**ULI101SA**

**Assignment 3**

1. **Chapter 8 ~ Question 1 – Page 366 [8 Marks]**

**Explain the following unexpected result:**

**$** **whereis date**   
date: /bin/date ...   
**$ echo $PATH**   
.:/Usr/local/bin: /usr/bin: /bin   
**$ cat > date**   
echo "This is my own version of date."   
**$. /date**

Tue May 21 11:45:49 PDT 2013

1. **Where is date**" tells us where the executable file "date" is located, according to the current $PATH value. The result tells us that the executable file "date" is found in /bin   
   **2. echo $PATH**   
   tells us what is the content of the $PATH environment parameter. Each path is separated by a colon ":".   
   **3. "cat > date"**takes standard input (stdin) from the keyboard and put the keyed in content into a new file in the current directory called date. The input should be terminated by a control-d, which is not mentioned in the question.   
   The file "date" usually has a permission of 644 or 600 (depending on the computer implementation), which means that it is not executable.   
   **4- . /date**attempts to execute the file ./date, but it is not executable (by default). Again, depending on the implementation of the system, it may return   
   "permission denied", or possibly search for the next directory from the **$PATH** environment variable, which outputs the current date and time.

**2. Chapter 8 ~ Question 3 – Page 366 [10 Marks]**

**What is the purpose of the PATH variable?**

The PATH variable specifies the directories in the order the shell should search • them. Each directory must be separated from the next by a colon.

1. Set the PATH variable 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:/bin:/usr/kerberos/bin:~/bin**

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

It is determined by the order in the path statement. The command will execute the /usr/bin file

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

If you need to run a file that is not in your path, you would use the full command directory structure like: /usr/bin/test or cd to the /usr/bin directory and type ./test

1. **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

3. Chapter 8 ~ Question 4 – Page 366 [3 Marks]

Assume you have made the following assignment:

$ person=zach

Give the output of each of the following commands:

1. **echo $person**

zach

1. **echo '$person'**

$person

1. **echo "$person"**

zach

**4. Chapter 9 ~ Question 2 – Page 413 [4 Marks]**

1. **a. How can you display the aliases currently in effect?**

Give the command **alias** to list aliases

**b. Write an alias named homedots that lists the names (only) of all hidden files in your home directory.**

$ alias homedots 'ls -d ~/.\*'

**5. Chapter 9 ~ Question 3 – Page 413 [4 Marks]**

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

Redirect standard output and standard error of a command to prevent it from sending output to the terminal: $ prog >& prog.out &

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

To move a process from the foreground to the background, first suspend it by typing **CONTROL-Z**, and then move it to the background with a **bg** command.

**6. Chapter 9 ~ Question 5 – Page 413 [12 Marks]**

**5- 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 these commands?

1. **ls ab**

ls will only look for the a file with the exact name ab

Ctrl-D will terminate the current shell process.

1. **less a**

No such file or directory

Ctrl-D will terminate the current shell process

**7. Chapter 10 ~ Question 2 – Page 512 [2 Marks]**

**2. The special parameter "$@" is referenced twice in the out script (page 425). Explain what would be different if the parameter "$\*" were used in its place**

If you replace "$@" with "$\*" in the out script, cat or less would be given a single argument: a list of all files you specified on the command line enclosed within single quotation marks. This list works when you specify a single filename. When you specify more than one file, the shell reports No such file or directory because there is not a file named the string you specified on the command line (the SPACEs are not special characters when

they are enclosed within single quotation marks)

**8. Chapter 10 ~ Question 5 – Page 512 [2 Marks]**

**When might it be necessary or advisable to write a shell script instead of a shell function?**

A shell function shall do nothing if it is not called from within another code segment. It actually does not have any identity of its own without its caller. On the other hand, a shell script can be executed directly from the shell prompt and be run by any other user too. Also, a shell function must be a way to accomplish a job when the job is needed very often. A shell script is a runnable, executable process, which can call other shell scripts and/or functions.

**9. Chapter 10 ~ Question 10 – Page 513 [10 Marks]**

**Using the find utility, perform the following tasks:**

1. **List all files in the working directory and all subdirectories that have been modified within the last day**

$ find . -mtime -1

1. **List all files that you have read access to on the system that are larger than 1 megabyte**

$ find / -size +1024k

1. **Remove all files named core from the directory structure rooted at your home directory.**

$ find ~ -name core -exec rm {} \;

1. List the inode numbers of all files in the working directory whose file-names end in .c.

$ find . -name "\*.c" -ls

1. List all files you have read access to on the root file system that have been modified in the last 30 days.

$ find / -xdev -mtime -30