Cabrillo College

Lab 5: Managing Files

The goal of this lab is to become proficient with system commands for copying, moving, renaming, creating and removing files within your home directory.

Preparation

Everything you need to do this lab can be found in the Lesson 6 materials at: http://simms-teach.com/cis90calendar.php. Review carefully all Lesson 6 slides, even those that may not have been covered in class.

Check the CIS 90 forum at: http://opus-ii.cis.cabrillo.edu/forum/ for any tips and updates related to this lab. The forum is also a good place to ask questions if you get stuck or help others.

Procedure

Log on to the Opus server so that you have a command line shell at your service. Be sure you are in your home directory to start this lab. We are going to reorganize the files in our home directory. This will involve making new subdirectories and moving files around. The questions asked during this procedure are for your clarification only. You will be graded on correctly performing the procedure. At the end of the lab you will submit your work using the submit command.

Part I - Making Directories

1. Display a listing of the files in your home directory using the **ls -F** command.

2. Now let's make some new directories using the **mkdir** command:
   - Make a new directory named *edits* for keeping our file edits using the following command:
     mkdir edits
   - View the new directory's contents using the **-a** option of the **ls** command. Do you see the two hidden files that were created with the directory?
   - You can make more than one new directory at a time by supplying two arguments to the **mkdir** command. Make two new directories, one called *docs* the other called *etc*
   - Verify that they were made in your home directory.

3. Now try to make a subdirectory in a directory that doesn't yet exist by entering the following command:
   mkdir class/labs
   What happens? Try the same command using the **-p** option:
   mkdir -p class/labs
   Verify that it worked this time.

4. Now that the *class* directory is made, you can make a second subdirectory called *tests*. Use the command: **mkdir class/tests**
   Notice how you did not need the **-p** option. Why not?

Part II - Renaming Files

Now that you have new subdirectories made, let's rename some of the existing directories. To do this, we use the **mv** command.

1. Let's rename the *Miscellaneous* directory to just plain old *misc*.
   mv Miscellaneous misc
   Notice how the syntax is: **mv** *oldname newname*
   Does it work?

2. Use the same command to change the name of *Poems* to *poems* with a lowercase p.

3. You can also rename a file using the same command. Unix doesn't distinguish between renaming directories and renaming ordinary files. Try renaming *proposal1* to *MarkTwain*. Did it work?

Part III - Moving Files

Now we will actually move some files from one directory to another, renaming some of them as we do it.

1. To move a file from one directory to another, we use the same syntax:
   mv *filename directory*
   The filename will be moved to the specified directory, keeping its same name. Let's try it: Use the **ls -i** command to look at the inode number of *MarkTwain*
   Now move that file to the *docs* directory we made earlier:
   mv MarkTwain docs

2. Change directory to the *docs* directory and look at the inode number of *MarkTwain*
   What do you notice? The inode number is the same; the file hasn't moved, only the name has been moved from one directory to another.

3. Now change back to your home directory, and let's move *proposal2* into the *docs* directory and change its name to *magna_carta* at the same time.
   mv proposal2 docs/magna_carta
   By specifying a new name after the destination directory, we can rename the file at the same time we move it.

4. See if you can do the same thing by moving the file, *timecal*, to your *bin* subdirectory and renaming it to *datecal*.
   (Make sure you are in your home directory.)

5. Move *proposal3* to the *docs* directory renaming it to *policy*.

6. Now let's put a couple commands together to accomplish the following task:
   - Change directories to your *poems* subdirectory.
   - Make a new subdirectory called *Anon*, (for Anonymous).
   - Move the three files, *ant*, *nursery*, and *twister* into that new directory.
   - Verify that you have done this, and change directory back to your home directory.

7. Good job! Did you try using one **mv** command for all three files?
   You can. The **mv** command has a third syntax:
   mv *file1 file2 file3 ... fileN directory*
   The last argument must be a directory name, and all the preceding files will be moved into that directory - keeping their original names. Let's try it. Move the files: *text.err, text.fxd, small_town*, and *spellk* into the *edits* directory using the command:
   mv text.* small_town spellk edits
   Check the results by listing the contents of the edits directory. Pretty fancy?

8. What happens when you move a file onto a file that already exists?
   Let's try it. Enter the command: **ls -l letter bigfile**
   Notice the sizes of the two files. (You may want to head them to remind you what they are.)

9. Now move *letter* onto *bigfile* by entering:
   mv letter bigfile
   Did you get any error message? Any kind of message?

10. Look at *bigfile* now. How big is it? What are its contents?
    Don't worry; we'll get *bigfile* back later.
    Move *bigfile* back to *letter*

11. Now try doing the following moves:
    - Move *what_am_i* to the *misc* directory, keeping the same name.
    - Move *mission* to the *etc* directory, changing its name to *motd*
    - Don't panic on this one, but try moving the file *better_town*, which is in the *misc* directory to the *edits* directory, keeping the same name.

Part IV - Copying Files

Copying works just like moving except that a second file is created, the contents of which is exactly like the original.

1. Copying files is useful for making backup copies. Let's make a backup of your *letter* file by typing the following command:
   cp letter letter.bak
   Notice that it has the same syntax as the mv command, but now you have two files.
   Run the **ls -li** command with arguments of *letter* and *letter.bak*
   Notice the sizes are the same, but they each have their own inode and their own data.

