1. Semester and Year : 2012 FALL
2. Course Number : CS-227
3. Course Title : Linux & System Software
4. Work Number : LA-07
5. Work Name : Lab 08
6. Work Version : Version1
7. Long Date : Sunday, 21, October, 2012
8. Author(s) Name(s) : Jake Waffle

Exercise 1

Explain the following unexpected result:

$ whereis data

$ echo $PATH

$ cat > data

echo ”This is my date.” (ctrl+d)

$ ./data

In the example above, it seems that the user is trying to execute the script that was created with the cat command, by means of redirection to a new file name. But the problem is that the file that was saved by default doesn't have permission to be executable. So this is something that has to be explicitly changed. It can be done with the following command:

$ chmod a+x date

Exercise 2

The two ways that one is able to execute a shell script that doesn't have execute permission for its file are:

1. One can first change the file's execute permission for a user, group and other (a user that doesn't belong to the file's group and doesn't own the file) before executing the file. That can be done with the chmod command with arguments “+x” to add the execute permission and “a” so that all users have this permission for the given file.
   Ex: $ chmod a+x fileName

2. The other way is to override the file permissions with the source or dot command (user784637, 2012.) What the previously stated command does is basically read the given text file and interpret it as a set of instructions. And it looks something like:
   $. fileName or
   $ source fileName

Exercise 3

The task at hand is to inform about the purpose of the PATH variable.

a) In order to change my PATH variable to the specified directories on the print out, I set it temporarily in the terminal:

$ PATH=/usr/local/bin:/usr/bin:/bin:/usr/kerberos/bin:$HOME/bin:$PWD

b) If there are two files with the same name, then the file with the directory that comes first in the $PATH variable will be the one that is used when called upon. So now looking at the work sheet's question, I think that the “doit” file located within /usr/bin is the one that will be executed given the $PATH variable as defined previously in Exercise 3a.

c) If the $PATH variable isn't setup to search the Present Working Directory, then one can explicitly give the full path to the file that is wanted to be accessed.

d) To add /directory to the end of the environment variable PATH without overwriting its contents, I'd use the following command.

$ PATH=$PATH:/directory

Exercise 4

The task is to give the outputs of a few commands assuming the following assignment.

$ person=zach

a) For the command

$ echo $person

the output is going to be what the $person variable is pointing to, which in this case is “zach.”

b) For the command

$ echo '$person'

the output is going to be the literal string within the single quotes, which in this case is

“$person.”

c) For the command

$ echo “$person”

the output will be dictated by the value pointed to by the variable within the double quotes. In this case that value will be “zach.”

Exercise 5

This task is centered around a given script on the work sheet.

a) In order to execute the script given on the work sheet, the script first needs to include in the first line of the script the path of the shell program being used (there also has to be “#!” before the shell path.) So the first line in the script may look like the following, assuming the bash shell is in use:

#!/bin/bash

I also have to note that the script either needs to have execute permission or needs to be ran with the source command.

b) The read builtin seems to work just like raw_input() in python, except that it requires that the message is printed before asking for the input. But the reason that read and cat are used in the different cases for input gathering is because of the input that is being gathered. When asking for a name, it's assumed more or less that only one line is going to be used to enter the name, so read is used (the input is submitted with the enter key.) But when asking for the journal entry itself, there's a good chance that more than one line is used. So cat is used, because it allows many lines of input to be redirected to a file (input is submitted with ctrl+d.)

Exercise 6

The tasks assumes that directories /home/zach/grants/biblios and /home/zach/biblios exist. And for each of the following sequence of commands, I have to determine what the working directory will be after execution.

a)

$ pwd

/home/zach/grants

$ CDPATH=$(pwd)

$ cd

$ cd biblios

The concluding working directory after the sequence of commands above is executed would be /home/zach/grants/biblios in this example. That is because “cd” executed alone will bring Zach to /home/zach/grants (which is equal to $CDPATH at this time.) Then the directory is changed to /home/zach/grants/biblios, because Zach was in /home/zach/grants while using the following command:

$ cd biblios

b)

$ pwd

/home/zach/grants

$ CDPATH=$(pwd)

$ cd $HOME/biblios

The sequence of commands above will conclude with the working directory being

/home/zach/biblios. This is because even though the $CDPATH is being set to /home/zach/grants, Zach isn't even referencing $CDPATH. He is instead referencing a full path name with the following command:

$ cd $HOME/biblios

And the path name above equals /home/zach/biblios.

Exercise 7

Two ways to identify the PID number of the login shell are:

$ ps

and

$ echo $$

The first command displays process statuses (we've used this in previous assignments.) While the second one specifically outputs the PID of the login shell (Answers.) The second one was totally unintuitive and isn't listed on echo's man pages. So “$$” must be some kind of environment variable that we didn't go over. But I at least tested the command and know that it works.

Exercise 8

Given the following command:

$ sleep 30 | cat /etc/inittab

The output of sleep will be a pause of x seconds I guess. Cat gets its input from the contents of /etc/inittab. And the shell won't display its prompt until 30 seconds has passed. So the command overall will just display the contents of /etc/inittab and wait 30 seconds until the user is allowed to access the prompt.

References

-Roderick W. Smith. linux administrator StreetSmarts. Induanapolis, Indiana: Wiley Publishing, Inc, 2007

-user784637. ”ask ununtu.” 2 September 2012. Web. 21 October 2012. <http://askubuntu.com/questions/183377/source-or-dot-operator-overriding-execute-permission.>

-“Answers.” Web. 21 October 2012. <http://wiki.answers.com/Q/What_two_ways_can_you_identify_the_PID_number_of_your_log in_shell.>