

BSC3402L (6): Experimental Biology — Comparative Genomics

Laboratory Section: Thursdays from 4:00 to 6:00 PM.

Introduction to Linux

Week One, Thursday, January 11, 2007

Author and Instructor: Fredrik Ronquist

Lab covers: (1) Logging into a Linux system; (2) Using the Windows-like interface to Linux (the so-called GUI or Graphical User Interface); (3) Using the UNIX shell (“Terminal”); (4) Printing from the classroom computers; and (5) Assignments.

Fredrik Ronquist
School of Computational Science
Florida State University
Tallahassee, FL 32306-4120
ronquist@scs.fsu.edu
850-645-1325

The computers we will be using during this course run Red Hat Linux (Fedora). Underneath its Windows-like surface, Red Hat Linux has a command-line (text driven, non-graphical) Unix system, much like Mac OS X does. In both Red Hat Linux and Mac OS X, you can access the underlying system through a Terminal application. A regular Windows (Windows XP etc) machine also has a command-line system underneath, which is accessible, for instance, if you run 'command.com' using the 'Run' option in the Windows start menu. However, the Windows command-line system is MSDOS and not Unix. Linux, referred to in the name Red Hat Linux, is simply a dialect of Unix, originally developed by the Finnish computer science student Linus Torvalds. On top of the Linux / Unix foundation, there is additional software, called KDE, creating the Windows-like look and feel of your classroom machines.

Unix is the platform favored by most computational biologists. The reasons are many, including: (1) Unix is more reliable than any other operating system; (2) Unix is written from the ground up to support multi-tasking; (3) Most clusters and supercomputers use Unix; (4) Unix is free and open-source; (5) Many new technologies are developed first on Unix systems; and (6) Unix has many convenient and powerful tools for managing jobs and text files.

To facilitate coming lab sessions, we will spend this first afternoon familiarizing ourselves with Red Hat and Linux / Unix. Like Mac OS X and more recent versions of Windows, Linux machines are typically password-protected. The computer keeps each user (also called account) separate, such that users can work simultaneously but independently of each other. Before you can continue with this lab, you need to acquire a user name and password.

1. Logging in

If your computer is running, you should see a log-in screen asking for a user name. Type in your user name. Note that the user name is case sensitive, like almost everything else on a Unix machine. That is, 'Fronquist' is different from 'fronquist'. You will be presented with a second screen asking for your password; type that in as well. After a while, you should be presented with a desktop much like the desktop on a Mac or a Windows machine. You might also get 'Kandalf's Useful Tips'. Close this window before proceeding.

2. Using the Graphical User Interface

Click on the Red Hat menu on the lower left; it works like the Start menu in Windows and gives you access to installed programs, among other things. Launch the Firefox web browser, by single-clicking (instead of double-clicking) on the Firefox symbol, which you should be able to find, among other places, in the quick-launch bar at the bottom of the screen. The Linux graphical user interface often uses single-clicking where Windows and Mac OS would use double-clicking. If you double-click on Firefox, the operating system will think that you want to start two different Firefox sessions under different user profiles, potentially, so it will ask you if you wish to create a new user profile. Just exit this window if it appears. Navigate to the course home page on Blackboard (<http://campus.fsu.edu/>) by simply typing in this URL in the address bar. Bookmark this page so that you can always get to it easily when you are using the classroom machines. You can set the

course page as your home page, if you wish, by selecting Edit>Preferences and pressing 'Use Current Page' in the Home Page pane. Make sure you can access the course syllabus by clicking on that link. If you wish, spend a few minutes surfing around, just to make sure that everything works the way it should. Close Firefox.

Create a new folder called "Lab1" on the desktop by right-clicking anywhere on the desktop. Note that folders are referred to as directories in Linux / Unix. Put the new folder in the 'Home' folder. Open KEdit (you will find this under 'Accessories' in the Red Hat menu) and type in some sample text (minimally 20 lines) into the window that appears. Save this document in the new folder (Lab1) under the name 'test.txt'. To find your new folder in the Home folder, you may need to select 'Home Folder' from the options at the left. You type in the name of the file you wish to save in the 'Location' box. Then close KEdit. Now open up the Home folder and the Lab1 folder by clicking on these folders on your desktop. Make sure you have created a new file and that it is located in the right folder. Notice that when you saved the file, the 'path' to that file (its location in the directory hierarchy) was indicated in the top bar of the KEdit window. If you saved the file directly to the Lab1 directory, the end of the path would have been './Lab1/'. A path that ends with a slash (/) is the 'address' of a directory; a path that ends in a letter (or any other character) is the 'address' of a file. To see the path to your file again, open KEdit once more and choose File>Open Recent in the menu. When your mouse hovers over 'Open Recent', you should see the complete path of your previous file, ending with './Lab1/test.txt'. Now close KEdit again.

