Essentials of Unix/Linux

Goal: To gain familiarity with, and comfort using, the Unix command line. The majority of bioinformatics programs are command line tools. Familiarity with the command line will make navigating these tools and their outputs much easier.

Notes about Syntax and Formatting

Before getting started with the classes provided in this worksheet, it is important to understand the way in which commands will be provided. In particular, the backslash character ("\") and the angle bracket characters ("<" and ">") are often a source of confusion for those not already familiar with their uses in Unix/Linux commands.

The following command illustrates the most common meaning of the backslash character in these exercises.

```
samtools faidx ~/rnaseq/magnaporthe_oryzae_70-15_8_supercontigs.fasta \
Chromosome_8.7 ~/magnaporthe_oryzae_70-15_8_single_contig.fasta
```

In this example command, the backslash – found by itself at the end of the line – denotes a line continuation. This is one command split onto multiple lines. In the Unix/Linux terminal, the command could have been written on a single line and it would wrap in the window. In print, we often do not have enough space to fit the command onto one line and it must be continued on the one below it.

So, When you see the backslash character at the end of a line do not type it – just continue typing the commands that are seen on the next line(s)

An example usage for the bowtie2-build command shows how angle brackets are often used in the commands for this workshop.

```
bowtie2-build [options] -f <reference_genome> <index_prefix>
```

In this example usage, the text enclosed in left and right angle brackets represents the names of user-specified parameters for the bowtie2-build command. In an actual command, it is expected that the angle brackets will be omitted; this will become clear as you compare the provided usages with the actual commands we will be using. It is also worth mentioning that this command includes "[options]." The square brackets ("[" and "]") simply denote that "options" represents many possible user-specified parameters rather than just one. Again, the square brackets are omitted.

Both the backslash character and the angle bracket characters have additional meanings in Unix/Linux commands. These are covered later in this class.

As a final note on formatting, it is important to realize that the formatting of the documents for this workshop prevents many of the commands from being copied and pasted into **PowerShell**, **PuTTY** or **Terminal** properly. The formatting often adds invisible characters to the commands, often breaking commands while making the problem very difficult to find. Thus, it is advisable to type the given commands out yourself, and to use tab-completion (discussed later in this class) to lessen the burden of doing so.

1.1 Getting Connected and Reconnected

1.1.1 On a PC

PowerShell

Open the **PowerShell** program. You can use the **ssh** (secure shell) command to connect to the MCC cluster:

- ☐ Let's try and connect (change myName to your LinkBlue ID:

```
• ssh myName@mcc.uky.edu
```

note: insert your assigned username and IP address in the appropriate fields - no space between username and the "@" symbol.

- ☐ After successfully logging in, you should now see a prompt, similar to:

```
[myName@mcc-login001 myName] $
```

You will see your name instead of **myName**. You are now "located" in your home directory (often abbreviated as ~ or ~/). The dollar sign $ at the end of prompt simply signifies the end of the prompt.

- ☐ To logout from your VM, just type:

```
• logout
```

This will close the connection.

- ☐ Try and log in again. We will assume that you can do this step in future classes without issue.

PuTTY

"PuTTY" is a free telnet/SSH client. Think of SSH (secure shell) as a simple, secure way for computers to communicate with each other.

- ☐ After locating PuTTY, just enter the IP address for your virtual machine, leaving the default port 22 unchanged, and click "Open". You will be prompted for your user name and password.

- ☐ After successfully logging in, you should now see a prompt, similar to:

```
[myName@mcc-login001 myName] $
```

You will see your name instead of **myName**. You are now "located" in your home directory (often abbreviated as ~ or ~/). The dollar sign $ at the end of prompt simply signifies the end of the prompt.

To exit PuTTY just type (note: do not type the bullet point - it just indicates that you should type everything that follows):

```
• exit
```

This will close PuTTY. Try to start PuTTY and log in again. We will assume that you can do this step in future classes without issue.

1.1.2 On a Mac

If you are working on a Mac, simply open the program **Terminal**, found in the *Utilities* folder. You can use the **ssh** (secure shell) command to connect to your virtual machine.

- ☐ Let's try and connect (change myName to your LinkBlue ID):

```
• ssh myName@mcc.uky.edu
```

note: insert your assigned username and IP address in the appropriate fields - no space between username and the "@" symbol.

