Using LINUX – a BCMB/CHEM 8190 Tutorial

Updated (1/17/12)

Objective: Learn some basic aspects of the UNIX operating system and how to use it.

What is UNIX? UNIX is the operating system used by most computers that do not use Windows or Mac OS. Most very large computers, and most specialized computers used by scientists use some version of UNIX. UNIX comes in many varieties, most of which have a lot in common. Macintosh computers, for instance, use UNIX (a version of BSD UNIX). Silicon Graphics computers use IRIX, lots of PC users use LINUX, etc. Fortunately, these varieties are very similar, and most of what you need to learn to be able to use any of these is common to them all. Some knowledge of UNIX and related subject matter is not only useful to scientists but obligatory. In this tutorial, we will get to know some simple UNIX commands, etc. The Biochemistry computer lab in the Life Sciences building at UGA has a collection of workstations (PCs) networked to a central server. These use a version of LINUX called RED HAT. The setup allows your accounts and files to reside on the server so that you can sit at (and login from) any workstation.

UNIX Help There are thousands of on-line UNIX help/tutorial sites, and hundreds of UNIX books for you to purchase (or check out of the library!). One such online UNIX tutorial can be found at: (**http://www.softlab.ece.ntua.gr/facilities/documentation/unix/unixintro/**).

UNIX Shells Unix "shells" are vehicles or interfaces for interacting with the UNIX "kernel". There are several different ones: Bourne shell (bash), C shell (csh / tcsh), Korn shell (ksh), etc.. For our tutorial today we will use the **c shell**. There are lots of websites that can tell you about the different shells (for instance **http://www.softlab.ece.ntua.gr/facilities/documentation/unix/shelldiff.html**).

Terminal We'll assume that you've sitting at one of the networked workstations. You will see a login page asking for your username (your accounts will be set up on the first day of class). After giving your user name you will be asked for a password. Enter the one given you and hit return on the keyboard. Remember that both your user name and password are case sensitive – anita, Anita, and ANITA are all different. One of your first tasks will be to change your password. At this point everyone at UGA should have completed the SecureUGA training modules. One of the modules gives advice on passwords. If you haven't completed this or need a refresher go to the website: https://secure.uga.edu/.

Once logged in you should see an icon called YourUserName_Home. Double click on this icon to get a home window. Go to the pull-down list under "File" on the top bar and select "open terminal". You will see a prompt containing the directory path you are in and ending with "$". This is where you will enter LINUX commands, followed by a return to execute them. You can expand this and any other window by moving the mouse pointer to any corner, holding down the left mouse button, and moving the mouse in the direction you want to expand. Clicking on the

"X" in the upper right corner will close the window. Obviously, you have to know, and remember, some commands in order to be able to use LINUX or UNIX effectively.

Entering Commands Before we enter commands, it is important to recognize some important facts about UNIX syntax. First of all, *UNIX is case sensitive*. Upper and lower case characters are distinct. For instance, the UNIX command **pwd** (print working directory) reports the name of the current directory. The commands pWd, PWD, PwD, Pwd, PWd, etc., are NOT valid commands. In what follows we ask you to enter some commands. What you type will be in **bold.**

The first thing you may want to do is change your password: At the prompt type

[UserName@........]$ passwd

(You have learned your first LINUX command) You'll be asked for your old password and then asked to create a new one. Don't forget your new password!

Type

[UserName@........]$ pwd

This command, "print working directory" prints the name of the current directory, or the "path" of the directory. The output should be something like the following:

```
[UserName@........]$ pwd
/home/UserName
```

Now type

[UserName@........]$ ls

This command lists the files and subdirectories of the current, or working, directory. You may not see anything listed. We will first have to create some files.

Options/Arguments and Man (manual) Pages. UNIX has a built-in help feature called the **man** pages (for "manual" pages) that helps to explain the commands and the options/arguments available for each of them. Type

[UserName@........]$ man ls

The man page(s) for the "ls" command will be displayed, one page at a time. You can access each page sequentially by pressing the space bar. You can quit any time by typing **q** (for quit). The options/arguments described for many UNIX commands are accessed by appending a dash (-) followed by (usually) single letter arguments that can be concatenated. Here is an example with four arguments concatenated. Type

[UserName@........]$ ls –ltFr

In this case, this command will list the directory contents in the "long" format (-l), sorted by when the files/directories were last modified (-t), oldest first (-r), and with a slash following all directories (-F) so that you can distinguish directories from other files.

