CPSC2800 Linux Hands-on Lab #5 Advanced File Processing

Project 5-1

The pipe operator directs the output of one command to the input of another. In Linux, this operator is very useful for combing commands on one line and yielding output that is easier to read or use. In this project, you use the pipe operator to direct the output of the ls command to the input of the more command so you can more easily view the contents of a large directory.

To redirect the output of the *ls* command to the more command:

1. Type **ls –l /etc** and press **Enter**. Notice that the output of the command scrolls by quickly.
2. Type **ls –l /etc | more** and press **Enter**. Record your output: _____
   _____
3. Notice the output fills the screen and pauses with the prompt "More" displayed on the bottom line. Each time you press the spacebar, the output advances to the next screen. Press the spacebar to scroll a screen at a time or press Enter to advance one line at a time until the command has finished. You also can type **q** at any point to exit the display of the directory contents.

Project 5-2

The pipe operator enables you to combine multiple commands on a single line. In this project, you pipe the contents of a directory listing into the sort command and then pipe the results into the more command.

To connect several commands with the pipe operator:

1. Type **ls /etc | sort –r | more** and press **Enter**. This command redirects the directory listing of the /etc directory to the sort –r command. sort –r sorts the directory listing in reverse order. The sort command's output is redirected to the more command.
2. After you execute the command, record your output: _____
   _____
3. Press the **spacebar** until the displayed output is finished.

Project 5-3

In this project, you use several features of the grep command and your learn to combine it with the head command for more manageable output. As you recall from lab#1, you can use the head command to retrieve the first 10 lines of a file. You can combine the grep and head command to retrieve only the first 10 lines containing the word or phrase. For example, here you use grep with head to find the first 10 lines in /etc/termcap that contain the characters "IBM."

To display lines in a file containing a particular word or phrase:

1. To see all the lines in the /etc/termcap file that contain the characters "IBM," type **grep ftp /etc/wgetrc**, and press **Enter**. Many lines fit the criteria, and the output scrolls by quickly. (if you do not have wgetrc file, use any available file to practice.)

2. Redirect the output of the grep command to the input of the more command. Type **grep ftp /etc/wgetrc | more** and press **Enter**. Record your output by screenshot: _____
3. Press the **spacebar** until the command output is finished.
4. Type **clear** and press **Enter** to clear the screen.
5. Redirect the output of the grep command to the head command. Type **grep ftp /etc/wgetrc | head** and press **Enter**.

The command that you typed in step 5 told grep to look for "IBM" in the /etc/termcap file, and then display the first 10 lines that are found. Record your output in step 5: _____

The grep command's options and wildcard support allow powerful search operations. In the next set of steps, you learn more about these options, such as performing searches on the basis of capitalization and by ignoring capitalization. You also learn to search using wildcard and metacharacter options to extend the range of your searching.

To expand the grep command's search capabilities through its options and regular expression support:

1. To see each line in the /etc/termcap file that contains the word "Linux," type **grep Linux /etc/services**, and press **Enter**.
2. Some lines in the file contain the word "linux". The search you performed in step 1 only displayed the lines that contain "Linux." The –i option tells grep to ignore the case of the search characters. Type **grep –i linux /etc/services** and press **Enter**. You see the lines that contain either "Linux" or "linux." Record your output: _____
3. Type **clear** and press **Enter** for better viewing of the next step.
4. The grep command supports regular expression characters in the search string. To see all the lines of the /etc/termcap file that start with "lin" followed by any set of characters, type **grep –i "^lin" /etc/services**, and press **Enter**. Record your output: _____
5. The grep command can process multiple files one after another. Type **grep linux /etc/*** and press **Enter**. You see the lines that contain "linux" from all the files in the /etc directory.
6. Type clear and press Enter.
7. The –l (lowercase L) option instructs grep to display only the names of the files that contain the search string. Type **grep –l linux /etc/*** and press **Enter**. You see the names of the files in the /etc directory that contain "linux." For what files is information displayed? _____

The grep command also searches files for phrases that contain spaces, as long as phrase is specified on the command line inside quotation marks. For example, grep can search for the phrase "IBM PC," as demonstrated in the next set of steps.

