Week 2: Intro to Bash

This week's lab is going to be short and sweet, and include lots of links to external resources (for your reference now and after this class is done). There's a lot of writing here, but there's not too much actual work involved yet.

Using the terminal (also called the command line) is crucial for bioinformatics, among many other things, and this lab will lay the foundation for you to become proficient in using tools on the terminal. Before we get into how to access it on various operating systems, let's discuss what it actually is.

1. What is the command line?

The command line is an interface designed for you to navigate directories and view files on a computer. This is also called the terminal; it looks like this:

```
[audrey@tilden:~$ ls
local  myenv  R
[audrey@tilden:~$ cd /space/s1/audrey/
[audrey@tilden:/space/s1/audrey$ ls
IAV_reads_simulator  influenza_reference  influenza_reference_GISAID  software
[audrey@tilden:/space/s1/audrey$ cd influenza_reference_GISAID/
[audrey@tilden:/space/s1/audrey/influenza_reference_GISAID$ ls
1d_copy_all_sequence.fasta  isolate_segments_report.txt     output_unique_NP.fasta
all_sequence.fasta          metadata_df.csv                 output_unique_NS.fasta
check_segments.py           MP.fasta                        output_unique_PA.fasta
combined_filter.fasta       NA.fasta                        output_unique_PB1.fasta
combined_unique.fasta       new_extract_id_filter_fasta.py  output_unique_PB2.fasta
copy_all_sequence.fasta     new_final_filtered.fasta        PA.fasta
count_HA                    new_final_HA.fasta              PB1.fasta
count_MP                    new_final_MP.fasta              PB2.fasta
count_NA                    new_final_NA.fasta              seqkit
count_NP                    new_final_NP.fasta              seqkit_linux_amd64.tar.gz
count_NS                    new_final_NS.fasta              unique_HA.fasta
count_PA                    new_final_PA.fasta              unique_MP.fasta
count_PB1                   new_final_PB1.fasta             unique_NA.fasta
count_PB2                   new_final_PB2.fasta             unique_NP.fasta
downloaded_raw_GISAID       no_primers                      unique_NS.fasta
duplicate_filter.py         NP.fasta                        unique_PA.fasta
duplicate_info              NS.fasta                        unique_PB1.fasta
failed_fasta                output_unique_HA.fasta          unique_PB2.fasta
filter_sequences.py         output_unique_MP.fasta          without_H3N1
HA.fasta                    output_unique_NA.fasta
audrey@tilden:/space/s1/audrey/influenza_reference_GISAID$
```

Example Terminal with Color Formatting

In the above example, you can see multiple types of commands, such as `ls` and `cd`. These are programs, included by default with unix systems such as Linux or Mac OSX. These programs are included as part of something known as a `shell`; in Unix distributions (and when downloaded on Windows) this shell is named `bash`, and on Mac OSX this is generally `zsh`. (The difference between the two is unimportant for our purposes right now.) I'll refer to the shell as `bash` from here on out.

The majority of the tools we will use in this course are operated from the command line, just like these. In essence, what you'll be doing is navigating around a file system just like you might on your own computer using Windows

explorer or the Mac OSX finder. This method has several advantages, though, because of all the programs you'll have access to which are best used on the command line.

2. How do I access the command line?

This will differ based on the operating system you're using. Let's divide this up into three categories.

- Linux (includes Chrome OS)
  Press `Ctrl+Alt+t` to open a terminal. Congratulations.
- Mac OSX
  Click on the Spotlight Search icon (magnifying glass) at the top right corner of your screen. Search 'terminal', and open the application. I recommend pinning this to your dock/taskbar; you'll be using it a fair bit. Detailed instructions here.
- Windows
  Go to the Windows store and download Windows Terminal.

If you have trouble getting this working, let me know. It should be a fairly painless process in most instances, though, now that Windows has its own terminal! (That's relatively new.)

---

3. Basic Commands

First we're going to want to do the things you're used to doing in normal file explorers like OSX Finder or Windows Explorer.The two most important commands are the ones you can see used in the example picture above: `ls` and `cd`.