Files, Directories, and Moving Around. Now we'll learn how to make new directories and learn how to navigate between directories. First type

[UserName@........]$ pwd

The response should be something like /home/UserName: this is the current directory (or the path to the current directory). Now, we will make a new directory called junk. The command is "mkdir" for "make directory". Type

```
[UserName@........]$ mkdir junk
[UserName@........]$ ls
```

The output should contain an entry "junk". Type

```
[UserName@........]$ ls -F
```

The entry "junk" should be followed by a "/", indicating that it is a directory. Now we'll change the current directory to the "junk" directory (we'll "go into" the "junk" directory). Type

```
[UserName@........]$ cd junk
```

The command "cd" stands for "change directory". Now, type

```
[UserName@........]$ pwd
```

The output should be something like /home/UserName/junk. Type

```
[UserName@........]$ ls
```

Since there are no files or directories in the "junk" directory, there should be no output for this command.

So, how do we get "back" to the previous directory? There are two ways, both using the "cd" command. First, in UNIX you can change to ANY directory simply by using the "cd" command followed by the complete path of the directory. In our case the path was "/home/UserName". So, type

```
[UserName@........]$ cd /home/UserName
```

What you will find is that, under normal circumstances, it requires a lot of typing to always enter the complete path to any directory for simple directory changes. In our case, we simply want to change from a subdirectory to the directory that contains it (in the hierarchical directory structure). In this case, we can simply type

```
[UserName@........]$ cd ..       or       [UserName@........]$ cd ../
```

(these two commands do the same thing)
This allows us to move "up" one directory (*from* /home/UserName/junk *to* /home/UserName).

Now, you should be back in the original directory (/home/UserName). Type

```
[UserName@........]$ cd junk
```

Now, you should be in the "junk" directory again. Type

```
[UserName@........]$ pwd
```

The output should be "/home/UserName/junk" Now, make a new directory in the "junk" directory called "test"

```
[UserName@........]$ mkdir test
[UserName@........]$ cd test
[UserName@........]$ pwd
```

The output should be "/home/UserName/junk/test". Now, go back to the original directory ("/home/UserName"). Type

```
[UserName@........]$ cd ../../
[UserName@........]$ pwd
```

The output should be "/home/UserName". So, from "/home/UserName/junk/test", to return to the original directory, one can use the command "cd ../" to move to "/home/UserName/junk", and then "cd ../" to move to /home/UserName", or go directly using "cd ../../".

Getting Tired of Typing? There are several useful shortcuts. One is that the up arrow will retrieve the previous command and place it on the current command line. You can then move along the text with the right and left arrows, delete with the backspace key, and type in new text – very convenient if you make a mistake in typing a command line. You can also use the mouse to highlight text anywhere on the screen. Hitting the center mouse button then pastes that text to the point where your command line cursor sits.

Creating and Editing Files We will now learn how to make text files and edit them. There are many nice text editors available for use on UNIX workstations. UNIX, however, has some built in text editors that are uniformly available in (most) all types of UNIX, so whatever type of workstation you sit down at, these should be available. One of them is called **vi**. This is a really good editor for you to learn to use, but it is somewhat difficult to learn, and takes a lot of time to learn. (An interesting commentary on text editors is available at **http://mark.stosberg.com/Tech/text_editor_review.html**).

We will use an editor called **gedit.** Gedit is quite easy to use, and our goal today is not to learn to "use" this editor, just to introduce you to it and use it to make some files. First, go to the "junk" directory. Type

```
[UserName@........]$ cd /home/UserName/junk
```

Type

```
[UserName@........]$ pwd
```

The output should be "/home/UserName/junk". Now we will create a file called "firstfile" type

```
[UserName@........]$ gedit firstfile
```

A new screen will appear. This is the **gedit** editor screen. Just start typing away. Type in a few lines of text. You can move about between the lines of text using the arrows on your keyboard (←, ↑, →, ↓). You can delete using the "delete" (backspace) key. Once you have some lines of text in the file, save your file by opening the 'File" option on the top menu bar (left mouse button) and selecting 'Save". Gedit will automatically save to the same file. If you select 'save as" and you will be asked for a new file name.

You will notice that the window that you used for UNIX command line entry is no longer active. It has transferred control to the editor window and will only become active when you close the editor. Close the window by clicking on the 'x' in the upper right corner or select quit from the file menu. Now type

```
[UserName@........]$ ls -F
```

Your "junk" directory should now contain one file called "firstfile" (and a subdirectory called "test").

Closing is not always convenient. Try opening the editor by typing:

```
[UserName@........]$ gedit firstfile &
```

Note that the command line window is still active. The addition of the "&" after the command starts the program in background. In this way you can open several applications at once. This will become useful later when you want to run some programs that take a while to complete. You can then work on other things while the program runs.