- ☐ After successfully logging in, you should now see a prompt, similar to:

```
[myName@mcc-login001 myName] $
```

You will see your name instead of **myName**. You are now "located" in your home directory (often abbreviated as ~ or ~/). The dollar sign $ at the end of prompt simply signifies the end of the prompt.

- ☐ To logout from your VM, just type:

```
• logout
```

This will close the connection.

- ☐ Try and log in again. We will assume that you can do this step in future classes without issue.

1.2 Exploring and Learning Basic Commands

Let's try to familiarize ourselves with your VM by using a few commands.

- ☐ Display your working directory using **pwd** (print working directory).

```
• pwd
```

You will see something like this:

```
[myName@mcc-login001 myName] $ /home/myName
```

This is your current working directory, meaning you are virtually located here. Any actions you do will assume you are here; and will create files here unless you instruct otherwise.

- ☐ Change to the shared project space for this workshop. This is where we will do most of the work in this class, and is shared between all members of the class.

- `cd /pscratch/jdu282_brazil_bootcamp2023/`

Once you settle on a working directory for a given project, there is no need to change directories again. In fact until you are familiar with the command line environment, you will get into trouble if you do so. You should learn how to operate on other directories and files by specifying the "paths" to these resources. More on this later…

☐ Let's see what's in the current directory. We will use **ls** to list its contents:

- `ls`

You should see a whole bunch of text indicating all the stuff that's in that directory. Now we'll create a new directory inside the current one:

- `cd students`
- `mkdir [your name]`

This has created a new directory called *example* inside the *unix* directory.

☐ List your current directory's contents again. You will now see a new directory called *[your name]* has been added:

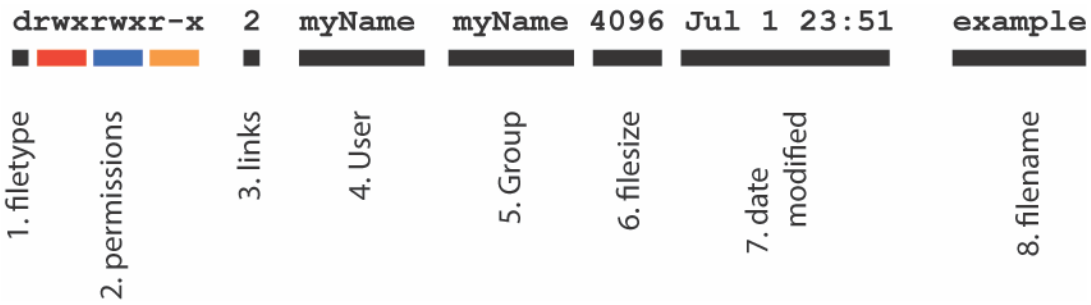
How do we know it's a directory? One, the name is colored blue. Another way to tell is the following:

☐ Repeat the previous step with option –l (lowercase letter L – not the number one). This is the "**l**ong" directory listing that shows more details:

- `ls –l`

```
myName@myName:~/unix$ ls –l
total 1416
drwxrwxr–x   2  myName  myname  4096      Jul  1 23:51  example
–rw–r––r––   1  myName  myName  1443704   Jul 17  2015  Excel.xlsx
```

What does this all mean? Below is an example key:



For file permissions, "r" means "can read", "w" means "can write", and "x" means "can execute". Notice there are nine columns (or three sets) of permissions in the example; these three sets of

permissions designate the settings for the user/owner (in red on the key), the group (green), and others (everyone else on the machine, in blue on the key).

So, other than by simply remembering it, how do you know what "–l" does?

☐ Show the manual for **ls**:

- `man ls`

Every Linux program has a manual page installed and is accessible via **man** ***program-name***. Press **q** to quit the **man** program.

☐ Change your current working directory to the newly created example directory:

- `cd [yourname]`

You can tell if this worked because the command line prompt should now say something like this but with your name instead of "jdu282"

```
[jdu282@mcc-login001 Julian]
```

☐ Let's change back to the home directory:

- `cd ../..`

Here instead of specifying an explicit path, we're giving a relative path.

