Wyoming INBRE Bioinformatics Workshop

June 14, 2016

Linux Tutorial

Vikram Chhatre & Nicolas Blouin

University of Wyoming

vchhatre@uwyo.edu
nblouin@uwyo.edu

Contents

1 Using the Terminal (Shell) 3

2 Navigation 3
2.1 Where am I? . . . 3
2.2 Unix Directory Structure . . . 3
2.3 Moving around the Directories . . . 4

3 Connecting to the server 5
3.1 Two-Factor Authentication . . . 5

4 File Manipulation 6
4.1 Creating New Files . . . 6
4.2 New Directories . . . 7
4.3 Copying and Moving Files and Directories . . . 8
4.4 Displaying File Contents . . . 9
4.5 Searching within Files using `grep` . . . 10

5 Documentation 11

1 Using the Terminal (Shell)

A terminal is a program that allows you to perform various tasks without the use of graphical user interface (gui). Most linux computers have some kind of terminal shell program available. Examples include `xterm`, `aterm` etc. On Mac OSX computers, there is only one choice – `Terminal.app`. When you first start the terminal shell, you will see something like the following.

```
user@uwyo:~$
```

Here the operator before the `@` sign is your username on this computer and the operator after (`uwyo`) is the name of the computer itself. The ∼ represents the path of the current directory and $ indicates that the unix is expecting your input. It is also called the dollar `prompt`. This prompt may look different on different machines; for example, on some distributions of linux, the $ is replaced by the symbol `>`.

2 Navigation

One of the important skills when using the terminal is to be able to navigate around various folders (directories) and files. When using a gui interface, you can navigate using pointing and clicking with a mouse. But here you will need to execute various commands to move around.

2.1 Where am I?

The command `pwd` will tell you where you are in the directory structure. The acronym stands for `print working directory`.

```
user@uwyo:~$ pwd
/home/user
```

This tells you that you are inside your home directory. Whenever you open a new terminal session, by default it always starts in your home directory. Notice that paths in unix use forward slashes, compared to backward slashes in windows. The first forward slash represents the root of the directory structure in linux. Think of it as the main trunk of the tree, from which various branches arise. The branches are analogous to the various directories that are located under the root directory.

2.2 Unix Directory Structure

So what exactly is inside the root directory? First let's navigate there using the `cd` (change directory) command. Then use the `ls` (list contents) command to check the contents of that directory.

```
user@uwyo:~$ cd /
user@uwyo:~$ ls
bin   dev  home        lib    lost+found  mnt  proc  run   snap  sys  usr  vmlinuz
boot  etc  initrd.img  lib64  media       opt  root  sbin  srv   tmp  var
```

As you can see there are a handful of directories here. Each has a specific set of function. The one you will most commonly use is the `home` directory where you will have full permissions (read-/write/execute). As a regular user of this unix computer, you will also have access to the other directories inside the root, but they will be mostly read/execute type permissions.

Now navigate to the home folder and take a look at what's inside.

```
user@uwyo:~$ cd /home
user@uwyo:~$ ls
user1  user2  user3  user4  user5
```

As you would expect, every user's home directory is located inside the `/home/` folder. You will only have access to your own home folder and not anyone else's.

2.3 Moving around the Directories

During the course of your operations you will need to move forward or backward along the directory structure. The `cd` command, as we saw before, will help with this. Let's look at some examples.

Go to your home directory from anywhere in the system (shortcut). Here, ∼/ is a direct replacement for `/home/user`.

```
user@uwyo:~$ cd ~/
```

Navigate to where some of the programs you will use are located.

```
user@uwyo:~$ cd /usr/bin
user@uwyo:~$ ls
vim        perl        python        ...
```

If you wanted to go one level back from `/usr/bin/` to `/usr`, use the following command.

```
user@uwyo:~$ cd ../
user@uwyo:~$ pwd
/usr
```

How will you modify that command to go two steps back?

3 Connecting to the server

Often with bioinformatics analysis, you will be performing tasks on a remote network server, one much more powerful than your workstation. Let's use the terminal program you just learned about to connect with this server.

