**CSCD 240**

**Lab 34**

**Requirements:** all answers are to be produced by using ssh to login to **cslinux.eastern.ewu.edu** to complete this assignment.When needed, you should capture the prompt, the command, and the output from the command.

1. Clearly explain the difference between **which, grep,** and **find**. (No need to capture for this question)
2. Issue the find command looking for the file named **ld** starting at the root directory.
   1. Assuming you are not logged in as root, you should get a list of errors as well as where the file was found. Capture the output and include it in your submission – you do not need to include all the permission errors just a few to get the idea but do include where the file was found.
   2. Repeat the command (again not as root) – illustrating a method of eliminating the error messages on the standard output and printing only what was found.
3. Find all files (not folders) in your home directory and its subdirectories, with size greater than 100 bytes.
4. In class we talked about the '–size' and ‘–name’ option for the **find** command.
   1. Issue and capture the results of the find command in your home directory that display all files that are greater than 1K. Do not display error messages.
   2. Explain this command: **find . -name "\*.txt" -exec wc -l {} ‘;’**
   3. Explain this command: **find . -name "\*.txt" -exec rm {} ‘;’**
5. Use a text editor on the remote machine to create a file named **frost.poem** in your home directory that contains the following text:

The Road Not Taken by Robert Frost

Two roads diverged in a yellow wood,

And sorry I could not travel both

and be one traveler, long I stood

And looked down one as far as I could

To where it bent in the undergrowth;

Then took the other, as just as fair,

And having perhaps the better claim

Because it was grassy and wanted wear,

Though as for that the passing there

Had worn them really about the same,

* 1. Use the **grep** command, capture both the command and the output, to finds all lines, including the line number, that end with a **comma ( , )**
  2. Use the **grep** command, capture both the command and the output, to finds all lines, including the line number, containing the word **as**
  3. Use the **grep** command, capture both the command and the output, to finds all lines, including the line number that starts with the word **and** (case DOES NOT matter).
  4. Use the **grep** command, capture both the command and the output, to finds all lines, including the line number that starts with the word **and** (case DOES matter).

1. Capture, creating a directory named **lab3 in your home directory**.
   1. Capture placing a copy of frost.poem in the directory **lab3**. There should be one copy of frost.poem in your home directory and one in **lab3**.
   2. Within your home directory, capture the **grep** command and its output that will ***recursively*** find all instances of the word **I** (case DOES matter) in all files that end with **.poem**.
2. Using a text editor create a file named **myScript** that contains the following:

#!/bin/bash  
string=”Hello World”  
echo $string

* 1. Try to execute the script with **./myScript** and capture the output. What will you see?
  2. Execute and capture the command that will change the permissions on myScript to be user executable without changing any other permissions.
  3. Execute the script with ./myScript and capture the output.

1. Capture the output of the command **printenv** and redirect the output into a file named penvout.txt.
2. Capture the output of the command **env** and redirect the output into a file named envout.txt
3. Capture the **diff** command, on envout.txt and penvout.txt. (man **diff** to find information about diff)
4. In the lab3 directory create the C file named lab3.c with the following code.

#include <stdio.h>

int main()

{

printf(“Hello World\n”);

return 0;

}// end main

1. Give the grep command that will start in your home directory and show the file names and line numbers containing the term “stdio” in all .c files in the home directory and all directories below the home.
2. Consider the following command ls – al | less.
   1. What does the command do?
   2. How does this command show advantages over **ls –al**?
3. Redirect the output of the command **ls –lah ~/.bashrc** to a file named **details.txt**.
4. Write a single command that can redirect the output of command **ls –l /bin/cp** to the existing file **details.txt** generated in question 14**, with** new contents to be appendedwithout overwriting the file.
5. Provide a single command that can count how many lines of text are contained in **details.txt** created in question 14.
6. Capture the command(s) used to send a running job to background without terminating it and restarting. (three steps in lecture notes, you can use the command **cp /dev/zero /dev/null** for demo purpose)
7. Define what a process is and what a job is, clearly explain how jobs differ from processes.
8. Consider the following command ls – al | less.
   1. How many processes are created with that command? ( **one for ls and one for less )**
   2. What exactly does “|” do in this command?
9. Write a **single** command that can check whether a program named **sshd** is running or notin the whole system. (hint: use ps together with pipe and grep)
10. Using the man page for **ps**
    1. Issue and capture the **ps** command with the appropriate options to allow listing of all processes in the system.
    2. Using the output from part **a**, what was the first process started and by whom was it started?
    3. What was the first non-root process that was started?
    4. What was the last process started and by whom?
11. Using a text editor create the following C programmed named almostEndless.c in your current directory. Please type in all statements carefully.

#include <stdio.h>

int main()

{

int x = 0;

while(x < 2000000)

{

printf("..");

fflush(stdout);

sleep(3);

x ++;

}// end while

return 0;

}// end main

* 1. Compile your program with **gcc almostEndless.c**
  2. Run your program with **./a.out**
  3. With your program running, describe the commands you would use (without using ctrl-c) to terminate that program, from the same terminal window in which it was started.
  4. Execute and capture the commands, using process notation (PID), to terminate a.out
  5. Restart your program with **./a.out**
  6. Execute and capture the commands, using job notation (job ID), to terminate a.out

1. In a single terminal window capture the command ( or commands) to start a.out 3 times each running as background jobs: **(This is tricky. Hint**: **IO redirection to redirect standard output** is useful here.)
   1. What are the job numbers of the above?
   2. What are the process ID numbers of the above?
   3. Capture the command and output to bring the second a.out to the foreground.
   4. Capture the command(s) to send a.out back to the background.
   5. Capture the command(s) to kill the third a.out using its job number.
   6. Capture the command(s) to kill the first a.out using its process number.
   7. Can ctrl-c be used to kill any job? Why or why not? Clearly explain why or why not. **Hint: try to use ctrl-c for a background job.**
2. What will be printed in each below command, after command **var=`date`** has been executed? Clearly explain why you see that output for each command below.
3. **echo “$var”**
4. **echo ‘$var’**
5. **echo var**

**TO TURN IN:**

* A PDF file - Name this text file your last name, first letter of your first name lab34.pdf. This file will contain all your answers.
* **I want the question copied and then the answer to the question below it.**
* A zip file that contains your PDF, all files created during this lab ( C files, scripts, IO directions etc).

You zip will be named your last name first letter of your first name lab34.zip. Turn in your zip file on Canvas 2014 cscd240🡪 Assignments🡪Lab34🡪submit.