# CS 656: Linux Command Line Lab

Instructor: *Sergio Salinas Monroy*
This assignment is based on Ben Roose's CS 444 laboratory assignments

**Deliverable: A single PDF file containing the terminal logs for all the exercises below. Screen shots will NOT be accepted. So make sure your logging is setup correctly before starting.** Your submitted file should include all your activity during the exercise, including failed commands, typos, and other mistakes. It is NOT necessary to restart or edit the logging if you make a mistake.

REMEMBER: To make typing faster while working through these exercises, you can use Bash command history and shortcut keys (keybindings), such as Tab, CTRL+A, or CTRL+E. You can see the Bash command history by typing `history` in the terminal or by using the up and down arrows.

## Exercise 1

### Navigating through the Linux File Hierarchical System (FHS)

1. Start the VM that you installed in the first lab, and open a terminal.
2. Document this lab assignment by logging the commands and outputs of the terminal. To start logging your terminal acvitity into a file type `script myLogFile.log`. This command records what you see on the terminal into the `myLogFile.log` file. If you need to pause the exercise, type `exit`. This will stop the logging. To resume logging into the same file, type `script -a myLogFile.log`. The `-a` flag appends the new logging to the log activity already in `myFile.log`.

   Note that if you do not use the `-a` flag, you will overwrite the contents of your file and lose your work.

   To see the contents of your log file, you can type `cat myLogFile.log`. Or use other commands that can interpret the log, such as `more`.
3. At the command prompt, type `pwd` and press Enter. The `pwd` command **p**rints your present **w**orking **d**irectory (in this case, it is your home directory). Any commands you enter in your present working directory affect the commands outcome.
4. To see the files in your present working directory, type `ls` and press the Enter key. The `ls` command lists files in your present working directory only.
5. To see how your present working directory affects the outcome of commands, change your present working directory to a new location by typing `cd /etc` and press Enter. The `cd` command allows you to **c**hange your present working **d**irectory. In this case, your present working directory is now the `/etc` directory (the location of several configuration files).
6. Type `pwd` and press Enter. Do you see `/etc` displayed? You should.
7. Type `ls` and press Enter. You should see different files than you did in step #5. You see different files because your present working directory is different than it was previously.
8. Type `cd` and press Enter to go back to your home directory. The `cd` command with nothing entered after it is a shortcut to change your present working directory to your home directory.
9. Type `pwd` and make sure you are back in your home directory.
10. You can see the type of files in your home directory by using the `{F` option to the `ls` command. Type `ls -F` and press Enter. You should see several files followed by a forward slash (/). This indicates the file is a directory rather than a file. Files will not have the slash.

11. You can see a different file type by typing `ls -F /bin/uname` and pressing Enter. An asterisk after a file means it is executable (can run as a program). In this command, `/bin/uname` tells **ls** to only display the file `uname` in directory `/bin`.

12. You can see another file type by typing `ls -F /etc/rc5.d/` and pressing Enter (several files will show). A @ after a file means it is a symbolic link. Symbolic links are analogous to shortcuts in Windows.

13. There are dot files (aka hidden files) in your home directory. Look at them by typing in `ls -a` and pressing Enter. A dot file, also called a "hidden file," starts with a dot (period) and is not displayed unless the `-a` option is used with the `ls` command.

    Dot files are usually used to store the configuration parameters for programs.

14. Now just type in `ls` and press Enter. You should not see the dot (hidden) files in this listing. This happens because you need to use the `-a` option on the `ls` command in order to see them.

15. Now try a long listing, by typing in `ls -l` and pressing Enter. The `-l` option is a lowercase L and not a number one. Don't worry about all information shown; you will learn about this information later.

16. Take a look at the items at the root (/) filesystem by typing `ls /` and pressing Enter. Can you tell whether these are directories or not?

17. Add an indicator to the end listed item by typing `ls -F /` and pressing Enter. Remember that the forward slash (/) indicator tells you that the item is a directory, whereas a blank indicates the item is a file.

18. Now you will try something different and attempt to change your present working directory to various places across the filesystem using relative and absolute directory references. First navigate to the /bin directory by typing `cd /bin` and press Enter. Entering a directory or file reference starting with a forward slash (/) is called using an "Absolute Directory Reference."

