**ULI101-ASSIGNMENT 3**

1. Explain the following unexpected result:

**$ whereis date**

date: /bin/date ...

whereis date tell us the location of file date, according to the $path value results shows us that executable file date is in /bin

**$ echo $PATH**

.:/usr/local/bin:/usr/bin:/bin

Shows the content of $PATH environment parameter, all the paths are separated by colon.

**$ cat > date**

echo "This is my own version of date."

It takes standard input from keyboard and direct them to the new file in current directory called date.

**$ ./date**

Sun May 21 11:45:49 PDT 2017

Attempts to execute date file, but this file is not executable by default therefore it will search for the next directory from the $PATH environment variable which shows current date and time as an output

1. What is the purpose of the PATH variable?

**a.** Set the PATH variable and place it in the environment so it causes the shell to search the following directories in order:

• /usr/local/bin

• /usr/bin

• /bin

• /usr/kerberos/bin

• The bin directory in your home directory

• The working directory

$PATH = usr/local/bin: /usr/bin: /usr/kerberos/bin: /~/bin: /~

**b.** If there is an executable file named doit in /usr/bin and another file with the same name in your ~/bin directory, which one will be executed?

The file in our home directory will be executed first because it is in the $PATH.

**c.** If your PATH variable is not set to search the working directory, how can you execute a program located there?

If we want to execute a program that is not in our PATH, we need to use full command directory structure example: /usr/bin/doit

**d.** Which command can you use to add the directory /usr/games to the end of the list of directories in PATH?

Export PATH = $PATH: /usr/games

1. The following shell script adds entries to a file named journal-file in your home directory. This script helps you keep track of phone conversations and meetings.

**$ cat journal**

# journal: add journal entries to the file

# $HOME/journal-file

file=$HOME/journal-file

date >> $file

echo -n "Enter name of person or group: "

read name

echo "$name" >> $file

echo >> $file

cat >> $file

echo "----------------------------------------------------" >>

$file

echo >> $file

1. What do you have to do to the script to be able to execute it?

Chmod +x journal   
makes the file executable

Chmod +x filename.sh

Run script: ./filename.sh

1. Why does the script use the read builtin the first time it accepts input from the terminal and the cat utility the second time?

It reads the name from standard input and cats it to journal rather than standard output.

1. Name two ways you can identify the PID number of the login shell.

Ps: this command displays information about active processes, it shows PID, TTY, time and CMD.

$$: displays process id of current processes.

1. **a**. How can you prevent a command from sending output to the terminal when you start it in the background?

We can prevent it from sending output to standard terminal by redirecting standard output and standard error.

$prog > & prog.out &

**b.** What can you do if you start a command in the foreground and later decide that you want it to run in the background?

We will first suspend the process by typing control-z then move it to background by bg command in that way we can successfully move the process from the foreground to background.

1. Assume that the working directory contains the following files:

adams.ltr.03

adams.brief

adams.ltr.07

abelson.09

abelson.brief

anthony.073

anthony.brief

azevedo.99

What happens if you press TAB after typing the following commands?

1. **less adams.l**

$ less adams.ltr.0

1. **cat a**

$ cat a

1. **ls ant**

$ ls anthony

1. **file az**

$ file azevedo.99

What happens if you press CONTROL-D after typing the following commands?

1. **ls ab**

$ ls will look exactly for the file name ab, control-d will terminate the current shell processes.

1. **less a**

no file or directory, control-d will terminate the current shell processes.

1. Write an alias named qmake (quiet make) that runs make with both standard output and standard error redirected to the file named make.log. The command qmake should accept the same options and arguments as make.

tsch $alias make “qmake \!\* >& make.log”

1. Write a filter that takes a list of files as input and outputs the basename (page 459) of each file in the list.

$cat base

read fn

while [ “a$fn” != “a” ]

do

basname “$fn”

read fn

done

1. When might it be necessary or advisable to write a shell script instead of a shell function? Give as many reasons as you can think of.

The shell script can be executed, run, and can call other shell script and functions whereas a function is not executed until and unless it is called by the code typically in a script. Shell scripts are also faster than the shell functions as too many functions running at same time in shell requires long to load so therefore shell script is a more efficient way.

1. Write a script to display the time every 15 seconds. Read the date man page and display the time, using the %r field descriptor. Clear the window (using the clear command) each time before you display the time.

While:

do

time = ‘date +%r’

echo -n $time

sleep 15

clear

done

1. Read the bash man or info page, try some experiments, and answer the following questions:
2. How do you export a function?

$export -f function name

1. What does the hash builtin do?

The hash bultin remembers the absolute pathnames of programs named in its arguments and without any arguments hash lists all of the commands it is tracking.

1. What happens if the argument to exec is not executable?

If the argument to exec is not executable file, exec fails.