Programming Bootcamp 2015
Getting acquainted with the terminal

Sarah Middleton

Last Updated: May 30, 2015

This lab is intended to help you feel more comfortable with the terminal/command line. I highly recommend you complete it before our first meeting, especially if you have never used the command line before. If you need more help with any of the material here, please direct your questions to the Piazza forum so everyone can contribute/benefit from the answers.

1 Using the terminal

What is the terminal?

The terminal, or command line, is a way of interacting with your computer using text-based commands rather than pointing and clicking with a mouse. You can think of it sort of like a text-only version of the graphical windows you normally use to navigate directories (folders) and files on your computer.

Just like when you use the graphical interface, you are always "in" some folder when you use the command line. This is called the "current directory". When you first open the terminal, your will be in your "home directory". This may be a strange system folder you've never seen before, but that's ok. Using certain commands, we can figure out which folder we're in, display the contents of the folder, as well as navigate to other folders. You can do most of the things you normally do when you use the graphical interface, such as open files, delete files, make new folders, move files around, run programs, and more. **We can also do many things with the terminal that we can not do with the graphical folder system–this is why it's so important to learn.** It will feel slow at first, but stick with it and it will soon feel more natural.

Anatomy of a command

A command will usually have three basic parts: the command itself, arguments, and options. A basic command might look something like this:

```
someCommand -o filename
```

`someCommand` - this is the basic command. It may or may not require further args/options.

`-o` - This is an option, which modifies the behavior of the command in some way. Options are usually optional. For example, if you use the `less` command to view the contents of a file, you can optionally put `less -S` to view the file without line-wrapping. Use the `man` command (short for "manual", see next section) to find out what options are available to each command. Note that the Windows terminal uses a forward slash (e.g. `/o`) to indicate options instead of a dash.

`filename` - This is what is sometimes called an "argument", or arg for short. An arg is just a piece of information, such as the name of a file, that may be optional or required for the command to work. Again, always refer to the `man` page if you're not sure what args are needed for a command.

Basic commands

Below is a list of basic terminal commands that we'll be using throughout the course. Unfortunately, commands differ somewhat between Windows and Unix (Mac/Linux) systems; in those cases you'll see two different commands listed. If you're on Windows and want a more Unix-like terminal experience (which, by almost all accounts, is vastly superior), see the section at the end on installing Cygwin.

Command (Unix)	Windows	Purpose	Examples
ls	dir	Display contents of current directory (files, folders)	ls
cd [dirname]		Change to a directory within the current directory	cd MyCode cd MyCode/research
cd ..		Go up/back one directory or multiple directories	cd .. cd ../../../MyCode
man [command]	help [command]	Manual. Display detailed info about a given command. Very useful!	man cd help dir man man
mkdir [dirname]		Make a new directory within the current one	mkdir NewFolder
pwd	cd	Prints the path of the current directory	pwd
less	more	View contents of a file one "page" at a time	less myfile.txt more myfile.txt

There are many more commands, but this is all we need for now.

Other useful tips

- Typing `cd` with no file name brings you back to your home folder from anywhere (Unix only).

- Tab auto-completion: if you start typing the name of a file/folder and then hit tab, the rest of the name will automatically be completed for you–but only it is in the current directory and there is no other file/folder in that directory with the same prefix that you have typed so far.

- To reuse a command you already entered previously, use the up and down arrow keys to cycle through your command history.

- When you use commands like `man` or `less`, your screen will be filled with text and you can use your arrow keys to scroll up and down (Unix only. In Windows use the enter key). Press `q` to go back to the command line (everything you had before will still be there).

Practice

1. First, open a terminal window:

   Windows: Go to Start > All programs > Accessories > Command Prompt.
   OS X: Go to Applications > Utilities > Terminal.

2. Your first goal is to figure out which directory on your computer you're currently in. This will be different depending on your operating system. In Windows 7, I see this next to my cursor:

   ```
   C:\Windows\System32>
   ```

   This makes things easy, since it tells me exactly where I am. In other terminal programs you may see something different. If you are on a Unix machine, try using `pwd`. Here's what I see when I do this on Cygwin (which emulates the Unix terminal on Windows):

   ```
   $ pwd
   /home/Sarah
   ```

   So in Cygwin I am in a directory called `Sarah`, which is within the directory called `home`.

