Jacob Metcalfe

Homework #6

Using what you’ve learned so far in this course, specifically week 7, answer the following:

Question 1. List three or more benefits of **getopt/getopts** verses traditional parsing?

No need to pass the positional parameters through to an external

program.

Can set shell variables to use for parsing (impossible for an

External process!)

There's no need to argue with several getopt implementations which had buggy concepts in the past (whitespace, etc)

Question 2. The following is true about file-redirection metacharacters: (Choose all that apply.)

**a. The < directs the content of a file to the command.**

b. The **2>** directs standard input of a command to a file.

**c. The > directs the standard output of a command to a file.**

d. The &**>** directs only standard error of a command to a file.

e. The **>>** directs standard output of a command and overwrites the file.

Question 3. Explain how to prompt a user for information and then capture the input.

First you have to read the information with the “read command”. Then you prompt the user of what information that you need. Then, after the command you put the variable that the user input will be stored into. You can then print their input by using echo “$variablename”

Example

read -p “Type in your first name: ” NAME

Question 4. The following are special variables called position parameters: (Choose all that apply.)

a. Y

**b. $1**

**c. $0**

d. $X

**e. $#**

Question 5. List three common read command options and explain their function.

-r – Do not allow backslashes to escape any character

-p – creates prompt for the user

-s – do not echo input coming from a terminal

Question 6. How can data from STDIN be sent to both a file and STDOUT?

Using the tee command you can send data from STDIN to two destinations, allowing for savings and displaying data at the same time.

Example of tee use

$ date | tee testfile

Mon Oct 05 18:55:01 MST 2015

Read the following shell script and answer the questions:

**$cat script1**

**#!/bin/bash**

**exec 3<$1**

**exec 4<$2**

**exec 5>$3**

**while read line <&3**

**do**

**echo “$line” >&5**

**done**

**while read line <&4**

**do**

**echo “$line” >&5**

**done**

Question 7. The script requires 3 filenames as arguments, e.g., **./script1 file1 file2 file3.** Which files are input files and which are output files?

File 1 and 2 are input and 3 is output

Question 8. Inside the while-loop, why does variable “**$line**” have to be enclosed by double quotes?

When printing this it has to be as a string, so double quotes allow it do to so. Single quotes don’t allow it to be a string in these cases.