On the information sheet provided to you, there is a username and password combination and details of the server. You can use this information to log onto the Unix server set up for today's exercise. We will use Secure SHell (SSH) protocol to talk to the remote server. In the following example, replace `train101` with the user name provided to you.

3.1 Two-Factor Authentication

To improve computer security, University of Wyoming has started requiring two factor authentication to grant access to network servers. Two factor implies password plus a second type of authentication. You all have received a USB key-like device which provides that second authentication. This device called **Yubikey** needs to be inserted into one of the USB ports on your workstation.

```
ssh train101@mtmoran.uwyo.edu
```

This will bring up the password prompt.

```
train101@mtmoran.uwyo.edu's password:
```

As you type the password, you will not see the cursor move. This is normal. During the two-factor authentication, we will use the password in combination with a long key present on the Yubikey. Enter the two as follows, don't forget the `comma`:

`PASSWORD,Press_Yubikey_Button_Once`

A keyboard `RETURN` after entering this information isn't needed. Pressing the Yubikey button will automatically generate a carriage return. If the authentication succeeds, the server will log you in and present a command prompt of style: `user@server:~$`.

```
Welcome to the University of Wyoming Mount Moran compute cluster hosted by the ARCC.
[train101@mmmlog2 train119]$
```

If you see this, you have successfully logged on to the server.

4 File Manipulation

4.1 Creating New Files

The most common unix file type is simple text. It is very easy to create new files and organize them into directories. Let's create a few files using the command `touch`.

```
[train101@mmmlog2 train119]$ touch file1.txt
[train101@mmmlog2 train119]$ touch file2.txt
[train101@mmmlog2 train119]$ ls
file1.txt        file2.txt
```

You can also create multiple files at once. For example:

```
[train101@mmmlog2 train119]$ touch file3.txt file4.txt file5.txt
[train101@mmmlog2 train119]$ ls
file1.txt        file2.txt        file3.txt        file4.txt        file5.txt
```

Note that these files are currently empty. You can populate a file with contents using a text editor (discussed later). However, sometimes you just want to insert a few words or characters. This can be done using the command `echo`.

```
[train101@mmmlog2 train119]$ echo "This is a test." > test.txt
[train101@mmmlog2 train119]$
```

Notice that the prompt returned after you executed this command. But no message was displayed. That's standard procedure in unix. A message is only presented in case of errors. No message indicates that the command was successful. You can now check the contents of the file we just modified, using the command `cat`.

```
[train101@mmmlog2 train119]$ cat test.txt
This is a test.
```

Files can be deleted using the command `rm` (**re**m**o**ve). Be cautious! File and folder deletion is permanent on unix. Think before you delete.

```
[train101@mmmlog2 train119]$ rm test.txt
[train101@mmmlog2 train119]$
```

You can delete multiple files at once.

```
[train101@mmmlog2 train119]$ rm file1.txt file3.txt file5.txt
[train101@mmmlog2 train119]$
```

If you want to delete all files that match a name pattern, you can do so using wildcard `*`. Be extremely careful with this option. This will permanently delete the files. Think before you delete

anything.

```
[train101@mmmlog2 train119]$ rm file*.txt
[train101@mmmlog2 train119]$
```

4.2 New Directories

Analogous to how you created new files, you can also create new directories. But the command is different. One or more directories can be created at the same time.

```
[train101@mmmlog2 train119]$ mkdir scripts
[train101@mmmlog2 train119]$ mkdir literature manuscripts data
```

At times you'd want to create a large number of directories with a similar naming pattern, e.g. seq1, seq2, seq3 etc.

```
[train101@mmmlog2 train119]$ mkdir seq{1..5}
[train101@mmmlog2 train119]$ ls
seq1  seq2  seq3  seq4  seq5

[train101@mmmlog2 train119]$ mkdir {1..5}seq
[train101@mmmlog2 train119]$ ls
1seq  2seq  3seq  4seq  5seq
```

Deleting the directories can also be done using the `rm` command. Also, wildcards and pattern matching can be used to delete multiple directories at once. Remember that deletion is permanent. There is no going back.

```
[train101@mmmlog2 train119]$ rm -r seq1 seq4
[train101@mmmlog2 train119]$ rm -r seq*
[train101@mmmlog2 train119]$ rm -r *seq
```

Here the `-` is called a switch, and `r` is an option passed to the `rm` program, which stands for `recursive` - i.e. delete all directories within this directory recursively.

