Tim Tanasse
CSCD 240
April 15, 2015
Lab # 34

- **1) Clearly explain the difference between which, grep, and find. (No need to capture for this question)**
  - **which** - Gives you the path of the command you give it as an argument
  - **grep** - Search tool that searches in the text of the files for a given pattern
  - **find** - looks for files/folders that fit a certain name or pattern
- **2) Issue the find command looking for the file named ld starting at the root directory.**
  - **a. 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.**

    ttanasse1@cslinux: $ find / -name ld
    find: '/run/samba/winbindd_privileged': Permission denied
    find: '/etc/ppp/peers': Permission denied
    find: '/etc/chatscripts': Permission denied
    find: '/etc/ssl/private': Permission denied
    find: '/.ssh': Permission denied
    find: '/root': Permission denied
    /usr/lib/compat-ld/ld
    /usr/lib/gold-ld/ld
    /usr/bin/ld
    /usr/share/doc/binutils/ld

  - **b. 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.**

    ttanasse1@cslinux: $ find / -name ld 2> /dev/null
    /usr/lib/compat-ld/ld
    /usr/lib/gold-ld/ld
    /usr/bin/ld
    /usr/share/doc/binutils/ld
- **3) Find all files (not folders) in your home directory and its subdirectories, with size greater than 100 bytes.**

  ttanasse1@cslinux: $ find -type f -size +100c
  /home/EASTERN/ttanasse1/.bash_history
  /home/EASTERN/ttanasse1/cscd240/lab1/my.copy.bashrc
  /home/EASTERN/ttanasse1/cscd240/lab1/lab1.copy/my.copy.bashrc
  /home/EASTERN/ttanasse1/cscd240/lab1.copy/my.copy.bashrc
  /home/EASTERN/ttanasse1/cscd240/lab1.copy/lab1.copy/my.copy.bashrc
  /home/EASTERN/ttanasse1/.PyCharm40/config/pycharm30.key
  /home/EASTERN/ttanasse1/date.txt
  /home/EASTERN/ttanasse1/.bashrc
  /home/EASTERN/ttanasse1/.profile
  /home/EASTERN/ttanasse1/.bash_logout

- **4) In class we talked about the 'size' and name option for the find command. c. Explain this command: find . -name "*.txt" -exec rm ;**
  - **a. 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.**

    ttanasse1@cslinux: $ find -type f -size +1k 2> /dev/null
    /home/EASTERN/ttanasse1/.bash_history
    /home/EASTERN/ttanasse1/cscd240/lab1/my.copy.bashrc
    /home/EASTERN/ttanasse1/cscd240/lab1/lab1.copy/my.copy.bashrc
    /home/EASTERN/ttanasse1/cscd240/lab1.copy/my.copy.bashrc
    /home/EASTERN/ttanasse1/cscd240/lab1.copy/lab1.copy/my.copy.bashrc
    /home/EASTERN/ttanasse1/date.txt
    /home/EASTERN/ttanasse1/.bashrc
  - **b. Explain this command: find . -name "*.txt" -exec wc -l ;**
    Counts the number of lines in txt files found in the current directory and all subdirectories.
  - **c. Explain this command: find . -name "*.txt" -exec rm ;**
    Removes the txt files found in the current directory and all subdirectories.
- **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:**
  **Frost Poem Here**
  - **a. Use the grep command, capture both the command and the output, to finds all lines, including the line number, that end with a comma ( , )**

    ttanasse1@cslinux: $ grep ".*,$" frost.poem
    Two roads diverged in a yellow wood,
    Then took the other, as just as fair,
    Because it was grassy and wanted wear,
    Had worn them really about the same,
  - **b. Use the grep command, capture both the command and the output, to finds all lines, including the line number, containing the word 'as'**

    ttanasse1@cslinux: $ grep " as " frost.poem
    And looked down ome as far as I could
    Then took the other, as just as fair,
    Though as for that the passing there
  - **c. 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).**

    ttanasse1@cslinux: $ grep -i "ând \— and " frost.poem
    And sorry I could not travel both
    and be one traveler, long I stood
    And looked down ome as far as I could
    And having perhaps the better claim
    Because it was grassy and wanted wear,
  - **d. 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).**

    ttanasse1@cslinux: $ grep -n "ând \— and " frost.poem
    4:and be one traveler, long I stood
    10:Because it was grassy and wanted wear,