To search a file for a phrase:

1. Type **grep "FTP LINKS" /etc/wgetrc** and press **Enter**.

You see all lines in the /etc/termcap file that contain the phrase IBM PC.

2. Type **clear** and press **Enter**.
   In the previous examples, grep searches the file specified on the command line. Grep can also take its input from another command, through the pipe operator.

To redirect the output of a command to the grep command:

1. Type **ls /etc | grep** magic and press **Enter**. You see a list of the files whose names contain the word "magic."
2. Type **clear** and press **Enter** to clear the screen.

Project 5-4

This project illustrates common use of the uniq command. To perform the project, start by using the vi or other editor to create a new fill called zoo1 in your working directory. This is a simple data file example, containing variable-length records that list animal names, food descriptions, pounds eaten daily, and food costs. Type the duplicate records in step 1 as shown. After you create the file, use uniq command to remove the duplicate records.

To remove duplicate lines with the *uniq* command:

1. Type the following using vi or other editor and save as **zoo1**.
   Monkeys:Bananas:2000:850.00
   Lions:Raw Meat:4000:1245.50
   Lions:Raw Meat:4000:1245.50
   Camels:Vegetables:2300:564.75
   Elephants:Hay:120000:1105.75
   Elephants:Hay:120000:1105.75

2. To use uniq to remove duplicate lines from the zoo1 file and use the output redirection operator to create the new file zoo2, type **uniq zoo1>zoo2**, and press **Enter**.
3. Type **cat zoo2** and press **Enter** to see the contents of zoo2. Notice that the uniq command removed the duplication lines. Record your output: _____

Project 5-5

In this project your explore the *comm* command. You start by creating the file my_list. Next you duplicate the file, and then use the comm command to compare the two files.

To use the *comm* command to compare files:

1. To create the file my_list, type **cat > my_list** and press **Enter**.
2. Type the following text, pressing Enter at the end of each line:
   Football
   Basketball
   Skates
   Soccer ball

3. Press **Ctrl+d**.
4. To copy my_list to a second file, your_list, type **cp my_list your_list**, and press **Enter**.
5. Now use the comm command to compare my_list to your_list. Type **comm my_list your_list** and press **Enter**.
6. You see the three-column output. (Note that the text showing column headings is inserted for your reference. This text does not appear on your careen.) Notice that the lines in the third column are those that both files contain. (How your columns line up will vary depending on the operating system.) The files are identical. Record your output: _____
   _____
7. Now add a new line to my_list. Type **cat >> my_list** and press **Enter**.
8. Type **Golfball** and press **Enter**.
9. Press **Ctrl+d**.
10. Use comm to compare my_list to your_list again. Type the **comm my_list your_list** and press **Enter**.
11. You see the three-column output, with the unique new line in my_list in column 1. Record your output: _____

Project 5-6

In this project, you use the diff command to compare the contents of the zoo1 and zoo2 files you created previously.

To use *diff* to find differences between two files:

1. Review the contents of zoo1 and zoo2 by typing **more zoo1 zoo2** and pressing **Enter**. Press the **spacebar** to see the second file's contents.
2. Type **diff zoo1 zoo2** and press **Enter**.
3. Record your output: _____
   This means that you need to delete the third and sixth lines from zoo1 so the file matches zoo2. (Note that in some versions of Linux, you might see 5d4 instead of 3d2 because another way to match the files is to delete the fifth line in zoo1, which is the same as the sixth line.)
4. To reverse the comparison order, type **diff zoo2 zoo1**, and press **Enter**. What do you see? _____
   _____
   This means that you need to add the two lines shown in zoo2, so the file matches zoo1. You would add the third line of zoo1 to go after the second line in zoo2. And you would add the sixth line of zoo1 to go after the fourth line of zoo2.

Project 5-7

In this project, you use the *wc* command to count the number of lines in a new file called courters.
To create a file and count its lines:

1. Type **cat > counters** and press **Enter**.
2. Type this text, pressing Enter at the end of each line:
   Linux is a full featured Unix clone.

Linux is available in free and commercial versions.