4.3 Copying and Moving Files and Directories

Here we will review two commands: `cp` for copying and `mv` for moving/renaming files and directories. For copying files, you will need to provide two things: file name and a new name for that file (if copying locally) or a new destination. The first example below copies the file locally using a different name. Second examples copies the file to a new destination without changing its name.

```
[train101@mmmlog2 train119]$ touch seq.txt
[train101@mmmlog2 train119]$ mkdir dna
[train101@mmmlog2 train119]$ cp seq.txt  seq_copy.txt
[train101@mmmlog2 train119]$ cp seq.txt dna/
```

Likewise, the next example copies the directory `dna` locally under a new name `dna_copy`. Copying directories requires the switch `-r`, which means recursive. In the second example, the directory is copied to a new location without changing its name.

```
[train101@mmmlog2 train119]$ cp -r dna dna_copy
[train101@mmmlog2 train119]$ mkdir temp && cp -r dna temp/
```

Notice how we combined two commands into one above. First we created a new directory called `temp` and then copied the directory `dna` into it. This requires separating the two commands with `&&` characters.

You may also `move` files and directories from one location to another, or you may rename them in place as follows. The first example is simply renaming the directory in place. In the second example, the directory is being moved to another location. Practice this command on some files. Keep track of where you are moving stuff.

```
[train101@mmmlog2 train119]$ mv rna rna_copy
[train101@mmmlog2 train119]$ mv rna temp/
[train101@mmmlog2 train119]$ ls temp/
dna        rna
```

4.4 Displaying File Contents

Sometimes you are dealing with large files containing for example, DNA sequences, or alignments. Before loading these large files into a text editor, you may just want to peek into them. Unix offers several commands to do this e.g. `less` and `cat`.

The `head` and `tail` commands will let you quickly look at the top and bottom of the file. These two commands can also be used in combination to look at almost any part of the file quickly. Let's look at some examples. First, let's copy a file containing ribosomal 16s sequences from a fungal isolate of Douglas fir, a long-living tree from California.

```
[train101@mmmlog2 train119]$ mkdir seq
[train101@mmmlog2 train119]$ cp /project/train119/TrainingData/ribo16s.fasta seq/
[train101@mmmlog2 train119]$ less seq/ribo16s.fasta

>gi|480327979|gb|JX519349.1| Afipia sp. 935 16S ribosomal RNA gene, partial sequence
GAGCGGGCGTAGCAATACGTCAGCGGCAGACGGGTGAGTAACGCGTGGGAACATACCTTTTGGTTCGGAA
CAACACAGGGAAACTTGTGCTAATACCGGATAAGCCCTTACGGGGAAAGATTTATCGCCGAAAGATTGGC
CCGCGTCTGATTAGCTTGTTGGTGAGGTAACGGCTCACCAAGGCGACGATCAGTAGCTGGTCTGAGAGGA

>gi|480327980|gb|JX519350.1| Afipia sp. 919 16S ribosomal RNA gene, partial sequence
GCAAGTCGAACGGGCGTAGCAATACGTCAGTGGCAGACGGGTGAGTAACGCGTGGGAACGTACCTTTTGG
TTCGGAACAACACAGGGAAACTTGTGCTAATACCGGATAAGCCCTTACGGGGAAAGATTTATCGCCGAAA
GATCGGCCCGCGTCTGATTAGCTAGTTGGTGAGGTAAGAGCTCACCAAGGCGACGATCAGTAGCTGGTCT
```

The command `less` provides a similar functionality as the man page viewer (see section on documentation below). You can scroll up or down using ↑ and ↓ keys, though you can't make any changes to the file. This is a read only view. To exit, simply press `q`.

The command `cat` will `cat`alogue all the contents of the file to the screen. This is useful to view contents of small files, but not of large files. Try it.

