ANCHORAGE PROGRAMMING WORKSHOP

Introduction to the Command Line

Navigate

See the white rectangle (or it may display as a blinking cursor, on some browsers)? That's the place where you enter commands, also known as the "command prompt."

In the directions below, when we say "Type **a command**," we want you to type the part in **bold** into the command prompt. Unless we add more information after the **bold** part, go ahead and hit "enter" after each command. There will be a few places where we have you wait, but not many.

- pwd

Type **pwd** into the terminal and hit **enter**. (The "enter" is going to be assumed after all of your commands, from this point.)

You should see output that looks something like */home/action*

This tells you where you are in the file structure. Linux, like Windows, organizes files into folders (also called *directories*) and subfolders (*subdirectories*). So if **pwd** returns */home/action*, you're in a directory called *action*, which is in a directory called *home*

- cd

Let's say you don't want to be in the */action* directory anymore. You want to go up a level and be in */home*.

Type **cd ..**

Type **pwd**

The output should be */home*

You've changed directories ("cd" - get it?), up a level (that's what ".." means). But what if you didn't really want to do that? What if you think, "No, I want to be in */home/action* now"? No problem!

Type **cd action**

Type **pwd**

Now, as you can see, you're back where you started, in */home/action*.

Make stuff

- mkdir

What if you aren't satisfied with the directories that are available to you? (Long-term, this is inevitable, right? You probably won't want to store all of your files in one place.) No problem!

Type **mkdir [your name]** — but don't literally type that; replace "[your name]" with your actual first name :)

Now change directory into the directory with your name. Scroll up for a reminder about changing directories, if you need it. Remember to use **pwd** to make sure you're in the right place.

- nano

It's time to make a text file! There are lots of text editors available—and lots of strong opinions about which is the best one—but for today, we're going to use a simple one. It'll work great for anything you want to do, from writing to-do lists to writing code.

Type **nano myfile** — (go ahead and hit "enter", but *don't panic* when the screen changes entirely!)

You're inside an application, on a text-only interface. (How cool is that?) As you can see, there are commands along the bottom. Where you see a caret (^), it means you hold down the ctrl key along with that letter to execute that command. But first let's enter some text, before we worry about any of the commands.

Type **whatever you like**. I typed "Hi, my name is Coral. I like birds and chinchillas and coffee." Pretty much anything will do.

Once you've entered some text, hold down the *ctrl key* and hit *X* (shortened from now on to *ctrl-x*)

A prompt appears at the bottom of the screen: *Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES)?*

Type *y*

The prompt now asks if this is the filename you want to write. It is, so

Hit *enter*

Congratulations! You just made a text file!

Look at stuff

- ls

How do you *know* you made a file, though? Not a problem. Let's have a look!

Type **ls**

The output should be *myfile*

You have just listed ("ls" - list) the contents of the current directory.

For fun, let's change directory up one level and look at what's inside that directory:

cd ..

ls

The output should be *[your name] README.md workspace* — the order might vary, based on where your name falls in the alphabet; that's fine

Now, change directory back into [your name], and make sure you're in the right place with **pwd**

One more thing: you can list the contents of directories other than the one you're in.

Type **ls ..**

The output should match what happened when you changed directory up a level and listed its contents, a moment ago, but you didn't have to change directory to see it! (Type **pwd** and confirm you're still in the same place, if you have doubts. :))

- more / less

  Let's say you want to know what's in *myfile* without opening it up in *nano*. That's totally doable.

  Type **more myfile**

  And, just to compare,

  Type **less myfile** — you can escape the screen that comes up by typing **q**

  It turns out, *less* is a bit more complex. For big files, *more* will let you go through it all in order, where *less*will let you scroll through it, both forward and backward.

  If you use either *more* or *less* on a file (or any stream of data — we'll do something fancy with *more* later, so you can see what I mean), you can escape by hitting **q** (short for "quit").

- file

  Let's say you don't know what type of file *myfile* is. No biggie.

  Type **file myfile**

  The output should be *myfile: ASCII text*

Move stuff

- cp

  Oh, but we forgot to add an extension, showing that it's a text file, didn't we? Not a problem.

  Type **cp myfile myfile.txt**

  And then, to see how it worked, list the contents of the current directory. The output should be *myfile myfile.txt*

  Now you have the original, *myfile*, plus an exact copy ("cp" - copy) with a different name, *myfile.txt*

- rm

  Now you have a file you don't need. No problem!

