**CORE**

UNIX

Shell Programming Exercises

© FDM Group Ltd 2015. All Rights Reserved.

Any unauthorised reproduction or distribution in part  
or in whole will constitute an infringement of copyright

**Exercise 1**

vi

|  |
| --- |
| **Introduction to Shell Scripting** |
| 1. Write a script called myVars that displays the following variables $HOME, $LOGNAME and $PATH.   $HOME  /home/local/FDMGROUP/yangyang.ma  $LOGNAME  FDMGROUP\yangyang.ma  $PATH  /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin |
| 1. Write a script called greeting which displays a welcome message as follows “Hello firstname.lastname”. In your script replace firstname.lastname with your actual UNIX username.   For example if your UNIX username is joe.bloggs, your script should display  **Hello joe.bloggs** |
| 1. Write a script called createFiles that creates a directory called mydir and three files called myfile1, myfile2 and myfile3. (Note this script will generate no output, you will need to check it has worked) |
| 1. Write a script called moveFiles that moves myfile1, myfile2 and myfile3 into mydir. Display a message informing the user that the move operation is complete and then display the contents of the mydir directory.   Your output should be:-  **move operation completed**  **myfile1 myfile2 myfile3** |
| 1. Write a script called lions6 that displays the first 6 lines of /examples/lionsInTheStreet file. |
| 1. Write a script called fileCount that counts how many files you have in your ~ directory. |

**Exercise 2**

|  |  |
| --- | --- |
| **Variables and Command Substitution** |  |
| 1. Write a script called enterWord which prompts the user to enter a word. The script should display ‘You entered ………..’   Your script should run as follows:-  **Please enter word**  hello  **You entered hello** | |
| 1. Write a script called countLetters which prompts the user to enter a word. The script should then display the number of letters in the word.   Your script should run as follows:-  **Please enter word**  hello  **5** | |
| 1. Modify your previous countLetters script so that it displays the number of letters as part of a sentence: ‘The word …… has ….. letters’ (hint: use command substitution)   Your script should run as follows:-  **Please enter word** hello  **The word hello has 5 letters**  Var=” **The word hello has “** $(read –p “**Please enter word**” ) | wc –m “letters” | |
| 1. Write a script called displayArg which takes an argument and then displays ‘The argument is …….’   Your script should be run as follows:-  **displayArg hello**  And would output:-  **The argument is hello**   1. Write a script called charCount which takes an argument and then displays the number of characters in the argument.   Your script should be run as follows:-  **charCount hello**  And would output:-  **5** | |
| 1. Write a script called nameFile which takes your first name as an argument and then creates a file with your first name as its name. The script should then display the filename and i-node of your new file (hint: use ls to display i-node)   Your script should be run as follows:-  **nameFile joe**  And would output:-  **2099456 joe** where 2099456 is the i-node    Filename=$(echo $LOGNAME | cut -d"." -f1 | cut -d"\\" -f2)  (yangyang)  filename=(cut -d"." -f1 $LOGNAME)  touch $filename  echo "nameFile $filename"  inode=$(cut -d" " f1 ls -i $filename)  echo "$inode $filename" | |
| 1. Write a script called renameFile which takes your first and last names as arguments. The script should then rename the file you created in the previous exercise so that its name has the following format: firstname.lastname. Finally the script should display the i-node of this file.   Your script should be run as follows:-  **renameFile joe bloggs**  And would output:-  **2099456 joe.bloggs** | |

**Exercise 3**

|  |  |
| --- | --- |
| **Conditional scripts** |  |
| 1. Write a script called numComp that accepts 2 numbers as arguments and compares the integers. The script will output 1 of the following messages   The first number is greater than the second  The second number is greater than the first  The 2 numbers are equal. | |
| 1. Write a script called stringComp that accepts two strings as arguments. It will compare the length of both strings and output 1 of the following messages   The first string is longer than the second  The second string is longer than the first  Both strings are of equal length  #!/bin/bash  if [ ${#1} -eq ${#2} ] ; then  echo "equal"  else  echo "More than 2 arguments have been passed into the script"  fi | |
| 1. Modify your scripts for previous questions (1 and 2) to check the user supplies 2 arguments.   If the wrong number of arguments are supplied display an error message “Incorrect number of arguments” and exit the script with an error exit status.  In addition if no arguments are supplied display an error message “No argument supplied” and exit the script with an error exit status.  If [ !($# -eq 2) ] && [ !($# eq 0) ] ; then  Elif [ $# eq 0 ] ; then  Else … | |
| 1. Write a script called delFile that asks a user to enter a file name.   Firstly the script checks the file exists and if doesn’t displays an error message “Invalid filename” and exits the script with an error exit status.  Assuming the file exists the script asks the user if they would like to delete the file.  If they answer y or Y, or anything starting with y it will delete the file.  If they answer n or N or anything starting with n it will not delete the file and exit the script.  Any other response and the script will generate an “Invalid response” message.    #!/bin/bash  read -p "Enter a file name " filename  if [ !(-e $filename) ]  if [ ! –f $filename ]  ; then  echo "Invalid filename"  exit 1  else  read -p "Do you want to delete it?" answer  if [ $answer = [yY]\* ] ; then  rm $filename  elif [ $answer = grep '^[n,N]’ ] ; then  exit 1  else  echo "Invalid response"  fi  fi | |
| 1. Write a menu script called myMenu to enable users to do the following:-   1 – Display the date and time  2 – Display the current working directory  3 – Display the processes being run by the current user  The script should display the menu and prompt the user to select an option (1-3).  If the user selects anything other than 1-3 the script should display an error message and terminate with an error exit status.  #!/bin/bash  cat<<eof  1 – Display the date and time  2 – Display the current working directory  3 – Display the processes being run by the current user  eof  read -p "Enter an option : " opt  case $opt in  1)  date ;;  2)  pwd ;;  3)  ps ;;  \*)  echo "Invalid option."  exit ;;  esac | |