- **6) Capture, creating a directory named lab3 in your home directory.**
  - **a. 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.**

    ttanasse1@cslinux: $ mkdir lab3
    ttanasse1@cslinux: $ cp frost.poem lab3
  - **b. 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'.**

    ttanasse1@cslinux: $ grep -r –include "*.poem" "Î \— I " .
    ./frost.poem:And sorry I could not travel both
    ./frost.poem:and be one traveler, long I stood
    ./frost.poem:And looked down ome as far as I could
    ./lab3/copy.frost.poem:And sorry I could not travel both
    ./lab3/copy.frost.poem:and be one traveler, long I stood
    ./lab3/copy.frost.poem:And looked down ome as far as I could

- **7) Using a text editor create a file named myScript that contains the following: #!/bin/bash**
  **string=Hello World**
  **echo $string**
  - **a. Try to execute the script with ./myScript and capture the output. What will you see?**

    ttanasse1@cslinux: /BashScripts$ ./myScript
    -bash: ./myScript: Permission denied
  - **b. Execute and capture the command that will change the permissions on myScript to be user executable without changing any other permissions.**

    ttanasse1@cslinux: /BashScripts$ chmod u+x myScript
  - **c. Execute the script with ./myScript and capture the output.**

    ttanasse1@cslinux: /BashScripts$ ./myScript
    Hello World

- **8, 9, 10) Capture the output of the command printenv and redirect the output into a file named penvout.txt.**
  **9) Capture the output of the command env and redirect the output into a file named envout.txt**
  **10) Capture the diff command, on envout.txt and penvout.txt. (man diff to find information about diff)**

  ttanasse1@cslinux: /BashScripts$ printenv > penvout.txt
  ttanasse1@cslinux: /BashScripts$ env > envout.txt
  ttanasse1@cslinux: /BashScripts$ diff penvout.txt envout.txt
  18c18
  ¡ _=/usr/bin/printenv
  —
  ¿ _=/usr/bin/env

- **11) 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
```

- **12) 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.**

  ttanasse1@cslinux: $ grep -rn –include "*.c" "stdio" .
  ./lab3/lab3.c:1:#include ¡stdio.h¿

- **13) Consider the following command ls  al — less.**

  - **a. What does the command do?**
    pipes the output from **ls -al** from the stdout as an input to less.

  - **b. How does this command show advantages over ls al?**
    Since **ls -al** can end up printing a lot of stuff, it allows keyboard based and often more controlable scrolling by window or by line.

- **14) Redirect the output of the command ls lah  /.bashrc to a file named details.txt.**

  ttanasse1@cslinux: $ ls -lah  /.bashrc > details.txt

- **15) 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 appended without overwriting the file.**

  ttanasse1@cslinux: $ ls -l /bin/cp >> details.txt

- **16) Provide a single command that can count how many lines of text are contained in details.txt created in question 14.**

  ttanasse1@cslinux: $ wc -l details.txt
  2 details.txt

- **17) 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)**

  ttanasse1@cslinux: $ cp /dev/zero /dev/null
  Ẑ
  [2]+ Stopped cp /dev/zero /dev/null
  ttanasse1@cslinux: $ bg %2
  [2]+ cp /dev/zero /dev/null &

- **18) Define what a process is and what a job is, clearly explain how jobs differ from processes.**
  A process is an instance of a executable program that is currently executing. A job is an execution of a program from the shell environment which is attached to that instance of the shell environment. A regular process runs independant of any shell, while a job will end when the shell ends.

- **19) Consider the following command ls  al — less.**
  - **a. How many processes are created with that command?**
    2 processes are started, one for **ls** and one for **less**.
  - **b. What exactly does — do in this command?**
    '—' is the pipe character, in this case it will take the output of **ls -al** and use it as a file for the input into **less**.
- **20) Write a single command that can check whether a program named sshd is running or not in the whole system. (hint: use ps together with pipe and grep)**

  ttanasse1@cslinux: $ ps aux — grep -c "sshd"
  78
- **21) Using the man page for ps**
  - **a. Issue and capture the ps command with the appropriate options to allow listing of all processes in the system.**

    ttanasse1@cslinux: $ ps -ef
  - **b. Using the output from part a, what was the first process started and by whom was it started?**

    ttanasse1@cslinux: $ ps -ef
    UID PID PPID C STIME TTY TIME CMD
    root 1 0 0 Apr13 ? 00:00:02 /sbin/init

    /sbin/init, started by root.
  - **c. What was the first non-root process that was started?**

    ttanasse1@cslinux: $ ps -ef — grep -v "root "
    UID PID PPID C STIME TTY TIME CMD
    102 621 1 0 Apr13 ? 00:00:00 dbus-daemon –system –fork –activation=upstart

    dbus-daemon
  - **d. What was the last process started and by whom?**

    apeter18 30952 30950 0 19:29 ? 00:00:00 /usr/lib/openssh/sftp-server

    sftp-server, started by apeter18
- **22) Using a text editor create the following C programmed named almostEndless.c in your current directory. Please type in all statements carefully.**