19. Now see where your present working directory is currently located by typing in `pwd` and pressing Enter. This indicates your present working directory is now at the /bin directory.

20. Use another absolute directory reference to take you back to the user account's home directory by typing `cd` *`directory`* and pressing Enter, where *directory* is your account's home directory, which you recorded in step #3. Remember that an absolute directory reference MUST start with a forward slash (/).

21. Check that you are in your account's home directory by typing in `pwd` and pressing Enter.

22. Try using a relative directory reference to enter a subdirectory under your account's home directory. Type `cd Downloads` and press Enter. (Note: if your home directory does not have a subdirectory named Downloads, then pick another subdirectory of your choice).

23. Check that you are in the Downloads directory (or the other directory you chose) by typing in `pwd` and pressing Enter.

24. Use another method to take you back to your home directory by typing `cd` ∼ and pressing Enter. The symbol after the `cd` command is a tilde (∼) symbol. The tilde (∼) symbol is typically located above the Tab key on your keyboard.

25. Check that you are in your home directory by typing in `pwd` and pressing Enter.

Exercise 1 continued on next page... 

26. Now you will navigate up one directory level using a relative directory reference. Type `cd ..` and press Enter. Yes, that is two dots or periods (..) after the `cd` command. Don't leave them out or this will not work! The two dots or periods (..) are a relative directory reference meaning "one directory level up." This is sometimes called the "parent directory."
27. See where you are by typing `pwd` and pressing Enter.
28. Now try something a little different: use just one dot (or period) with the `cd` command. Type `cd .` and press Enter. Yes, that is one dot or period (.) after the `cd` command. Don't leave it out or this command will not work!
29. See where you are by typing `pwd` and pressing Enter. You stayed in the exact same place! This is because the single dot or period (.) is a relative directory reference meaning "current directory."
30. Now navigate to your home directory using another relative directory reference. Type `cd $HOME` and press Enter. Be sure to use the correct case (Yes, $HOME is all upper-case letters) or this command will not work correctly. $HOME is an environment variable that points to your current home directory.
31. Check that you are in your home directory by typing in `pwd` and pressing Enter.
32. Now try to navigate up two directory levels using a relative directory reference. Type `cd ../..` and press Enter.
33. Type `pwd` and press Enter to see where your present working directory is currently located.
34. Use the `cd` command to navigate to your account's home directory using one of the four methods explored in this exercise.
35. If you need to stop the logging, type `exit` and press Enter, or press the key combination CTRL+D.

# Exercise 2

### Creating and Using New Subdirectories

1. Resume logging if needed by typing `script -a myLogFile.log`. Make sure you include the `-a` flag to avoid overwriting your previous exercise.
2. At the command prompt, type `cd` and then `pwd`. The `pwd` command shows your present working directory (in this case, it is your home directory).
3. Type `ls -F` and press Enter. You should see several file names with a / at the end of their names, indicating that they are subdirectories.
4. Using the home directory you recorded for *HomeDirectory* in this command, type in `mkdir` *HomeDirectory*`/NewDir` and press Enter. This will create a new subdirectory in HomeDirectory called NewDir. You are using an absolute directory reference with the `mkdir` command to create the new subdirectory.
5. Type `ls -F` and press Enter. You should see your newly created subdirectory.
6. Using the home directory you recorded for *HomeDirectory* in this command, type `cd` *HomeDirectory*`/NewDir` and press Enter. This will change your present working directory to the new subdirectory. Notice that you used an absolute directory reference with the `cd` command.
7. Type `ls -F` and press Enter. You should not see any files in the new subdirectory you just created. Why? You haven't put any files in there yet!

Exercise 2 continued on next page... 

8. Type `touch MyFile` and press Enter. The `touch` command either creates an empty file or updates the access and modification time stamps of an existing file. In this case, you just created a blank empty file called MyFile.

9. Type `ls -F` and press Enter. You should now see the file you just created, called MyFile in the new subdirectory.

10. Using the home directory you recorded in step for *HomeDirectory* in this command, type in `ls -F` *`HomeDirectory`*`/NewDir` and press Enter. You should see the same file listed as in the preceding step. In the preceding step, you looked at the files in your present working directory. In this step, you used an absolute directory reference to see files listed there.

11. Type `cd ..` and press Enter. (Don't miss those two periods after the cd command, or this step won't work.) This will move your present working directory from the new subdirectory up one level to your *HomeDirectory*.