Now we'd like to edit "firstfile" . Your file should open in the gedit editor window. You should see the file that you created earlier. Make some changes in your file. Then save your changes by selecting 'save' under the file menu., This will overwrite firstfile. If you want to keep both versions use 'save as' and enter the name "**secondfile**" when gedit asks you for the file name. Now go to the still active command window and type

```
        [UserName@........]$ ls -F
```

You should see 2 files, "firstfile" and "secondfile" (and one subdirectory, "test"), and possibly a ~firstfile. This latter file is a temporary save in case you inadvertently overwrite a file. Close the gedit window.

Copying, Moving, and Renaming Files and Directories

Now we will learn some basics of copying and deleting files and directories. The current directory should be /home/UserName/junk. If not, go to that directory by typing

```
        [UserName@........]$ cd /home/UserName/junk
```

List the directory contents, discriminating between files and directories, by typing

```
        [UserName@........]$ ls –F
```

There should be two files ("firstfile" and "secondfile") and one directory ("test"). Now, lets make a copy of "firstfile". We'll call it "thirdfile". Type

```
        [UserName@........]$ cp firstfile thirdfile
```

This command makes an identical copy of "firstfile" (using "cp", which stands for "copy"), and gives it the name "thirdfile". You can verify this if you like by using the gedit editor to look at thirdfile. Let's make a copy of "secondfile", and call it "fourthfile". Type

```
        [UserName@........]$ cp secondfile fourthfile
```

Now, lets move the file called "fourthfile" out of the current directory and into the "test" directory using the UNIX "mv" command. We can do this with either of the following commands

```
        [UserName@........]$ mv fourthfile /home/UserName/junk/test     or
        [UserName@........]$ mv fourthfile ./test
```

The first of these defines the destination directory path explicitly, whereas the second defines it relative to the current directory ("./" indicates that the directory that follows is a subdirectory of the current directory). While we're at it, let's move the file called "thirdfile" into the test directory also

```
        [UserName@........]$ mv thirdfile ./test
```

Now, let's move "thirdfile" back to /home/UserName/junk

```
        [UserName@........]$ cd test
        [UserName@........]$ mv thirdfile ../
        [UserName@........]$ cd ../
        [UserName@........]$ pwd
        [UserName@........]$ ls –F
```

The output from the last two commands should tell you that the current directory now is /home/UserName/junk, and that the file "thirdfile" has been returned from the "test" directory to /home/UserName/junk.

```
        [UserName@........]$ pwd
        /home/UserName/junk
        [UserName@........]$ ls -F
```

```
firstfile       secondfile      test/           thirdfile
[UserName@........]$
```

Type

[UserName@........]$ ls ./test

This command will list the contents of the "test" directory. There should be one file in "test"; "fourthfile".

Now, let's rename the directory called "test". We will give it the new name "bob". This is also accomplished using the "mv" command. Type

[UserName@........]$ mv test bob
[UserName@........]$ ls –F
[UserName@........]$ ls ./bob

The last commands will show that the directory "test" is now called "bob". "bob" contains the single file "fourthfile".

```
[UserName@........]$ ls -F
bob/            firstfile       secondfile      thirdfile
[UserName@........]$ ls ./bob
fourthfile
[UserName@........]$
```

Now, we'll make a copy of the "bob" directory. We'll call it "obo". Type

[UserName@........]$ cp –r bob obo

The "cp" (copy) command, used to copy files, is also used to make copies of directories. The option/argument "-r" must be used (try copying a directory without the "-r" option….see what happens).

Let's make one more copy of "bob" for the next exercise. Type

[UserName@........]$ cp –r bob boo
[UserName@........]$ pwd
[UserName@........]$ ls -F

We now have three subdirectories of /home/UserName/junk called "bob", "obo", and "boo", each of which contain the file "fourthfile".

```
[UserName@........]$ pwd
/home/UserName/junk
[UserName@........]$ ls -F
bob/             firstfile        secondfile
boo/             obo/             thirdfile
[UserName@........]$
```

Removing / Deleting Files and Directories In Windows, and in other nice, friendly interfaces like Mac OS X, there is a trash can for you to throw the files and directories in that you don't want any more. If you decide later that you really don't want to throw them away, you can go and retrieve them from the trash. This is a nice and safe way to deal with deleting files. In Unix, there is no nice safe way. There is no trash can – there is no safety net. Once you remove a file, it is gone, never to return. Don't forget this. Be careful.