3. Display the contents of your current directory using `dir` if you are on Windows, and `ls` otherwise. You will see a list of all files and folders. When I do this in Windows, I get a huge list of files that don't mean much to me, since I am in a system folder.

4. Your next goal is to figure out how to get to the directories on your computer where you normally store your files, e.g. My Documents. Since this will be different for everyone, I can't give you exact directions. You will need to use a combination of `cd` and `ls/dir`

to navigate around. Remember that you can use `cd ..` to go "up" a directory (i.e. move to the directory that contains the current directory).

Here is how I did it, though it will most likely be different for you:

```
C:\Windows\System32> cd ..
C:\Windows> cd ..
C:\> cd Users
C:\Users> cd Sarah
C:\Users\Sarah> cd Documents
C:\Users\Sarah\Documents>
```

It's important that you know how to do this before we start the course, since we'll need to access and run code that will most likely be stored in one of your document folders.

5. Now we are going to make a directory where you will store your code for this course. Navigate to a directory where you would like this new directory to be, then type:

   ```
   mkdir PythonBootcamp
   ```

   You can call it something else, if you want. Next, display the contents of your current directory using `ls/dir`. You should see the new directory you created. Navigate into the new directory, and then create another new one called `lab1`. You'll use this for the exercises for the first class.

6. Navigate to a folder where you already have some documents stored. Try viewing a file using `less` or `more`. You will notice that if you do this with non-plain text documents, such as PDFs and Word documents, you just get a bunch of strange characters. (If you get a message about the file being binary, press "y" to continue). We can only view plain text files in the terminal. If you have a plain text file (the extension will probably be .txt), try that. It should be readable.

   If you do not have any plain text files, go ahead and create one using a plain text editor (just copy and paste any old text in there, the more the better). Verify that you can view this in the terminal. Make sure you know how to scroll through the text and exit back to the command line (see Useful Tips above).

7. Try the following commands (use a directory with at least a few files in it):

Unix	Windows
ls	dir
ls -l	
ls -l -S	dir /O:S
ls -R	dir /S

   Note what changes when you use different options. Use the `man` command (`help` on Windows) to view the manual page for this command to verify what the different options do. Take note of any other options you think would be useful, and feel free to

try them out. I also suggest you look at the manual pages for `less/more`, as there are many useful options there.

2 Optional: Installing Cygwin [Windows Only]

The native Windows command line is unfortunately somewhat clunky compared to what you'll find on Unix systems. If you are on Windows and plan to use the command line at all beyond this course, you might consider installing Cygwin. This is a very popular terminal program which essentially emulates the type of terminal you'd find on a Mac or Linux machine. It has many more features and allows you to use most Unix commands.

(Please note that this is not required for this course. You can do just fine with the regular Windows terminal for the simple things we'll be doing here. Cygwin will make your life easier in the long run, though.)

You can download Cygwin here: `http://www.cygwin.com`. During installation, a window will pop up allowing you to pick extra packages to install. Expand the menu labeled "Python" and select (click the arrows) next to the package called "python: Python language interpreter". I don't think you'll need any other packages for this course. Click next to finish installation. When you run Cygwin, you will start in a special home folder in the Cygwin file system. There won't be anything in there.

To more easily access other files on your computer from this home folder, e.g. your "My Documents" folder, create a symbolic link (shortcut) to it like so:

```
ln -s ../../cygdrive/c/Users/Sarah/foldername LinkName
```

replacing everything after "Users" with whatever folders you want (you will need to figure out what they are called on your computer). Type `man ln` to learn more about creating symbolic links. As you can see here, you must go through `/cygdrive/c/` to get to the rest of the files on your C: drive.

3 Further reading

If you want to learn more about using the Unix terminal, I highly recommend the Unix/terminal introduction found here: `http://korflab.ucdavis.edu/Unix_and_Perl/unix_and_perl_v2.3.7.pdf`. It goes into a lot more detail and explains some of the nuances that I've glossed over, but is very readable and newbie-friendly. The Unix stuff starts on pg. 15.