COMPSCI 2211b
Software Tools and Systems Programming

Assignment #2

Shell Scripts

Posted:	January 31st 2018
Due:	February 14th 2018 11:55PM
Total:	100 Points (5% of Final Grade)

Learning Outcomes

By completing this assignment, you will gain and demonstrate skills relating to:

- Reading and understanding shell scripts.
- Creating your own shell scripts.
- More practice with **tr**, **grep**, **wc**, **sort** and **head**.

Instructions

For this assignment, an electronic submission is required through OWL. This submission should include a document in PDF format that contains answers for each of the questions listed in the following sections as well as a copy of each shell script you created in plain text format. Your PDF file should be named *"userid_assign2.pdf"* where *"userid"* is your user id. For example, if your UWO e-mail is *"dservos5@uwo.ca"* your file should be named *"dservos5_assign2.pdf"*. Your shell scripts should be named as per the directions given in the question.

Your answers should be clearly labelled with question and part numbers. Your shell scripts should be included in both the PDF and as their own plain text file. In addition to your answers, this document should also contain your full name, student number, UWO user id, the course code, date, and assignment number.

All Linux commands and shell scripts given must run correctly on the course server (cs2211b.gaul.csd.uwo.ca) in the Bash shell. A question will only be marked as correct if your output can be replicated on the course server.

You will be assessed on the following:

- Completion of each question correctly.
- Providing output where required.
- Providing a PDF formatted file following the naming convention.
- Providing a plain text copy of each shell script you create with the correct file name.
- Including your full name, student number, etc. as described above.
- Assignment submission via OWL.

Questions

Question 1 (10 Marks)

Consider the following bash script ***test1***:

```
1  #!/bin/bash
2  # My Shell Script!
3
4  x=$1
5  echo $0
6  echo $1
7  echo $#
8  echo $*
9  echo $x
10
11 shift
12
13 echo $0
14 echo $1
15 echo $#
16 echo $*
17 echo $x
```

(a) Explain what each numbered line of the script does, what it would output (if anything) and why.

(b) Show the output produced by the following call to test1:

```
test1 a b c
```

Question 2 (10 Marks)

Consider the following bash script ***test2***:

```
1  #!/bin/bash
2
3  echo -n 'Input Number: '
4  read x
5
6  echo expr $x + 7
7  echo "expr $x + 7"
8  echo 'expr $x + 7'
9  echo `expr $x + 7`
```

(a) Explain what each numbered line of the script does, what it would output (if anything) and why.

(b) Show the output produced by the running ***test2*** and inputting the number 10.

Question 3 (15 Marks)

Consider the following bash script ***good***:

```
1  #!/bin/bash
2
3  echo
4  hour=`date +%H`
5
6  if [ "$hour" -lt 12 ]
7  then
8    echo "GOOD MORNING"
9  elif [ "$hour" -lt 18 ]
10   then
11     echo "GOOD AFTERNOON"
12   else
13     echo "GOOD EVENING"
14 fi
15
16 echo
```

(a) Explain in one or two sentences what this script does.

(b) Rewrite this script to not use ***elif***. Make sure you include a plain text copy of the rewritten script named ***good2*** in your submission.

Question 4 (20 Marks)

Write a bash shell script called ***dobackup*** that does the following in order:

1. Output the text "Input a directory name:".
2. Read a directory path from the user via standard input. You can assume that the path contains no spaces and you do not need to support the tilde character (∼).
3. Check if the directory path is valid (i.e. that the directory exists) and print an error and exit if it is invalid (does not exist).
4. If a directory named ***backup*** already exists in the current working directory, delete it and all the files and subdirectories it contains.
5. Make a copy of the directory given by the user to the current working directory and name it ***backup*** (this copy should contain all files and subdirectories of the original).
6. Use the **tar** command to make an archive of the backup directory named ***backup.tar*** in the current working directory. We did not go over the **tar** command in class so it is up to you to read the man page for **tar**. You do not need to use compression.
7. Rename ***backup.tar*** such that its file name contains the current date in the following format:

backup-<day>-<month>-<year>.tar

where <*day*> is the current day (with a leading 0 if one digit), <*month*> is the current month (with a leading zero if one digit) and <*year*> is the current 4 digit year. For example, if today was January 31st, 2018 the file should be renamed to ***backup-31-01-2018.tar***

8. Use **scp** to send a copy of the tar file to a directory in your home directory named ***mybackups*** on the course server (it is assumed that this directory already exists). This command should be written so it will work on the lab computers or the course server. Rather than using your username with the **scp** command use the value of the *$USER* variable (contains the current user). It is OK if **scp** asks the user for their password.