```c
#include <stdio.h>

int main()
{
        int x = 0;
        while(x < 2000000)
        {
                printf("..");
                fflush(stdout);
                sleep(3);
                x ++;
        }// end while

        return 0;
}// end main
```

- **c. 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.**
  use ctrl-z to suspend the job, then us ps to look up the PID of the job, then kill -p n the PID of the job.
- **d. Execute and capture the commands, using process notation (PID), to terminate a.out**

  ttanasse1@cslinux: /lab3$ ./a.out
  ...Ẑ
  [1]+ Stopped ./a.out
  ttanasse1@cslinux: /lab3$ ps
  PID TTY TIME CMD
  19348 pts/0 00:00:00 bash
  21996 pts/0 00:00:00 a.out
  22080 pts/0 00:00:00 ps]] ttanasse1@cslinux: /lab3$ kill -9 21996
  [1]+ Killed ./a.out
- **f. Execute and capture the commands, using job notation (job ID), to terminate a.out**

  ttanasse1@cslinux: /lab3$ ./a.out
  ...Ẑ
  [1]+ Stopped ./a.out
  ttanasse1@cslinux: /lab3$ kill -9 %1

  [1]+ Stopped ./a.out

- **23) 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.)**

  ttanasse1@cslinux: /lab3$ ./a.out > /dev/null &
  [1] 25098
  ttanasse1@cslinux: /lab3$ ./a.out > /dev/null &
  [2] 25102
  ttanasse1@cslinux: /lab3$ ./a.out > /dev/null &
  [3] 25105

  - **a. What are the job numbers of the above?**
    1, 2, and 3
  - **b. What are the process ID numbers of the above?**
    25098, 25102, 25105
  - **c. Capture the command and output to bring the second a.out to the foreground.**

    ttanasse1@cslinux: /lab3$ fg %2
    ./a.out > /dev/null
  - **d. Capture the command(s) to send a.out back to the background.**

    ^Z
    [2]+ Stopped ./a.out > /dev/null
    ttanasse1@cslinux: /lab3$ bg %2
    [2]+ ./a.out > /dev/null &
  - **e. Capture the command(s) to kill the third a.out using its job number.**

    ttanasse1@cslinux: /lab3$ kill -9 %3
    ttanasse1@cslinux: /lab3$ jobs
    [1] Running ./a.out ¿ /dev/null &
    [2]- Running ./a.out ¿ /dev/null &
    [3]+ Killed ./a.out ¿ /dev/null
  - **f. Capture the command(s) to kill the first a.out using its process number.**

    ttanasse1@cslinux: /lab3$ kill -9 25098
    [1]- Killed ./a.out > /dev/null
  - **g. 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.**
    No, it can't be used to kill background jobs. It sends a specific signal to the job, which will kill it if it doesn't handle it itself.

- **24) 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.**

  - **a. echo $var**
    Since double quotes doesn't start literal, you will see the word 'date', without the single quotes.
  - **b. echo $var**
    Since single quotes make the inside characters all literal, you will see '$var', without the single quotes.
  - **c. echo var**
    Since echo is not explicitly told to treat var as a variable, it treats it as characters and prints 'var', without the single quotes.