3. Type **Ctrl+d**.
4. To find the number of lines in counters, type **wc -l counters**, and press **Enter**.
   Linux report that the file contains _____lines.
5. To find the number of bytes in counters, type **wc -c counters**, and press **Enter**.
   Linux report that the file contains _____bytes.
6. To find the number of words in counters, type **wc –w counters**, and press **Enter**. Linux report that the file contains _____words.
7. To count words, characters, and lines in counters, type **wc –lwc counters**, and press **Enter**. Linux reports : _____
   If you enter **wc counters**, you get the same output as entering **wc –lwc counters**.

Project 5-8

sed is stream editor that enables you to work on specific lines in a file and modify their contents. In this project, you use *sed* to display lines and edit a file that you create. The focus of this project is on using *sed* commands from the command line.

To use sed to manipulate a file:

1. Create the new file, **unix_stuff**, in your working directory by using the *vi* or other editor (such as gedit). The unix_stuff file should contain the following lines:
   Although UNIX supports other database systems,
   UNIX has never abandoned the idea of working with
   flat files. Flat files are those that are based on pure
   text with standard ASCII codes. Flat files
   can be read by any operating system.
2. To display only lines 3 and 4, type **sed -n 3,4p unix_stuff**, and press **Enter**. (The –n option prevents **sed** from displaying any lines except those specified with the p command.)
   This means “find lines numbered (-n) 3 and 4 in the file unix_stuff and display them (p).” What do you see? _____
3. In **sed**, you can place two commands on one line. If you want to delete lines 3 and 4 and then display the file, you must use the –e option to specify multiple commands on the same line. To delete lines 3 and 4 from unix_stuff and display the results, type **sed -n -e 3,4d -e p unix_stuff**, and press **Enter**. What do you see? _____
   Line 3 and 4 are not actually deleted from the file, but simply filtered out so that they are not displayed on the output to the screen.
4. To display only lines containing the word “Flat,” type **sed -n /Flat/p unix_stuff**, and press **Enter**. What do you see? _____
5. To replace all instances of the word “Flat” with “Text,” type **sed -n s/Flat/Text/p unix_stuff**, and press **Enter**. The s command substitutes one string of characters for another. What do you see? _____

Project 5-9

You continue working with the *sed* command in this project, so that you learn how to append lines from one file to another. First, you use the *vi* or other editor to create a new script file, *more_stuff*. You use the append command, *a*\, in the *more_stuff* file with the lines to be appended by *sed* to the file *unix_stuff*. (you could accomplish the same outcome by using *cat more_stuff >> unix_stuff*, but the purpose here is to show you the versatility of *sed*.) You must terminate each line, except for the final line of the file being added, with a backslash character. In this project the *$* preceding the *a*\ tells *sed* to append *more_stuff* to *unix_stuff* after the final line in *unix_stuff*; without *$*, *sed* repeatedly adds all the lines in *more_stuff* after each line in *unix_stuff*.

To create a script file to append lines to another file using sed:

1. Use vi or other editor to create the script file **more_stuff** to have the following lines:
   $a
   Informix and Oracle, two major relational database
   companies have installed their RDBMS package on UNIX
   systems for many years.
2. To use the *sed* command to run the script file, type **sed -f more_stuff unix_stuff** and press **Enter**. What do you see? _____
3. Use *vi* or other editor to create the file **stuff_replace**.
   s/UNIX/Linux/
   s/abandoned/given up/
   s/standard/regular/
   The lines in this file instruct *sed* to replace all occurrences of "UNIX" with "Linux", "abandoned" with "given up", and "standard" with "regular".
4. Type **clear** and press **Enter**.
5. Execute *sed*, with the script file you crated in step 3, on the unix_stuff file. Redirect sed's output to the file unix_stuff2. Type **sed -f stuff_replace unix_stuff > unix_stuff2** and press **Enter**.
6. Type **cat unix_stuff2** and press **Enter**. What do you see? _____

Include your experiences and answers to all the underlying parts in your report. Include the following at the beginning of your report.

- Name: _____
- UTC ID: _____
- Course Number and Name: _____
- Semester: _____
- Lab Name and Number: _____
- I spent _____hours and _____minutes to finish this hands-on lab.
- I have _____ (percent) finish this lab.
- I expect _____ (A, B, C, or F) of this lab.

- This lab helps me to master Linux Operating System, environment, and commands. Choose a number to indicate how much the lab is helpful.

  1 2 3 4 5

  (less helpful) (more helpful)