- `ls`

`ls` is fundamental and super important. Short for the word 'list' (proof), `ls` shows you the contents of a directory, and `ls -ls` shows you more information about each file in the directory!

```
[audrey@tilden:/space/s1/audrey/influenza_reference_GISAID/downloaded_raw_GISAID$ ls -ls
total 3276172
 30232 -rw-r--r-- 1 audrey audrey  30957568 Jun 23 23:24 gisaid_epiflu_isolates_10.xls
 29908 -rw-r--r-- 1 audrey audrey  30623744 Jun 23 23:24 gisaid_epiflu_isolates_11.xls
 19960 -rw-r--r-- 1 audrey audrey  20438528 Jun 23 23:24 gisaid_epiflu_isolates_12.xls
 23400 -rw-r--r-- 1 audrey audrey  23960576 Jun 23 23:24 gisaid_epiflu_isolates_1.xls
 18512 -rw-r--r-- 1 audrey audrey  18955264 Jun 23 23:24 gisaid_epiflu_isolates_2.xls
 23280 -rw-r--r-- 1 audrey audrey  23837696 Jun 23 23:25 gisaid_epiflu_isolates_3.xls
 27404 -rw-r--r-- 1 audrey audrey  28059136 Jun 23 23:25 gisaid_epiflu_isolates_4.xls
 27684 -rw-r--r-- 1 audrey audrey  28346880 Jun 23 23:25 gisaid_epiflu_isolates_5.xls
 29060 -rw-r--r-- 1 audrey audrey  29756416 Jun 23 23:25 gisaid_epiflu_isolates_6.xls
 28316 -rw-r--r-- 1 audrey audrey  28994048 Jun 23 23:25 gisaid_epiflu_isolates_7.xls
```

- Advanced usage (not necessary, just fun)
  `ls` has many options that can change what it shows you and how that data is displayed. My favorite is `ls -thora`, which will show all the files in your current directory including hidden files, sort by date modified, and present it to you in a different format than standard `ls`. Try it! For a full list of options, see the `ls` manual page by typing `man ls` and pressing enter.

- cd

`cd` is probably the most common command you'll use, and the easiest (in my opinion) to remember. `cd` stands for 'change directory'. It just moves you from one folder to the next, exactly like clicking on a folder in a file explorer.

`cd` has the advantage of being able to navigate anywhere in one command if you give the full location of a directory; the most analogous function is in Windows explorer when you paste the location of a file in the top navigation bar. (If there is one on OSX let me know, I don't use that.) Let's wait a little bit to try that one out.

---

4. Creating, modifying, and moving files & directories

Alright, now to take your fancy new terminal for a spin.

When you SSH into the class server, you're automatically going to be in your home directory (`/home/studentX`, where X is your ID number). This is your personal workspace, for storing important files and doing whatever else you might need to do for the course. To see the full path, type the following command:

```
pwd
```

- pwd

`pwd` stands for print working directory. It gives you the name of the folder you're currently at, which is useful for moving files around and navigating quickly. The location of files and directories on the terminal is called the `PATH`.

Next, let's make a directory that we'll work in for today. Call it whatever you like- if you're not feeling creative, call it 'tutorial' or 'sandbox' or what have you. You can create a directory with the following command:

```
mkdir tutorial
```

- mkdir

`mkdir` stands for, as you might have inferred, 'make directory'. Pretty straightforward. You have to provide a name for this directory, though- ex. `mkdir tutorial`.

Now we're going to navigate into this folder- remember our old friend `cd`? Try it now-

`cd tutorial` (or whatever you named it)

Now that you're here (welcome!) let's try creating a file. It'll be empty at first, but that's okay. You can create files in bash like so:

`>filename`

Creating and deleting files

Creating files is easy- just use the `>` character, followed by the name of the file you'd like to create. (like so: `> test.file` or `>test.file`)

This file will be empty unless you're putting something in it. Also, if there's a file with the same name as the one you're trying to create, `>` will overwrite that file- be careful!

Removing files is also pretty simple, although removing directories is not (for good reason). To remove files, use `rm`, like so:

`rm test.file`

`rmdir directory`

---

5. Connecting to the server

The server is a powerful computer designed to manage, store, and deliver resources (like files, data, or applications) to other computers.

SSH

SSH stands for 'Secure Shell'. Remember how the terminal is also called a shell? We're just going to be connecting to a new terminal session on the server over the internet. This is one of two most important commands to remember for this class, so write it down somewhere and remember you can always come back here to see it again.

Usually the it looks like: ssh username@server_ip_address

```
ssh username@tilden.biol.berkeley.edu
```

Finishing up

Now, to finish the tutorial, and your lab for today. Let's make a file called 'hello_world.txt'.

```
>hello_world.txt
```

Then we're going to add some text to it, using only the command line!

First, try this: `I love science`. See how it prints text to the terminal for you to read? Now we're going to put that text in a file, using echo and the `>>` operator. Try this:

```
echo "Hello world!" >> hello_world.txt
```

The `>>` operator will append the output of another program to the end of a file; since your `hello_world.txt` is currently empty, "Hello world!" is all that file will consist of.

Next, you're going to want to turn in the PATH to your file; to get the PATH of a file (rather than a directory, like `pwd` does), try this:

```
realpath hello_world.txt
```