Finally, `head` and `tail` will show you, by default, the first and last 10 lines of the file respectively. But you can manipulate them to display more or fewer lines using the switch `-n`. For example:

```
[train101@mmmlog2 train119]$ head -n 3 seq/ribo16s.fasta

>gi|480327979|gb|JX519349.1| Afipia sp. 935 16S ribosomal RNA gene, partial sequence
GAGCGGGCGTAGCAATACGTCAGCGGCAGACGGGTGAGTAACGCGTGGGAACATACCTTTTGGTTCGGAA
CAACACAGGGAAACTTGTGCTAATACCGGATAAGCCCTTACGGGGAAAGATTTATCGCCGAAAGATTGGC


[train101@mmmlog2 train119]$ tail -n 5 seq/ribo16s.fasta
GTACCTTCAGAAGAAGCCCCGGCTAACTACGTGCCAGCAGCCGCGGTAATACGTAGGGGGCAAGCGTTGT
CCGGAATCATTGGGCGTAAAGCGCGTGTAGGCGGCTTGGTAAGTCCGCTCTGAAAGCCCGGGGCTCAACC
CCGGGAGGCGGGTGGATACTGTCAAGCTCGAGTCCGGAAGAGGCGAGTGGAATTCCTGGTGTAGCGGTGA
AATGCGCAGATATCAGGAGGAACACCAATGGCGAAGGCAGCTCGCTGGGACGTGACTGACGCTGAGACGC
GAAAGCGTGGGGAGCAAACAGGATTAGATACCCTGGTAGTCCACGCCGTAAACGATGGGCACTAGGTGTG
```

4.5 Searching within Files using `grep`

`Grep` is a versatile and highly efficient tool to quickly search within text files for specific bits of information. For example, we can query the 16s sequence file from the earlier example for several important bits of information. The format of this file is called `fasta`, which consists of two lines of information per sequence. The first line contains the name of the sequence and the second line contains sequence itself. The first line always starts with the character `>`.

Thus by counting the numer of times the character `>` appears in the file, we can find out the number of records present in the file. Try the code below. Here, the `-c` switch tells grep to `count` the number of occurrences.

```
[train101@mmmlog2 train119]$ grep -c '>' seq/ribo16s.fasta
14
```

So there are 14 records in the file. If you wanted to just look at their names, but not the sequences, run the same command without the `-c` switch. We are only displaying the first few records here to save space.

```
[train101@mmmlog2 train119]$ grep '>' seq/ribo16s.fasta

>gi|480327979|gb|JX519349.1| Afipia sp. 935 16S ribosomal RNA gene
>gi|480327980|gb|JX519350.1| Afipia sp. 919 16S ribosomal RNA gene
>gi|480327981|gb|JX519351.1| Bradyrhizobium sp. 857 16S ribosomal RNA gene
```

Furthermore, if you were interested in count occurrences of a specific bit of DNA sequence in this file, `grep` can help with that also. Let's say for example, you want to count occurrences of a simple sequence `ATGGC`

```
[train101@mmmlog2 train119]$ grep -c 'ATGGC' ribo16s.fasta
32
```

This functionality can be very useful for quickly getting stats on large sequence files.

5 Documentation

Do these exercises make you wonder what other options may be available under the various commands you tried? Before you Google, know that unix provides comprehensive documentation for all these commands, available at your fingertips. In unix, they are called man pages, short for manual pages. Try the `man` command.

```
[train101@mmmlog2 train119]$ man mkdir

MKDIR(1)                                                User Commands

NAME
       mkdir - make directories

SYNOPSIS
       mkdir [OPTION]... DIRECTORY...

DESCRIPTION
       Create the DIRECTORY(ies), if they do not already exist.

       Mandatory arguments to long options are mandatory for short options too.

       -m, --mode=MODE
              set file mode (as in chmod), not a=rwx - umask

       -p, --parents
              no error if existing, make parent directories as needed
```

You can navigate the manual page(s) by using ↑ and ↓ keys. To search within the manual page, type `/search-term` and hit `return` key. To navigate within the search results, press `n` to go down, and `N` to go up. To exit the manual page, simply press `q`. Fancy? Try manual pages for other commands.