1. Modify your greeting script (Exercise 1 task 3) to greet you with 1 of the following messages

“Good morning firstname.lastname” if the time is before 1pm

“Good afternoon firstname.lastname” if the time is after 1pm and before 5pm

“Good Evening firstname.lastname” if the test is after 5pm

(Hint: Use the manual page to find out more about the date command)

h=`date +%H`

if [ $h -lt 12 ]; then

echo Good morning

elif [ $h -lt 18 ]; then

echo Good afternoon

else

echo Good evening

fi

**Exercise 4**

|  |  |
| --- | --- |
| **Using Flow control – Loops** |  |

1. Write a script called fileCounter, which accepts any number of filenames as arguments. For each file it will display the file size in characters.

#!/bin/bash

for file in \*

do

du -sk $file

done

|  |
| --- |
| 1. Modify your fileCounter script to also check each file exists, and if it is doesn’t exist to move onto the next file in the list.   #!/bin/bash  for file in \*  do  if[ !-f $file ]  then  continue  fi  du -sk $file  done |
| 1. Write a script called total which accepts a number as an argument and adds all numbers from 1 to that number and displays the total.   For example if 5 was the argument it would perform 1 + 2 + 3 + 4 + 5 giving an answer of 15.  #!/bin/bash  sum=0  for ((i=1;i<=$1;i++))  do  sum=$((sum+i))  Done  echo $sum |
| 1. Write a script called lineCounter that accepts a filename as an argument. It should run through the file line by line, displaying a character count for each line. The script should include a line counter, which records the line number and is incremented when each line is processed.   The output should look like:-  Line 1 contains 23 characters  Line 2 contains 12 characters  …..  #!/bin/bash  count=1  while read line  do  characters=$(echo -n $line | wc -c)  echo "Line $count has $characters characters"  ((count++))  done<$1 |
| 1. Modify your myMenu script from Exercise 3 item 5 to enable users to return to the menu and make another selection after successfully completing an option. In addition, add an extra option to allow the user to Quit the menu.   until [ $opt -eq 4 ]  do  cat<<eof  1 – Display the date and time  2 – Display the current working directory  3 – Display the processes being run by the current user  4 - Quit  eof  read -p "Enter an option : " opt  case $opt in  1)  date ;;  2)  pwd ;;  3)  ps ;;  4)  exit 1 ;;  \*)  echo "Invalid option."  exit ;;  esac  done |

1. Write a script called letterLoop that will prompt the user to enter a word. It should then use a loop to output each letter in turn. For example, if the user enters "picasso", the output should be:

Letter 1: p

Letter 2: i

Letter 3: c

Letter 4: a

Letter 5: s

Letter 6: s

Letter 7: o

#!/bin/bash

count=1

read -p "Enter a word: " word

length=$(echo $word | wc -c)

for ((count=1;count<$length;count++ ))

do

letter=$(echo $word | cut -c$count)

echo "Letter $count : $letter"

done

length=${#word}

Exit code 0 Success

Exit code 1 General errors, Miscellaneous errors, such as "divide by zero" and other impermissible operations

Exit code 2 Misuse of shell builtins (according to Bash documentation) Examp

done

**Exercise 5**

|  |  |
| --- | --- |
| **Functions** |  |
| 1. Write a script called calculator, which accepts two integers as parameters and perform basic arithmetic. Your script should include a menu as follows:-     ===========  CALCULATOR  ===========  1. Add the 2 numbers together  2. Multiply the 2 numbers together    Create 2 functions to perform options 1 and 2. The results should be included in a message, for example “The sum of 5 and 6 is 11”  Add additional functions to check:-  The user has supplied just 2 numbers  The input is numeric | |

**Exercise 6**

|  |  |
| --- | --- |
| **Built-in Commands** |  |
| 1. Modify your calculator script from the previous exercise (Exercise 5 task 1), to enable users to specify their option while calling the script, and the option will be as the following:   -a for addition.  -s for subtraction  -m for multiplication  -d for division | |