2. Another reason to use the **cp** command is to copy a file from somewhere else on the system to your directory. In this case, the filename stays the same, but the second argument is a new directory into which we put the file. Try it with:
   cp /etc/hosts etc
   You have just copied the file *hosts* from root's */etc* directory to your own personal *etc* directory. Verify that the file is in your *etc* directory.

3. Copy your graded labs into your class/labs directory with the command:
   cp lab0?.* class/labs
   What does the question mark (?) and asterisk (*) do?

4. Now use the **cp** command to copy the file *sonnet6* from /home*/cis90/depot* directory to your *Shakespeare* directory. (Notice that your *Shakespeare* directory doesn't have a *sonnet6* in it.)

5. Like the **mv** command, the **cp** command is also destructive. If the target file exists, it will be destroyed, and copied over. If you want the **cp** command to warn you about destroying a target file, then you must use the **-i** option. Try this in your home directory:
   cp -i letter letter.bak
   Notice how it asked you whether you wanted to overwrite the *letter.bak* file.

6. The **cp** command may also be used to copy multiple files to a single directory in the same way you used the **mv** command.
   More useful is copying an entire directory. Let's copy the entire *Shakespeare* directory to a directory called *Sonnets*.
   First, **cd** to your *poems* directory. Now issue the following command:
   cp -r Shakespeare Sonnets
   Verify that it worked using the command: **ls S***
   (the **-r** stands for recursive.)

Part V - Removing Files and Directories

Removing files is inherently destructive in nature. Unix does not give any warnings or opportunity to un-remove a file. So be careful in using this command.

1. Make sure you are back in your home directory. What command did you use?

2. The remove command is dangerous in it's simplicity. It takes one or more filenames as arguments. Try removing your *empty* file with the command:
   rm empty
   Try listing the file; it's gone - for good - no warning - poof.

3. If you would like the remove command to ask you if you're sure, use the **-i** option. Try:
   rm -i letter.bak
   answer with anything but a 'y' or 'yes' and it will not remove the file. Go ahead and remove it.

4. For some fun, change your directory to *Lab2.0* and run the **ls** command.
   - Now run the command: **rm *.***
   - What was removed? Is it like DOS?
   - Try removing the file *annual report*

- Do you see why spaces in file names is not a good idea? How could you remove that file?
- Finally, try this command: **rm -i ***
- Answer y to all questions. Are all the files gone? Change back to your home directory.

5. From your home directory, try to remove the directory, *Lab2.0*
   The remove command doesn't work on directory files. For that we have the **rmdir** command. Try it:
   rmdir Lab2.0
   Why didn't it work?

6. List the entire contents of the *Lab2.0* directory using the *ls -a* command.
   Remove the hidden file *.junk*, and then try the **rmdir** command again.

7. The lesson here is that **rmdir** removes only empty directories. There is a way to remove an entire directory and its contents. This is obviously a very dangerous command. From your home directory, run the following command:
   rm -r Lab2.1
   Notice that the directory and all its contents are gone, no warnings.
   Be careful when using this recursive option to the **rm** command.

8. Remove your *Sonnets* directory and its contents.

Part VI - Linking Files

Linking files is a way of giving additional names to an existing file. You are not duplicating data, just adding another name into your directory.

1. I told you I would get your *bigfile* back for you. The file can be found in the */home/cis90/depot* directory. Rather than copying it from there, let's do a hard link to it. Use the following commands from your home directory:
   ls -l ../depot/bigfile

2. Notice the link count field, the owner and the size of the file. Now enter:
   ln ../depot/bigfile .
   You have just made a link to *bigfile* in your home directory. Do a long listing of *bigfile*. What is the link count now? Who owns it?
   You now have access to rsimms' *bigfile*, as if it is in your directory.

3. Change directory to your *etc* directory and link the *motd* to the name *greeting*. Use the **ln** and the filenames as the two arguments. Use the **ls -li** command to verify your success.

Submittal

You have now finished Lab 5. Your home directory is now reorganized. To submit your work to be counted for this lab, you must run the **submit** command from your home directory. This will take a snapshot of your home directory and send it to your instructor.

Run **check5** to check your work and make sure you didn't forget anything. Run **verify** to doublecheck you submitted your lab for grading.

Grading Rubric (30 points total)

30 points for successfully completing all steps.
Less 1 point for each step not completed correctly.

Be sure to submit your work and extra credit before the deadline. **Remember, late work is not accepted.**

Extra credit (optional)

For two points extra credit complete:

In NETLAB+:
NISGTC Linux+ Series 1
Lab 7a: Using the BASH Shell
Send me a "signed" summary screenshot (see instructions below).

For one point extra credit complete:

In NETLAB+:
Red Hat Systems Administration - RH124
14.6. Practice: Making Links Between Files
Send me a "signed" summary screenshot (see instructions below).

Summary screenshot:

1) Before ending the lab maximize the terminal window you used.
2) Use: **history** to show the commands you issued during the lab.
3) Use: **echo "***firstname lastname* **was here"** as your signature.
4) Take a screen shot of the above and email it to: `risimms@cabrillo.edu`

Extra credit is due when the lab is due. **Remember, late work is not accepted.**