Make sure you are in the directory /home/UserName/junk. Type

[UserName@........]$ cd /home/UserName/junk

```
[UserName@........]$ pwd
[UserName@........]$ ls –F
```

You should see the three directories “bob”, “obo”, and “boo”. We are going to delete two of these. There are two commands for removing files and directories, “rm”, and “rmdir”. The first, “rm” will remove files or directories, with the correct options. The second, “rmdir”, will remove directories that are empty (contain no files or subdirectories). Let’s test this latter fact. Type

```
[UserName@........]$ rmdir obo
```

You should get an error message saying “Directory not empty”. So, first let’s go into the “obo” directory, remove all of its contents, and then try this again. First go into the directory. Type

```
[UserName@........]$ cd obo
[UserName@........]$ ls –F
```

You should see that there is only a single file in the directory “obo”, called “fourthfile”. To remove this file, type

```
[UserName@........]$ rm fourthfile          or          [UserName@........]$ rm *
```

The first command removes only the file fourthfile. Since that was the only file in the directory, the directory will now be empty. The second command gives an example of the use of “wildcards” in UNIX commands. This command removes all files in the directory. You have to be extremely careful using wildcards when deleting files, so that you don’t inadvertently delete files or directories that you had not intended to.

Now, if you return to the “junk” directory, you should be able to remove the “obo” directory using “rmdir”. Type

```
[UserName@........]$ cd ../
[UserName@........]$ rmdir obo
[UserName@........]$ ls –F
```

The last command should confirm that the directory “obo” is gone.

You can remove directories that are *not* empty using “rm” with the “-r” option. Like the “cp” command, the “-r” option indicates that a recursive operation is to be performed on a directory (and its contents). We will remove the directory “boo” with this command. Type

```
[UserName@........]$ rm –r boo
[UserName@........]$ ls –F
```

You’ll see that “boo” is gone.

Examining File Contents You have seen that text editors like gedit can be used not only to create new files but also to look at the contents of files that have already been created. UNIX provides other means also to view or otherwise examine the contents of text files.

Make sure you are in the directory /home/UserName/junk. Type

```
[UserName@........]$ cd /home/UserName/junk
[UserName@........]$ pwd
[UserName@........]$ ls –F
```

You should see the file “firstfile” that you created earlier. Type the following

```
[UserName@........]$ cat firstfile
```

The command “cat” simply lists the contents of the file to the screen. This is good for quickly dumping the contents of (short) files to the screen to get a quick look at what is in them. It is not so good, usually, for long files. Try the following

[UserName@........]$ more firstfile
The more command allows you to examine the contents of large text files one page at a time, like the manner in which the "man" pages are displayed. If the file is more than one page, pressing the space bar will scroll to the next page. Typing **q** will exit the command.

UNIX has a really nice command called "grep" that allows you to search through files in a directory for words or strings. The command replies with a list of all the files containing the word or string. For instance, my "firstfile" contains the word "machine" (it contains this word twice). If I type

[UserName@........]$ grep machine *

I get the following response:

```
[UserName@........]$ grep machine *
grep: bob: Operation not permitted
firstfile:So, even today, on both my SGI machines and on my Macs, when I pull up a
firstfile:worry that it won't be there on any UNIX machine when I sit down at one.
secondfile:So, even today, on both my SGI machines and on my Macs, when I pull up a
secondfile:worry that it won't be there on any UNIX machine when I sit down at one.
thirdfile:So, even today, on both my SGI machines and on my Macs, when I pull up a
thirdfile:worry that it won't be there on any UNIX machine when I sit down at one.
[UserName@........]$
```

The grep command found each instance of "machine" in "firstfile", "secondfile", and "thirdfile". If there had been other files with the word "machine" in them, "grep" would have found them also. This is a very powerful command for finding any type of information in text files. There are also options for searching for files that do *NOT* contain a certain string (see man pages for lots of options), etc.

Finding Files The Windows and Mac OS X interfaces have nice utilities for finding files on disks. The Mac OS X utility is based on the UNIX "find" command. We will demonstrate how to use the UNIX command to find files.

First, go the directory /home/UserName

[UserName@........]$ cd /home/UserName

Now, we will search all subdirectories for a file called "fourthfile" (we happen to know that this file is in our directory /home/UserName/junk/bob). Furthermore, let's assume that we really don't remember if we named the file "fourthfile", or "fifthfile" or perhaps "fourthfil", etc. We know that "fil" is in the name. Type the following command

[UserName@........]$ find ./ -name '*fil*' -print

This command will search the current directory and all subdirectories ("./") for all files that contain "fil" ('*fil*') in the name (-name), and it will print (-print) the results to your screen. If you execute the command you will find "thirdfile", "fourthfile", etc. :

```
[UserName@........]$ find ./ -name '*fil*' -print
.//junk/bob/fourthfile
.//junk/firstfile
.//junk/secondfile
.//junk/thirdfile
[UserName@........]$
```

You will also see a number of files that begin with a period (.xxx). There are configuration files that are normally hidden – you seldom want to change them. They appeared because of the wildcard (*) at the beginning of the test search. You can also see these files if you use the **–a** modifier with your **ls** command. The “find” command is a very powerful one. Consult the man pages for lots of options and also examples.