You can view the file system as a tree. Meaning:

```
cd ..                   # change to my parent directory
cd ../..                # change to my parent's parent directory
```

Also, note that "." refers the current working directory:

```
cd .                    # change to where I am (doesn't change anything)
```

Hints for a productive time:

1) Your shell supports tab-completion, which means you don't usually have to type the full names of commands or files. For example, if your file's name is FinalProjectDecember172010.txt, you could simply type **cat Final<tab>** and it would complete the line for you. If there are two or more files starting with FinalXXX, pressing <tab> twice will display a list of all the possibilities. Then one can simply type the next unique character and hit <tab> again to complete the whole file name.

   ☐ Try it using a file/folder name in your directory

   Alternatively, your shell also supports wildcard characters (the * character), so typing **cat Final*** would have edited everything that started with the word "Final".

2) I used the **cat** command above. What does it do? How can you find out? Use **man**.

3) A quick way to repeat commands is to scroll through your previous commands by using the up and down arrow keys.

1.3 Learning a Text Editor

Most bioinformatics analysis programs cannot read files that are written in familiar programs such as Word, Notepad, Pages, TextEdit, Excel, etc. Using files produced by these programs can cause many frustrating problems that are often hard to diagnose.

☐ As an example, let's have a look at the first 10 lines of the provided Excel document (*Excel.xlsx*):

- head Excel.xlsx

It doesn't make much sense, does it? That's because it is written in a compressed format. **Even if we were to export this file as a .txt document, many programs would have a problem opening it.** For this reason, it is vital to learn a plain text editor. There are many text editors available for Linux (**vim/nano/pico/emacs/joe**), but we'll focus on **nano** today. Let's start by using **nano** to create and save a text file that simply contains your name. Let's save it in your *example* directory.

☐ **Navigate back to the *home* directory**:

- `cd ~/`

☐ Start **nano** and we will edit a file called *name.txt*. **nano** will create the file if it does not exist. You can add a directory path before the filename to specify where it should go (by default, this would be created in your current working directory). The following edits *name.txt* within the *example* directory:

- `nano students/[yourname]/name.txt`

Note: we don't need to be in the *example* directory to create a file there. We simply tell the system where we want the file to be created by specifying the path.

☐ The **nano** screen has two main sections: a buffer to edit text and a menu of common shortcuts. The ^ character in the shortcuts section represents the control key. For example, ^O means hold

the control key and "o" key together. Shortcuts in **nano** do not use the shift key, so ^O really means ^o. WriteOut means save (think of it as writing out to disk).

- ☐ Test **nano** by entering your name.
- ☐ Save your file. (^o).
- ☐ **nano** will prompt you to give a filename for the file you will save. By default, this is the file (*example/name.txt*) you gave to **nano**. Hit enter to write to that file.
- ☐ Exit **nano** (^x).
- ☐ Open your file again with **nano** and verify the results. **Remember to specify the file path**.

Pressing ^g (the control key and g together) will display common **nano** shortcuts. ^c will display a status bar with your current position (line, column, and character numbers). You can quit the shortcuts info by hitting "**q"**.

Copying and pasting can be difficult at first. When using **PuTTY**, highlighting with the mouse selects and copies text; right-clicking pastes text. In **PowerShell**, you can highlight text with your mouse and press the enter key to copy text, and, as in **PuTTY**, you can right click to paste text.

Within **nano**, ^k will cut the current line and ^u will paste. You can copy the current line by pressing the shift key, alt key, and 6 key simultaneously. There might be a noticeable pause when pasting large amounts of data.

You can use **nano** to view a text file safely; if you accidentally edit the file, just exit without saving by answering "no" when prompted.

Alternatively, you can use a command called "cat" (concatenate):

```
cat example/Name.txt
```

If the file is long, "less" is a command that will let you page-up and page-down the file. It is often more useful than cat for viewing large files.

```
less example/Name.txt
```

1.4 Using SCP to Transfer Files

scp (secure copy) is a command-line tool that can send files to a remote machine or that can download files from a remote machine. Pick a random (non-sensitive or embarrassing) file from your computer and let's upload it to MCC.

SCPing to a remote machine involves the basic syntax of: 1) define the file (path-to-file/filename); 2) define the destination and name of the copied file (machine-address:path-to-file/filename):

```
scp <file_to_transfer> \
username@address.of.destination.machine:target_directory/filename
```

SCPing from a remote machine uses the syntax: 1) Specify the required target file's location (machine-address:path-to-file/filename); 2) indicate where the file is to be saved on the local machine and what its name should be (path-to-file/filename).

```
scp username@address.of.remote.machine:source_directory/filename \
target_directory/filename
```

In each case, transfer will begin upon entry of a valid password.