12. Check that you are back at your *HomeDirectory* by typing `pwd` and pressing Enter. You should see the same directory that you recorded. If you don't, use your `cd` command to put your present working directory back to *HomeDirectory*.

13. *Warning:* This next command will generate an error. At the command prompt, type `mkdir Stuff/MoreStuff` and press Enter. You should get an error message similar to: `mkdir: cannot create directory 'Stuff/MoreStuff': no such file or directory.` This `mkdir` command attempts to create a subdirectory called Stuff and another subdirectory within Stuff called MoreStuff. However, a subdirectory cannot be created if a parent directory does not exist. The `mkdir` command knows the Stuff subdirectory does not exist yet, so it refuses to create anything.

14. Type `mkdir -p Stuff/MoreStuff` and press Enter. The `{p` option forces the `mkdir` command to create both the Stuff subdirectory and the MoreStuff subdirectory. Notice that you used a relative directory reference to create the *HomeDirectory*/Stuff/MoreStuff subdirectories.

15. Type `ls -RF Stuff` and press Enter. The MoreStuff subdirectory should display.

16. *Warning:* The next command will generate an error. Type `rmdir Stuff` and press Enter. You should get an error message similar to `rmdir: failed to remove 'Stuff': Directory not empty.` This `rmdir` command attempts to delete a subdirectory. However, if the directory contains subdirectories and/or files, it will not work, as demonstrated here.

17. Type `rmdir Stuff/MoreStuff` and press Enter. You got no error messages! Why? Because the subdirectory MoreStuff is empty. Notice you used a relative directory reference to remove the subdirectory MoreStuff.

18. Type `rmdir Stuff` and press Enter. You got no error messages, because the subdirectory Stuff is now empty (you deleted the subdirectory contents in the preceding step.)

19. Next you will attempt to use the `rmdir` command to delete the NewDir subdirectory you created earlier. Do you think this will work? Using the home directory you recorded for *HomeDirectory* in this command, type `rmdir` *`HomeDirectory`*`/NewDir` and press Enter. It did not work. Do you know why?

20. Remove both the MyFile file from NewDir and the directory by typing the command `rm -iR` *`HomeDirectory`*`/NewDir` and pressing Enter. When the command asks something similar to "descend into directory?" type y and press Enter. Type y and press Enter for each item it asks to remove. This will remove the NewDir subdirectory and all its contents.

21. Type `ls -F` and press Enter. Is the NewDir subdirectory gone?

Exercise 2 continued on next page... 

22. If you need to end logging, type `exit` and press Enter or press the key combination CTRL+D.

# Exercise 3

### Copying Files

1. Resume logging if needed by typing `script -a myLogFile.log`. Make sure you include the `-a` flag to avoid overwriting your previous exercise.
2. At the command prompt, type `cd` and then `pwd`. The `pwd` command shows your present working directory (in this case, it is your home directory).
3. Type `touch chores` and press Enter. This will create an empty file in your present working directory (pwd) called chores.
4. Type `ls chores` and press Enter. You should see the chores file listed.
5. Type `cp chores chores.bck` and press Enter. This will create a copy of the file chores and give it a new name, chores.bck.
6. Press Up Arrow twice to reverse through your command history until you reach `ls chores`. Type `*` so the command becomes `ls chores*`. Press Enter and you should see the files chores and chores.bck.
7. Type `mkdir Todo` and press Enter. This will create a subdirectory in your pwd called Todo.
8. Type `ls -F` and press Enter. You should see the subdirectory Todo.
9. Using the tilde (∼) home directory shortcut method, type `cp chores ∼/Todo/` and press Enter. This will copy the file chores into the subdirectory you just created, Todo.
10. Press Up Arrow twice to reverse through your command history until you reach `ls -F`. Type `Todo` so the command becomes `ls -F Todo`. Press Enter and you should now see a copy of the file chores in the subdirectory Todo.
11. Type `cp chores Todo/done` and press Enter. This will copy the file chores from your pwd to the subdirectory, Todo, and rename the file to done.
12. Press Up Arrow twice to reverse through your command history until you reach `ls -F Todo`. Press Enter and the file done should be in the Todo subdirectory as well as the file chores.
13. Type `cp -R Todo NewTodo` and press Enter. This `cp` command with the `-R` (recursive) option will copy an entire subdirectory and all of its contents to a new subdirectory. In this case, you are copying the Todo subdirectory and all of its files to a new subdirectory called NewTodo.
14. Press Up Arrow twice to reverse through your command history until you reach `ls -F Todo`. Press Enter and note the files located here.
15. Type ls -F NewTodo and press Enter. Note the files located here. Do they match the files in the preceding step? They should, because you copied them over into this new subdirectory in an earlier step.
16. Type `rm -Ri NewTodo` and press Enter. Type `y` and press Enter to all the questions asking if it is okay to descend into the directory, delete the files, etc. Remember, the `rm` command with the `-R` (recursive) option descends down into the subdirectory to delete all the files, and the `-i` (interactive) option asks you if it is okay to delete the files/directories.