Directing Input and Output In UNIX, the output from one command can be directed in particular ways. The output can be directed into a file, or it can be directed to serve as the input for another command. These are usually referred to as “redirection” and “piping” or “pipes”. These are very powerful aspects of UNIX and are often used to reduce human intervention (and workload).

Make sure you are in the directory /home/UserName/junk. Type

```
[UserName@........]$ cd /home/UserName/junk
```

First, we’ll direct the output from a simple command into a file. For instance, we’ll list the contents of our directory and put it into a file, rather than display the output on the screen. Type

```
[UserName@........]$ ls –F > directorylist
```

In this command, we took the output from the command “ls –F” and created a new file called “directorylist” with the output of “ls –F” in it. Type

```
[UserName@........]$ cat directorylist
```

You’ll see that the file “directorylist” contains the output from the command “ls –F”
You can also append output to an existing file using “>>”. Type

```
[UserName@........]$ ls –F >> directorylist
[UserName@........]$ cat directorylist
```

You’ll see that the output of ls –F now appears twice in “directorylist”. If you use > instead of >> you will over write the previous contents. Again, be cautious using these commands.

Now we’ll use a UNIX “pipe” to use the output of one command as the input for the next, and then we’ll direct the result of the second command into a file. Type

```
[UserName@........]$ ls *t* | grep bob * > result
[UserName@........]$ cat result
```

The file “result” will contain all instances of the occurrence of the string “bob” in files in your directory that have a ’t’ in their name (for instance, “bob” should appear in the file “directorylist”).

Aliases Are you tired of typing “ls –F” al the time? Maybe you would like to minimize keystrokes, or maybe you are tired of trying to remember certain commands, etc. Well, UNIX allows you to make up your own commands with the “alias” command. For instance, maybe every time you want to list your directory contents you want the output to be consistent with “ls –F” but you don’t want to type “ls –F” all the time, maybe you would just like to type “l” (lower case “L”). Try typing this

```
[UserName@........]$ alias l ‘ls –F’     (“l” is lower case L, not “one”)
[UserName@........]$ l
```

Now, simply typing “1” will execute “ls –F”. You can alias any commands you like. These alias definitions will disappear when you logoff and you will have to re-enter them next time. There is a way to shortcut this process as well. You can add them to your t shell file (.tshrc), a script that runs every time that you log in. Changes in this file don't take effect until the next login unless you use the '**source**' command. Changing shell files can be risky – mistakes can disable your account. So we won't do this.

Monitoring and Terminating Processes When you open a terminal (a Bourne shell, C shell, or a K shell), you begin a "process". Likewise, if you start a program running from the shell, a new process is created. You can monitor the processes that are running on your computer. You can stop the ones that belong to you. Processes are monitored using the "ps" command. Type

```
[UserName@........]$ ps –u
```

This command displays the processes of the current user (you). If you look at the output, on the right side is the "command", i.e. in this case it will tell you that you are using something like a "csh" or a "bash" (C shell or Bourne shell). On the left the output tells who you are and the PID ("process ID"). The PID is important if you want to end a process. If you need to end a process, you would use the "kill" command. For instance you would type (don't actually type this)

```
[UserName@........]$ kill –9 PID
```

where PID is the process identification number. The "-9" option guarantees a "sure kill".

If you want to see what *everyone* is doing, you can type

```
[UserName@........]$ ps -ef.
```

Some Miscellaneous Commands There are many, many UNIX commands. Some are quite useful and powerful……some are there to add certain conveniences…..some are there for no apparent reason. Here are some miscellaneous commands:

```
[UserName@........]$ clear    (clears the screen)
[UserName@........]$ whoami   (tells you who you are logged in as)
[UserName@........]$ who  (tells you everyone who is logged in)
[UserName@........]$ date  (returns the day, date, time and year)
[UserName@........]$ cal  (displays a calendar for the current month)
```

Logging Out To leave the system close windows using the "x" in the upper right corner or typing "**exit**" on the command line. Then go to the pull-down list from the "System" tab on the top line and select "Log Out".