The folders (directories) you see on a Unix machine may not be stored on that machine. Your folders on the classroom machines, for instance, are stored in a separate file system that is accessed by all classroom machines. The folders belong to your account and they can be viewed regardless of what classroom machine you log into. Test this for yourself by logging out from your machine (Log out is at the bottom of the Red Hat menu). Trade places with a peer. Log in using the new machine and make sure that you can see all the folders you saw on the old machine, including your new folder "Lab1" and the new text file 'test.txt' you just created.

3. Using the Command-Line Interface

For many computational and other tasks, we will be using the command-line Linux / Unix interface. Everything you can do by dragging and dropping folders etc in the graphical user interface can also be done by typing in text commands. The command-line interface also allows you to do a lot of things that you cannot do or cannot do easily with the graphical user interface. To access the command line, launch **Terminal**, which should be available in the quick-launch bar at the bottom of the screen. A terminal window is commonly called 'shell' by Unix people. By default, Terminal will open the most popular type of shell, a *bash* shell (bourne again shell; named after Stephen Bourne, who wrote the first programmable shell for Unix).

The most difficult part of using the command line for the first time is to keep track of the directory you are currently working in. By default, the initial folder or 'directory' where you enter the system is your *home directory*. It is equivalent to the folder named 'Home' on your desktop. The 'prompt' (what is printed to the left of the cursor) in the shell indicates where you are. Close to the end of the prompt, you will probably have the symbol ~, which is a Unix synonym for your home directory. Now click on the 'Home' folder on your desktop

and examine its contents. Then ask the Unix system to display a list of the contents of the home folder by typing **ls** (abbreviation of *list*) after the prompt in the shell (the Terminal window). When you hit return, you should get a list of the directories and files that are contained in the 'Home' folder. Note that the directories are colored blue and the files are colored black (if you have any). If you want to see more information about each item than just the name, type **ls -l** and hit return. The system should now display the files and directories together with file sizes and some other information. The first set of seven letters/symbols is important. It will read something like 'drwxrwxrwx' with some of the letters probably replaced by dashes (-). The first position indicates whether the item is a directory (d) or a file (-) (a few other types are also possible). The next three positions indicate whether you have permission to read (r), write (w) and execute (x) the item. Any letter replaced by a dash indicates that you lack the corresponding privilege. The next group of three letters indicate the same privileges but for the group of users to which you belong. The last three letters signal the privileges for users that do not belong to your user group.

The third and fourth columns of the output lists the name of the owner and the group for each file. Then there is a column indicating the file/directory size, then the date when the file/directory was last modified, and finally the name of the file/directory.

Now we are going to move, in the shell, from your home directory to the 'Lab1' directory. Do this by simply typing **cd Lab1** (cd is for **c**hange **d**irectory). Confirm that you are in the right directory by typing **ls -l**. The **-l** flag stands for the *long* format. It is very common to give options to commands in Unix by adding single letters preceded by a dash after the command name. Note that there is no space between the dash and the ell (l). After you type 'ls -l', you should get a list with the 'test.txt' file as the only member. Listing the contents of a directory is a nice way of checking where you are in the system. Another way of confirming that you are in the right directory is to type **pwd** (print working directory), which will give you the full name of the current directory, including its 'address' in your folder system. Note that forward slashes are used in Unix to separate a parent directory from its daughter directory. The name will include at least one slash; you might see something like './<your_login_name>/Lab1'. The above directory name and address, or 'path' as it is also called, should be read as: "I am currently in the folder Lab1, which is contained in the folder <your_login_name>. Your login name is a synonym for your home directory, just as the tilde (~) is. Note that most addresses of web sites (URLs) are paths pointing to a file in a folder on a Unix system (the Unix machine running the web server). For instance, this lab tutorial has the address: <http://www.scs.fsu.edu/~ronquist/compngen/LabUnix.pdf>, which should be interpreted as pointing to the file 'LabUnix.pdf' in the folder 'compngen' in the folder '~ronquist' on the machine called (with the address) 'www.scs.fsu.edu'. The 'http://' is not part of the address, it defines the protocol that should be used in accessing the file.

We are now going to move back and forth between the 'home' and 'Lab1' directories by using different methods. First change from 'Lab1' to 'home' by typing **cd ..** (cd followed by space and two periods). The two periods are used to refer to the parent directory, the next more inclusive level. In this case, this is our home directory. Confirm with **ls** that you are in the right directory (the directory should contain the 'Lab1' folder).