Exercise 3 continued on next page... 

17. Type rm -Ri Todo and press Enter. Type `y` and press Enter to all the questions asking if it is okay to descend into the directory, delete the files, etc.
18. Type `rm -i chores*` and press Enter. Type `y` and press Enter to the question asking if it is okay to delete the files.
19. If you need to end logging, type `exit` and press Enter or press the key combination CTRL+D.

# Exercise 4

**Moving and Renaming Files**

1. Resume logging if needed by typing `script -a myLogFile.log`. Make sure you include the `-a` flag to avoid overwriting your previous exercise.
2. At the command prompt, type `cd` and then `pwd`. The `pwd` command shows your present working directory (in this case, it is your home directory).
3. Type `touch math` and press Enter. This will create an empty file called math.
4. Type `ls math` and press Enter. You should see the file math located in your pwd.
5. Type `mv math algebra` and press Enter. This will rename the file math to algebra.
6. Type `ls -F` and press Enter. The file named math should not be shown, because it has been renamed (using the `mv` command) to algebra. Of the two files, only the file named algebra should be shown.
7. Type `mv algebra gym` and press Enter. This will rename the file algebra to gym.
8. Press Up Arrow twice to reverse through your command history until you reach `ls -F`. Press Enter. The file named algebra should not be shown, because it has been renamed (using the `mv` command) to gym. Of the two files, only the file named gym should be shown.
9. Type `mkdir School` and press Enter. This will create the subdirectory School in your pwd.
10. Using the tilde (∼) home directory shortcut method, type `mv gym ∼/School/` and press Enter. This will move the file gym to the subdirectory School.
11. Type `ls -F` and press Enter. This will list all the files and subdirectories in your pwd. The file gym should be gone, because you moved it into the subdirectory School.
12. Press Up Arrow once to reverse through your command history until you reach `ls -F`. Type `School` so the command becomes `ls -F School`. Press Enter. The file gym should now be showing, because you are listing out the contents of the subdirectory School where you moved the file gym to.
13. Type `touch lunch` and press Enter. This creates a blank empty file called lunch in your pwd.
14. Type `mv lunch School/` and press Enter. This does NOT rename the file lunch to School, because School is a subdirectory in your pwd. Instead, this moves the file lunch to the subdirectory School.
15. Type `touch reading` and press Enter. This creates a blank empty file called reading in your pwd.
16. Type `mv reading School/history` and press Enter. This step moves the file reading to the sudirectory School and then renames it to history.

Exercise 4 continued on next page... 

17. Using the tilde (∼) home directory shortcut method, type `ls ∼/School` and press Enter. You should see three files located in the subdirectory School and they should be: gym, history, and lunch.
18. Type `mv School College` and press Enter. This renames an entire subdirectory named School to College.
19. Type `ls -F College` and press Enter. You should see that the three files gym, history, and lunch are now located in the subdirectory College.
20. Type `ls -F School` and press Enter. You should get an error message on this step. That is because this subdirectory no longer exists. You renamed School to College in a preceding step—therefore the subdirectory School no longer exists.
21. Type `rm -Ri College` and press Enter. Type `y` and press Enter to all the questions asking if it is okay to descend into the directory and/or delete the files. Remember, use the `rm` command with the `{R` option to descend down into the subdirectory College to delete all the files, and the `{i` option to ask you if it is okay to delete the files/directories.
22. End logging this exercise by typing `exit` and press Enter or press the key combination CTRL+D.

