Crash Course

The Command Line

What is the command line?

The *command line interface* (CLI) is a user interface to a computer's operating system. The user interacts with a visual prompt by typing in *commands* and receiving a response from the system.

AKA: terminal (OSX/Linux), Command Prompt (Windows), shell, console.

Why use the command line?

- The command line gives you direct access to your computer's operating system.
- Users are able to communicate with the computer at a lower level, performing more powerful functions with a computer language.
- Gives the user more freedom to perform tasks, automate scripts, override restrictions compared to the traditional graphical user interface (GUI).

Quick Note

The Linux/OSX terminal and the Windows Command Prompt differ slightly because they are based on different operating systems. Windows users will see that some commands we cover differ for their operating system. I will try to provide the equivalent Windows commands whenever necessary, as well as any useful resources to help along the way.

Getting Started

Mac - Open the Terminal application

Windows - Open the Command Prompt application

NOTE the command line uses the $ (OSX/Linux) or > (Windows) to indicate it is ready to receive a command. In the examples, DO NOT type these characters along with the commands. Everything after the $ or > are the commands to type.

Navigating in the command line

- Launching the command line will open the shell in your home directory.
- A *directory* is the same as a folder. In the command line you can navigate to directories using commands, much like how you navigate to folders in the Finder application.

Directory and File Paths

The command line displays and interacts with directories/files with a directory or file *path*. A path represents the location of a file or directory.

```
/Users/yestevez/cisc2350/index.html
```

Tells us that *index.html* is located in the *cisc2350* directory, within the *yestevez* directory, within the *Users* directory.

The Root Directory

The *root* directory is the directory that contains all directories and files on your computer. The root directory is represented by the forward slash character (/). All paths begin with the root (/) because it holds everything in your computer.

```
/Users/yestevez/cisc2350/index.html
```

*** NOTE -** the root directory contains **your entire computer**. Unless you are experienced using the command line, I recommend you do not alter any files or unfamiliar directories within it.

**** Windows Users -** the root directory can also be represented by the backslash character (\)

pwd

print working directory

The `pwd` command stands for *print working directory*. It prints out to the console *the current directory you are in*.

```
$ pwd

/Users/yestevez
```

cd

change directory

The `cd` command stands for *change directory*. It accepts an *argument* after it that is the file you want to move to.

```
$ pwd

/Users/yestevez

$ cd /Users/yestevez/Documents

[The terminal navigates to the Documents directory]

$ pwd

/Users/yestevez/Documents
```

Special Characters

The command line has special characters that represent different paths

- `~` your home directory.
- `.` the current directory.
- `..` the parent directory.
- `-` the last directory you were in.

```
$ pwd

/Users/yestevez/cisc2350

$ cd ../

[The terminal navigates to the parent directory]

$ pwd

/Users/yestevez

$ cd -

[The terminal navigates to the last directory]

$ cd ~

[The terminal navigates to your home directory]

$ pwd

/Users/yestevez/
```

`ls`

list directory

The `ls` command stands for *list directory*. It displays the contents of a directory. `ls` can accept an argument to specify the directory to list. Without the argument, it lists the current working directory.

```
$ pwd

/Users/yestevez/cisc2350

$ ls

README.md  assignments/  docs/  samples/

$ ls assignments/

assignment1/
```

mkdir

make directory

The `mkdir` command stands for *make directory*. It creates an empty directory. Can accept an argument as the path you want to create your new directory in. With no arguments, creates in current working directory.

```
$ pwd

/Users/yestevez/cisc2350/assignments

$ ls

assignment1/

$ mkdir assignment2

$ ls

assignment1/  assignment2/
```

rm

remove

The `rm` command stands for *remove*. It deletes files or directories by accepting an argument as the file/directory to delete.

WARNING using the `rm` command **permanently deletes** the file/directory.

```
$ pwd

/Users/yestevez/cisc2350/assignments

$ ls

README.md   assignment1/   assignment2/

$ rm README.md

$ ls

[deletes README.md file]

assignment1/   assignment2/
```

Command options

- Command line *options* are additional arguments passed with a command to give the command line extra instructions on how to perform the command.
- Options are usually preceded by a dash character (-) followed by a letter or word that corresponds with the option

```
$ rm -rf README.md
```

Remove a directory

The `rm` command can be combined with options to remove directories

- `r` removes the file hierarchy
- `f` remove files without prompting

```
$ ls

assignment1/  assignment2/

$ rm -rf assignment2/

assignment1/  assignment2/

[deletes assignment2 directory]

$ ls

assignment1/
```

man Pages

man pages are command manuals that you can access from the command line. They provide information about what the command does, how to run it, and the arguments and options it can take.

```
$ man rm
```

Will display the manual for the rm command

You’re ready for the command line.

Helpful Resources

Command Line Crash Course (Mac/Linux & Windows)

Terminal Crash Course and Reference Sheet (Max/Linux)

A Command Line Crash Course (Mac/Linux & Windows)

Learn the Command Line (Interactive course)