Unit 3 Lab - File and Directory Operations Estimated Time : 1Hr. 30 Min.

Setup: A working, installed Linux system with an unprivileged user account with user name 'student'

Exercise 1: Directory and file organization

Scenario/ Story: A bunch of files have accumulated in your home directory, and you have decided that it is time to organize things. You plan to create several new subdirectories, and to copy and move files around to fit into your new scheme. Additionally, you have several files that are not needed at all, which must be deleted.

Tasks:

1. Log in on terminal as user student with password
2. Immediately after logging into the system, you should be in your home directory. Verity this "print working directory" command. $ pwd
3. Check to see if you have any files in your home directory using each of the following commands
   a. $ ls b. $ ls –a c. $ ls –al

   Why do the first and second command return different numbers of files?
   What is the size of the largest file currently in your home directory as reported by the third command?
   Do you have any subdirectories in your home directory?
4. You will now use touch to create the files needed for experience. The details of how the expansion used in the following command works will be covered in a later unit. For now simply type the following exactly as you see it. Verify with your colleagues.
   a. $ touch {report,memo,graph}_{sep,oct,nov,dec}_{a,b,c}{1,2,3}
5. Use the ls command to examine the results of the last command. You should find that it created 108 new, empty files (no need to count) in your home directory. These files represent data files that you will use in the reminder of this sequence. If for some reason you do not see these files, ask the instructor for assistance; without these files, the reminder of this lab will not work.
6. In order to organize your files you must first create some new directories. Use mkdir to create some subdirectories directly your home directory:
   a. $ mkdir a_reports
   b. $ mkdir september october november december
7. Create some additional subdirectories inside one of your new directories using the following commands
   a. $ cd a_reports to change to the directory. Then
   b. $ mkdir 1 2 3 Use ls to verify that you 3 new directories named 1,2 and 3 under you're a_reports subdirectory.
8. Begin by moving all of the 'b' reports out of your home directory and grouping them by month. When working with wildcard patterns, it is a good idea to pre-verify the operation to ensure you are operating on the correct files. One way to do this is to replace your command with a harmless command using the intended wildcard pattern.
   a. $ cd
   b. $ ls –l *dec?b? You should see the 9'december',"b" files listed. Move one of them to the subdirectory:
   c. $ mv graph_dec_b1 december Now move the rest of them with :
   d. $ mv *dec?b? december
   e. List the contents of the december subdirectory to verify the move operation was successful
9. Move all of the remaining "b" reports into their respective directories:
   a. $ mv *nov?b? november
   b. $ mv *oct?b? october
   c. $ mv *sep?b? september

10. You will now collect the 'a' reports into their respective corresponding numbered directories. Notice the use of ~ as shorthand for 'your home directory'. The combination of the wildcard and the pattern specifies all the files that end in _a1 in your home directory.
    a. $ cd a_reports
    b. $ mv ~/*_a1 1/
    c. The 'september ' 'a1' files are old and no longer needed. Use echo to make sure you've created a pattern that matches only these files, then delete them, and verify that the other 'a1' files were moved properly:
        i. $ cd 1
        ii. $ echo *sep*
        iii. $ rm *sep*
        iv. $ ls
11. Move the final "a2" and "a3" reports into their respective directories. To make life interesting we'll move them from the current directory, using both relative and absolute pathnames. First, use pwd to identify the current directory
    a. $ pwd (/home/student/a_reports/1)
    b. Verify the pattern that references the "a2" files with echo, then move them using absolute pathnames:
        i. $ echo /home/student/*a2*
        ii. $ mv /home/student/*a2* /home/student/a_reports/2
    c. Ever though your current directory is /home/student/a_reports/1, you can move files from /home/student to /home/student/a_reports/2 because you specified the files from pathnames – in this case absolute pathnames
        i. Now move the "a3" files using relative path names. Again first verify the pattern references the correct files
            1. $ echo ../../*a3*
            2. $ mv ../../*a3* ../3
12. Return to your home directory, and use ls to verify that the only files remaining in this directory are the "c" files (ie. Graph_dec_c1, graph_dec_c2, ..)
13. The "c1" and "c2" report files for each month are important, and you want to make a backup of them in another directory:
    a. $ mkdir /tmp/archive
    b. $ cp report*[12] /tmp/archive/
    c. Additionally, all the report files for the month of december should be backed up to the /tmp/archive directory. Note the use of the –i option to have cp prompt before overwriting any files:
        i. $ cp –I report_dec* /tmp/archive/
        ii. cp: overwrite `/tmp/archive/report_dec_c1'? n
        iii. cp: overwrite `/tmp/archive/report_dec_c2'? n
14. Now that you have backed up the few "c" files that are important to you, you want to delete all of the files still remaining in your home directory. Examination of the remaining files reveals that the wildcard *c* will match of them. Why would you NOT want to execute the command rm *c*?
    (hint : try ls *c*
15. Delete the remaining "c" files in your home directory. Once more we'll use echo before issuing a destructive command.
    a. $ echo *c[1-3]
    b. $ rm *c[1-3]
    c. $ ls (a_reports december november october september)

A more organized home directory, with files placed into the appropriate sub-directories and some files backed up