheet http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=7097

Master Powershell http://powershell.com/cs/blogs/ebook/default.aspx

24.3 Go Forth

From now on you shouldn't be afraid of using the command line. If you are aspiring to be a programmer, it is the best first step to understand how a computer is a "language machine." The shell is much like a tiny little programming language that most people can understand easily.

My advice for getting good at the the CLI is to force yourself to use it every day, no matter how painful it seems. Part of the "pain" is having to remember the commands without a visual cue. The advantage of a GUI is that you get a cue from the graphics to remind you of how to use the tool. With the CLI you have to dredge every command from nothing, which is irritating at first.

However, I have a trick for you to get over this pain. Create your own cheat sheet of CLI tricks you use all the time. Make yourself use the CLI to do things, and when you run into something you just *have* to use again, write it on your cheat sheet. The next time you need to do that, look at your cheat sheet and you'll remember.

Eventually you won't need the cheat sheet. In fact, I'd say most of my daily shell usage consists of 10 commands, most of which are in this little book. Memorizing 10 commands is really easy, so there's nothing stopping you.

If you run into trouble with this book, feel free to email me at help@learncodethehardway.org and I'll help out.