Before we move on, let us create a few new folders, for instance 'Lab2' and 'testalongfoldername'. Do this by typing **mkdir Lab2** and **mkdir testalongfoldername**. Confirm with **ls** that the new folders have been created. Now we will move from the home directory to the test directory. However, let us first see what happens if we type in an erroneous file name. Type **cd Testalongfoldername**. Because Unix is case-sensitive, the folder 'testalongfoldername' is not the same as the folder 'Testalongfoldername', so the folder you specified cannot be found by the system ("No such file or directory"). When you have very long commands, it is easy to make typing mistakes like this one, causing a lot of unnecessary frustration. Fortunately, the bash shell offers a service called command completion, which helps to address the problem. The idea is to ask the Unix system to try to complete your command based on the part of the command that you have already entered. In our situation, we have three folders: 'Lab1', 'Lab2', and 'testalongfoldername'. Already when we have typed **cd tes**, the system should be able to figure out that we want to change directory to the folder named 'testalongfoldername' because this is the only directory starting with 'tes'. This is exactly what it does if we simply press <tab> after typing **cd tes**. Note that the system will put a forward slash after the name to indicate that it is a folder name and not a file name. If the shell is not able to guess the last part of the command after we have pressed <tab>, it will not do anything. Pressing <tab> twice should produce a list of the possible alternative completions.

No matter where we are in the folder hierarchy, we can always shift back to the 'Home' folder in a single step by typing **cd** (cd without a folder name). Test this and confirm that you are back in the home directory by typing **ls -l**. Now, let's remove the folder 'testalongfoldername'. We could do that by using 'rmdir testalongfoldername' (rmdir for remove directory), but this would not work if the folder contained some files. So let us use the more general (and much more dangerous) **rm -r testalongfoldername**. This will remove (rm) the folder testalongfoldername along with all contents. The r in -r stands for recursively. If you do not give the -r flag, rm will only remove files matching the name you specify, not folders. Check that the folder is now gone from the home directory.

Unix allows wild-card characters (*) to be used in specifications of file names. For instance, you could use 'rm test*' to remove the files 'test1', 'test2' and 'testing' at the same time. The command 'rm' is extremely dangerous because there is no 'Trash' can on a Unix system. The deleted files are immediately and irretrievably lost when 'rm' removes them.

Now, it is time to look more closely at the file 'test.txt' using the most common text file handling commands in Unix. First, change to the directory 'Lab1'. Then display the last few lines in the file by typing **tail test.txt** (you can also use command completion and type **tail tes<tab>**, for instance). The last few lines in your file should now be output to screen. You can display the first few lines in the file by typing **head test.txt**. To see the entire file, we can use the 'more' command by typing **more test.txt**. The command 'more' will display the file one page at a time; you can display the next page by pressing the space bar. Quit 'more' by pressing **q** (a common exit mechanism for Unix commands/programs) (if 'more' hits the end of the file, it will quit automatically by itself).

On most Unix commands, there is help information available by typing 'man <command>' (man for manual). Assume for instance that you needed some help on the options of the 'ls' command. Type **man ls** to access the help information for this command. Turn pages using the space bar and exit by pressing **q**, as if you had displayed the manual pages using 'more'.

Editing text in Unix is commonly done using the text editors 'emacs' or 'vim', the latter also known as 'vi'. However, both of these editors are quite complicated and probably overkill for this course. Instead, we suggest you use the small text editor 'pico' for your text editing needs (you can also use one of the available graphical text editor in your Linux system). Start pico by typing **pico** at the shell prompt. This will start you on a new file. If you wanted to edit an existing file, you would type 'pico <filename>' instead, which would open up an editing window with the contents of the file <filename>. At the bottom of the screen, you will see some help information giving you an overview of available commands. The ^ sign refers to the Ctrl key. For instance, ^C means <Ctrl> + C. Write some text in the window and save the file by pressing <Ctrl> + O (WriteOut). The first time you save the file, the system will ask you for the name to save the file under. When you're done, exit 'pico' by pressing <Ctrl> + X. Check that the saved file contains the text you entered by using **more**. The file should be in your home directory.

You may want to copy or move a file to a different location. Assume, for instance that you wanted to copy the file 'test.txt' from the folder 'Lab1' to the folder 'Lab2'. There are several ways of doing this; here is one: First change to the home directory by typing **cd**. Now use the copy command 'cp' by typing **cp Lab1/test.txt Lab2**. This will copy the file Lab1/test.txt to the folder Lab2. Note that the path to test.txt is given relative to the current location. If we had been in the directory Lab1, our copy command could have been given as **cp test.txt ~/Lab2** (copy the file 'test.txt' in the current directory to the folder Lab2, which is contained in the home directory). Sometimes you want to move a file without copying it. This is accomplished using the 'mv' command. Say, for instance, that we wanted to move the file that we just created with 'pico' in the home folder to the folder 'Lab1'. Do this, for instance, by navigating to the folder Lab1 (type **cd Lab1**) and then typing **mv ~/<filename> .** (where <filename> is the name of the file created by 'pico' and the period is used as a synonym of the current working directory, which happens to be the Lab1 directory where we want the file to end up). The move command can also be used to rename a file. For instance, rename your 'test.txt' file 'test1.txt' by moving to the Lab1 directory, if you are not already there, and then typing **mv test.txt test1.txt** (move the contents of test.txt to the file test1.txt). Type **ls** and confirm that 'test.txt' has been renamed 'test1.txt'.