☐ Type the following but replace <your.username> with your linkblue username:

```
• scp [file on your computer]
  yourname@mcc.uky.edu:/pscratch/jdu282_brazil_bootcamp2023/studen
  ts/yourname/
```

☐ At the prompt, enter your MCC password.

This will copy the file across the network to your student folder in our shared MCC space.

☐ It can be a bit tricky to try and scp from the MCC or another cluster to your personal computer (e.g., off the top of your head, what is the IP address of your personal computer? I have no idea for my computer, but I do know my MCC address. So it's often easier to go from your personal computer to MCC or from cluster to cluster.

1.5 Downloading from the Internet

Normally, when you want to download a file from the internet, you open a browser search for the file and then download it by clinking on a link or an icon. How, then, do we download from the internet to a remote machine when we are not sitting in front of that machine and we have no browser? Fortunately, there are command line tools that allow us to accomplish this task. here, we will use a program called **wget** (web get) to download a file from the National Center for Biotechnology Information ftp site.

☐ Make sure you are in your student directory in pscratch.

☐ Type the following **(for wget, type it all on one line without the backslash at the end of each line):**

```
• mkdir example
• wget https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/146/045/\
  GCF_000146045.2_R64/GCF_000146045.2_R64_genomic.fna.gz -O \
  ./example/yeast.nt.gz
```

You should see the following runtime message:

```
myName@myName:~/example$ wget
https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/146/045/GCF_000146045.2_R
64/GCF_000146045.2_R64_genomic.fna.gz -O example/yeast.nt.gz
--2021-01-28 11:43:48--
https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/146/045/GCF_000146045.2_R
64/GCF_000146045.2_R64_genomic.fna.gz
Resolving ftp.ncbi.nlm.nih.gov (ftp.ncbi.nlm.nih.gov)... 165.112.9.229,
2607:f220:41e:250::7, 2607:f220:41e:250::11, ...
Connecting to ftp.ncbi.nlm.nih.gov
(ftp.ncbi.nlm.nih.gov)|165.112.9.229|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3843460 (3.7M) [application/x-gzip]
Saving to: 'yeast.nt.gz'
```

```
yeast.nt.gz
100%[=====================================================================>]
3.67M  1.38MB/s    in 2.7s

2021-01-28 11:43:54 (1.38 MB/s) - 'yeast.nt.gz' saved [3843460/3843460]
```

☐ List the contents of your directory to view the results:

- `ls example`

A gzipped (compressed) archive was downloaded. It must be decompressed before we can look at it.

☐ Let's take a quick peek at its contents:

- `head example/yeast.nt.gz`

Again, the contents are not human-readable.

☐ So let's decompress it:

- `gunzip example/yeast.nt.gz`

This will produce a file called *yeast.nt*.

☐ List the directory to check.

☐ Let's have another peek to check the decompression:

- `head example/yeast.nt`

☐ This shows the first ten lines in the file. You should see a "sequence header" line with information about the sequence, followed by lines with As, Cs, Gs and Ts – the actual DNA sequence data.

☐ Suppose we want to remove this file now.

- `rm example/yeast.nt`

List the contents of your directory. The *yeast.nt* file is gone, permanently. There isn't really a concept of a trashcan when dealing with the terminal/console. To read more about **rm**, check out its man page.

☐ Re-download the file using **wget**. Remember, a quick way to repeat commands is to scroll through your previous commands by using the up and down keys.

☐ Create a new directory underneath your *example* directory called *source*. You should know how to do this by now, without a reminder…

☐ Move the .gz file you just downloaded into the new directory by the following (recall that you can use tab to complete the names of files/directories – but not the file that you are about to create):

- `mv example/yeast.nt.gz example/source/yeast.nt.gz`

☐ Suppose we want to copy this file back to the original directory:

- `cp example/source/yeast.nt.gz example/yeast.nt.gz`

☐ List your current directory and then the example directory to check that a copy was made.

☐ Suppose we want to copy this file to a backup in a different directory and rename it at the same time.

- `cp example/source/yeast.nt.gz example/yeast.backup.nt.gz`

Note: the first argument of **cp** (copy) is the file you wish to copy, while the second argument is the destination, which can also include directory information.

☐ Check that the backup file was copied and then remove it.