# Exercise 5

**Exploring Case Sensitivity and File Globbing**

1. Resume logging if needed by typing `script -a myLogFile.log`. Make sure you include the `-a` flag to avoid overwriting your previous exercise.
2. At the command prompt, type `cd` and then `pwd`. The `pwd` command shows your present working directory (in this case, it is your home directory).
3. Type `touch fall.dat` and press Enter. The "ll" in the file name is two lowercase L's (not number ones). This will create an empty file called fall.dat.
4. Press Up Arrow once to reverse through your command history until you reach `touch fall.dat`. Press Left Arrow until you reach the letter `a`. Replace `a` with `e` so the command becomes `touch fell.dat`. Press Enter to create an empty file called fell.dat.
5. Press Up Arrow once to reverse through your command history until you reach `touch fell.dat`. Press Left Arrow until you reach the letter `e`. Replace `e` with {i,u} using curly braces around the letters so the command becomes `touch f{i,u}ll.dat`. Press Enter to create two empty files called fill.dat and full.dat at the same time!
6. Double check that all four of the files were created. Type `ls -F` and press Enter. If you do not see the four files, you will need to add the missing file(s). Otherwise, this exercise will not work properly.
7. To view only the four files, you will need to use a wildcard symbol. Type `ls f*.dat` and press Enter. By using a wildcard symbol, such as the asterisk (*), you are creating what is called a "filter." Using a filter along with commands (such as `ls`) allows you to "filter out" files or directories you do not wish to perform the command upon. The asterisk (*) used as a filter in this command says, "Only show me the files that start with an f and end with .dat, but have any string in between."
8. There is another way to view only the four files; you will need to use a different wildcard symbol. Type `ls f?ll.dat` and press Enter. The question mark (?) can also be used as a wildcard character. However, unlike the asterisk, it is used to wildcard only one character. Used as a filter in this command, the question mark matches files with any character at the second character position in the file name.

Exercise 5 continued on next page... 

9. Another way to view only the four files is to use brackets. Press Up Arrow once to reverse through your command history until you reach `ls f?ll.dat`. Press Left Arrow until you reach the question mark (`?`). Replace `?` with `[a-z]` so the command becomes `ls f[a-z]ll.dat`. Press Enter and you should again see these four files. The brackets represent a single character position and give you multiple options for wildcarding. In this case, the brackets were used along with a range of potential choices for a single character in that position. The range was any character from a to z.

10. You can also use the brackets to narrow the choices (filter) of files. Press Up Arrow once to reverse through your command history until you reach `ls f[a-z]ll.dat`. Press Left Arrow until you reach the letter `z`. Replace `-z` with `e` so the command becomes `ls f[ae]ll.dat`. Press Enter. Here you only found files that had either an a or an e in the single character position.

11. You can also use curly braces to specify a comma separated list (filter) of files. Press Up Arrow once to reverse through your command history until you reach `ls f[ae]ll.dat`. Press Left Arrow until you reach the right bracket (`]`). Replace `[ae]` with `{a,u}` so the command becomes `ls f{a,u}ll.dat`. Press Enter. Here you only found files that had either an a or u in the single character position.

12. Just like commands, file and directory names are case sensitive. Try this out by first using a command that will work. Press Up Arrow once to reverse through your command history until you reach `ls f{a,u}ll.dat`. Press Left Arrow until you reach the right curly brace (}). Replace `{a,u}` with just `a` so the command becomes `ls fall.dat`. Press Enter and you should see only the fall.dat file listed.

13. Now try a command that won't work and shows how file names are case sensitive. Press Up Arrow once to reverse through your command history until you reach `ls fall.dat`. Press Left Arrow until you reach the letter `f`. Replace `f` with the capital letter `F` so the command becomes `ls Fall.dat`. Press Enter and you should not see this file listed. This is because you used a capital F in the file name instead of a lowercase f. File and directory names are case sensitive. In other words, Linux treats fall.dat and Fall.dat as two different and distinct files.

14. Now try to delete the files you created in this exercise, using file globbing. Type `rm -i f[aeiu]ll.dat` and press Enter. Type `y` and press Enter to all the questions asking if it is okay to delete the files. Notice that it is okay to squish together the various selections for the single character position. (Yes, you could have just used an asterisk instead, but this helps to reinforce a new concept.)

15. End logging this exercise by typing `exit` and press Enter or press the key combination CTRL+D.