Steven Tang

CSE 460

1/17/17

Lab 2

Objective:

The objective of this lab is to get familiar with several UNIX commands. As an addition to learning the commands, we should be able to use it in the command line as well as in a script. We will learn to write a script that will find the user inputted process and terminate all of them if they are running.

Basic Shell Programing

ginfo script

```
1.  ./ginfo
2.  hello steventang
3.  Today is
4.  Thu Jan 19 19:09:31 PST 2017
5.  Number of user login :
6.         2
7.  Calendar
8.      January 2017
9.  Su Mo Tu We Th Fr Sa
10.  1  2  3  4  5  6  7
11.  8  9 10 11 12 13 14
12. 15 16 17 18 19 20 21
13. 22 23 24 25 26 27 28
14. 29 30 31
15.
16.
17. Script done on Thu Jan 19 19:09:31 2017
```

- ./ginfo runs the script while .ginfo does not. The ./ directs us to the current directory while just a . is in the current shell. In order for us to run our script, we need to use the ./ syntax.

1. How do you define variable x with value 10 and print it on screen?

```
1)  clear
2)  x=10
3)  echo $x
```

2. How do you define variable xn with value “Rani” and print it on screen?

```
1)  clear
2)  xn=Rani
3)  echo $xn
```

3. How do you print the sum of two numbers, say, 6 and 3?

```
1.  clear
```

```
x=6
y=3
sum=$(($x+$y))
echo $sum
```

4. How do you define two variables x=20, y=5 and then print the quotient of x and y (i.e. x/y)?

```
clear
x=20
y=5
echo $((x/y))
```

5. Modify the above question to store the result of dividing x by y to a variable called z.

```
clear
x=20
y=5
z=$((x/y))
echo $z
```

- When running ./test.sh and then running echo $XYZ, we get nothing. When we run . ./test.sh and then run the command echo $XYZ, we get 2017. This happens because we run ./test.sh, we create a new instance while . ./test.sh does not.

Awk:

- When I run the command ps auxw | awk '{print $1 "\t\t" $2}' it shows all the processes that's running for the user by printing it to the terminal

Starting New Processes:

- Before changing our test_system.cpp file, we see all our system instances ID's that aren't running in the background. The difference when changing system("ps –ax &") was that it showed all of the programs that are running in the background. The & denotes that we want the processes that are running in the background.

Shell Programming Practice:

- Grep –v Shows all the selected lines that doesn't match the specified pattern. This is called invert-match
- When using top, top consumes the most CPU. It fluctuates between 1.5 and 2%. The one that contains the most memory is mdwork32.

terminateProcess.sh

```
process=$(pgrep $1)
if [ "$process" == "" ] #if pgrep returns a null, that means no process is running
then
    echo "No such task exists"
else
    kill $process
```

Type Script of running 4 robot process in the background and terminating them:

```
Script started on Wed Jan 18 23:49:49 2017
[?1034hbash-3.2$ ./robot &
[1] 1616
bash-3.2$ ./robot &
[2] 1617
bash-3.2$ ./robot &
[3] 1618
bash-3.2$ ./robot &
[4] 1619
bash-3.2$ ./T[KterminateProcess &[Krobot
[2]   Terminated: 15          ./robot
[4]+  Terminated: 15          ./robot
[1]-  Terminated: 15          ./robot
[3]+  Terminated: 15          ./robot
bash-3.2$ ps
  PID TTY           TIME CMD
 1353 ttys000    0:00.16 -bash
 1614 ttys000    0:00.01 script
 1615 ttys001    0:00.01 /bin/bash -i
bash-3.2$ ei[Kxit
exit

Script done on Wed Jan 18 23:50:10 2017
```

Evaluation

In this lab, I had completed all the work that was required. I wrote multiple shell scripts and learned how they worked. I also distinguished the difference between different commands such as "./ginfo" and ".ginfo", and "./testShell.sh" and ". ./testShell.sh". I had also completed my script program that would allow the user to terminate a program's instances that is running in the background. As for my score, I would give myself a 20/20.

Score: 20/20