We will cover one last thing before returning to the graphical user interface. The Unix system contains a number of commands for conveniently and safely accessing remote machines and copy files between machines. This is achieved using the 'ssh' (secure shell) and 'scp' (secure copy) commands. You have access to the account 'compngen' on the machine 'ronquistg5.scs.fsu.edu'; the password is 'guest'. To access this account from your shell, type **ssh compngen@ronquistg5.scs.fsu.edu** (open a secure shell (ssh) displaying the home directory of the user 'compngen' on the machine ronquistg5.scs.fsu.edu). Your machine will ask you if you want to add the new machine to your list of trusted sites. Just answer yes. After that, you will be asked for the password of the user 'compngen' on the machine ronquistg5.scs.fsu.edu. Type in **guest**.

After the password has been checked, you will be given a prompt acknowledging that you are logged into the account `compgen` on the machine `ronquistg5.scs.fsu.edu`. Note that, although your shell looks much the same, you are now working off of a remote machine. You can now run any command available on that remote machine. Navigate to the folder 'mrbayes' by typing `cd mrbayes` or simply `cd mrb <tab>` to use command completion. Then list the contents of this directory using `ls -l`. Choose one file for copying back to your system, for instance the file 'mb.h'. We would copy it back to the home folder on our machine 'class10' by typing `scp mb.h <your_user_name>@class10.scs.fsu.edu:~` (secure copy the file 'mb.h' in the current working directory to your account on the machine `class10.scs.fsu.edu` into the home directory (~). You will be asked for your password and, after that, the file should be copied over. Now copy an additional file using the same mechanism, for instance the file 'mcmc.c'. You could type in the entire `scp` command from scratch but `bash` offers you a more convenient option. Press the **up-arrow key**. You should now get the most recently entered command replicated on your command line. If you press the up arrow more times, you will move back to the second most recent command, the third most recent command, etc. The down arrow will take you back down the history list of commands to the most recent command. Edit the most recent command by deleting 'mb.h' and replacing it with 'mcmc.c'. Then hit enter and the copying process will start as before. When the file has been copied, log out of the remote machine by typing `exit`. The prompt should now signal that you are back onto your classroom machine. Use `ls` to confirm that the files were copied over correctly. You can also use 'more' to examine the files. If you copied the file 'mb.h', it is a C header file forming part of the source code of MrBayes, one of the programs we will be using later in the course; 'mcmc.c' is a C source file for MrBayes.

Question to think about, if you have time: When you copied your files from the remote machine into your home directory using 'scp', did you have to log back into the machine you were sitting in front of or could you have used any machine in the classroom? [Answer: You could have used any classroom machine, since the files are copied onto the shared file system.]

Before closing the shell, delete the folder 'Lab2' and its contents, and the files you have added to the home directory (unless you want to keep them of course). Then exit the shell by typing `exit`. The terminal window and the Terminal application should close automatically.

4. Printing from the Linux machines

Return to your graphical KEdit and open up a new file. Enter some text and try printing the file by simply choosing File>Print in the drop-down menus. Check that the selected printer is 'pr152', which is the classroom printer. When you click OK, your page should print to the printer in the classroom. You can also test opening the text file you created with `pico` in KEdit and print that file.

5. Class Assignment

You will have two assignments today. The first assignment is to create a text file listing the Unix commands that you used today and that you think you will use during the course, and write a few words reminding you about the function of each command. Create this file on your Unix system with `pico` or with KEdit and hand it

in to Clemens before you leave. Make sure you store this file for future reference in a directory where you can find it; you can add Unix commands to it later on as we cover a few additional commands in the phylogenetics labs. Make sure your name and the course name (Comparative Genomics Spring 2007) are both on the page you hand in (no more than one page please).

6. Homework Assignment

Spend the last part of the lab session thinking about an interesting topic that you could work with during this course. It should be something within the field of computational comparative genomics, broadly construed. Find a suitable heading and then briefly describe your topic in one to five sentences. Finally try to find three testable scientific hypotheses relating to the topic of your choice and derive one testable prediction for each. Do not worry if the description of the topic is vague at this point; we will refine it during the course. It is also possible for you to change topic later on if you like. To be able to formulate good hypotheses, you are allowed to make bold assumptions that may be false, as long as these assumptions are made explicit in your text. The assignment is due next week during the lab session. Make sure the course name (Comparative Genomics Spring 2007) and your name are both indicated on the sheet you hand in (no more than one page).