  Type **rm myfile** — and then list the contents of the directory; the output should be *myfile.txt*

  You've just removed ("rm" - remove) *myfile*.

  This is a command to use carefully! Removing things is *serious business*.

- mv

  Copying and deleting every time you want to rename a file is a little tedious. There's a way to do the same thing in one step:

  Type **mv myfile.txt mynewfile.txt** — and then list the contents of the directory; the output should be*mynewfile.txt*

  You just *moved* ("mv" - move) a file. (You might ask why the command isn't "change name," or something. Valid question. This command *is* used for changing filenames, but it is also used for moving files between directories.)

If you want to try copying or moving your file out of */home/action/[your name]* and into */home/action*, for instance, here's the command: **cp mynewfile.txt ../mynewfile.txt** — you can replace 'cp' with 'mv' — and then you can move it back by typing **mv ../mynewfile.txt mynewfile.txt**

Find stuff

To show you how to find things, we're going to have to go further afield. And we're going to digress for a moment.

Type **cd /usr/bin**

Type **ls**

Whoa, right? Here, try this:

ls | more — that character in the middle is the horizontal line, usually located above the "enter" key on your keyboard

WHOA, RIGHT? — This technique is referred to as "piping output to more", which is a very useful thing to keep in your toolbox when you're faced with large files, or, in this case, large output streams from commands. (It is totally reasonable to have questions about this. Ask!) As you can see, you can hit **enter** to keep scrolling through the whole file list; when you're tired of looking at the list, remember, you can type **q** to get out of *more*.

One more tool for our toolset, before we move on:

Did you see the sequence of files, *pygettext*, *pygettext2.5*, *pygettext2.6*, etc.? (They're later in the alphabet, so you're more likely to see them on an *ls* than on an *ls | more*. No biggie if you don't go looking for them. Just believe me that they're there, OK?)

Type **more pyg** *and then hit the* ***tab*** *key*. — your command should now say **more pygettext**.

That's right; it will try to complete file names for you after you've started typing them, if you hit *tab*. Tab completion is AMAZING and makes every command line user's life better. But it's got limits, of course. The file you actually want is pygettext2.7; but because there are multiple files that all start with "pygettext", it completed as far as it could, and you'll have to specify further by finishing the filename yourself.

Make sure the command line says **more pygettext2.7** and hit **enter**

You can browse the file if you want, and hit *q* at any point to get out of *more* — big file, right? It would be awful to have to look for a specific word in that file, wouldn't it?

- grep

Type **grep verbose pygettext2.7** — remember that you can use tab completion on the filename!

You'll see output something like this (but better-indented):

```
--verbose
'style=', 'verbose', 'version', 'width=', 'exclude-file=',
verbose = 0
elif opt in ('-v', '--verbose'):
options.verbose = 1
if options.verbose:
if options.verbose:
```

What's happening? The command *grep* searches within a file for instances of a particular set of characters, in this case "verbose". (It doesn't have to be a whole word. Any character string will work. And, yes, capitalization DOES matter.) The output shows you all of the places "verbose" appears within the file*pygettext2.7*.

It's time to learn another important tool: flags.

Type **grep -n verbose pygettext2.7**

Now the output has line numbers! The *-n* flag told the command *grep* that you want to know what line number each instance of the character string "verbose" appeared on.

(There are flags for **ls**; I usually use the flags **a**, **h**, and **l**, so when I list the contents of a directory, I do it by typing **ls -ahl** — see how the flags are combined? It's fine to go ahead and try that command in this or any directory. **ls -ahl /home/action** will give you a full list of all of the files in the directory we started in, for instance! I also like to use the **-c** tag with **nano**, so that I can see line and column numbers as I work.)

- man

Want to know more about how to use *grep*, or any other command? Use *man*

Type **man grep**

As you can see, there are a number of flags and options available.

Feel free to scroll through, or to hit **q** to exit.

(Real talk: I usually Google any command I want to use, if that option is available to me, rather than using*man*. Different people have different preferences for finding information.)

Just a couple more things....

- up arrow

Hit the **up arrow** (↑)

The last command you typed (probably *man grep*) is now on the command line! If you keep hitting the up arrow, you can scroll through your command history. This saves a lot of time when you're typing long commands that you need to use more than once!

- ctrl-c

You can stop the current command from running by hitting **ctrl-c**