Add comments to your code explaining what every line does as well as a comment at the beginning of the file (under `#!/bin/bash`) that contains your name, student number and a one or two sentence description of what the script does.

Test your script both on the course server and on a lab computer in MC244. Ensure that it works on both systems and that a copy of your backup archive is sent to the course server in both cases. Include a plain text copy of your script named ***dobackup*** as well as a copy in your PDF file.

Question 5 (20 Marks)

Write a bash shell script called ***avg*** that takes the average of the integer arguments given to it on the command line. The average should be sent to the standard output. Assume that all arguments are valid integers and that there can be more than 9 arguments.

Your shell script must use a loop and at least one variable.

The number you output should have two decimal places.

Your shell script should output an appropriate error if no arguments are given to it.

You should include a comment at the beginning of the file (under `#!/bin/bash`) that contains your name, student number and a one or two sentence description of what the script does. Also add at least **two** comments to lines in your script explaining what they do.

Test your script and provide the output for the following commands:

Command	**Correct Output**
`avg`	*Need at least one argument!*
`avg 42`	42.00
`avg -50 0 50 100`	25.00
`avg 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20`	10.50

Make sure to both include your shell script in the PDF and as a plain text file named ***avg***.

Question 6 (25 Marks)

Write a bash shell script called ***nums*** that takes zero, one or two arguments and gives the following output:

Command	Result (output to standard out)
`nums`	"Usage: nums option filename"
`nums 0`	"Error: no filename given!"
`nums 1`	"Error: no filename given!"
`nums 2`	"Error: no filename given!"
`nums 3`	"Error: no filename given!"
`nums 0 file`	Outputs the number of lines in the file.
`nums 1 file`	Outputs the file with all letters made upper case and all whole numbers replaced with *s. For example: "I have 123 apples" would become "I HAVE * APPLES".
`nums 2 file`	Outputs only the lines in the file that are 7 letter palindromes. The palindromes should contain only letters (not numbers or special characters). The line cannot contain any other text or be any length other than 7 characters long (not counting line breaks). Capitalization should be ignored.
`nums 3 file`	Outputs the 4 largest numbers in the file. The file may contain text, in which case the text should be ignored. Numbers may be on the same line and separated by spaces or text and/or on different lines. ***Hint:*** *You will need to use a combination of the commands* ***cat***, ***head***, ***tr*** *and* ***sort****. Check the man page for* ***sort*** *to learn how to sort numbers and set the order.*
Any other option number	"Invalid option!"
If ***file*** does not exist	"***file*** does not exist!" where ***file*** is the name of the file.

You should include a comment at the beginning of the file (under `#!/bin/bash`) that contains your name, student number and a one or two sentence description of what the script does. Also add at least **four** comments to lines in your script explaining what they do.

Make sure to both include your shell script in the PDF and as a plain text file named ***nums***.

Example File (*file.txt*):

```
Hello World!
Racecar
deified reviver
123+456=579
ROTATOR
42
See citation (150) in Chapter 3.789.
Time to repaper the wallpaper.
```

Example Output:

```
$ nums
Usage:  nums options filename

$ nums 1
Error:  no filename given!

$ nums 4 file.txt
Invalid option!

$ nums 1 badfilename
badfilename does not exist!

$ nums 0 file.txt
8

$ nums 1 file.txt
HELLO WORLD!
RACECAR
DEIFIED REVIVER
*+*=*
ROTATOR
*
SEE CITATION (*) IN CHAPTER *.*.
TIME TO REPAPER THE WALLPAPER.

$ nums 2 file.txt
Racecar
ROTATOR

$ nums 3 file.txt
789
579
456
150
```