**5th Semester - UE18CS305 – Operating Systems Laboratory**

**WEEK 2: Shell Scripting** **Date: 04/09/2020**

STUDENTS ARE REQUIRED TO PRACTICE ALL THE COMMANDS PROVIDED IN THIS MANUAL, THEN ANSWER **6 EXERCISE PROBLEMS** GIVEN IN THIS MANUAL AND SUBMIT THE RESULTS IN WORD OR ZIP FORMAT ON EDMODO

DUE DATE FOR SUBMISSION: 10/09/2020 11:59 PM

**What is Shell Script?**

Normally shells are interactive. It means shell accepts command from you (via keyboard) and execute them. But if you use command one by one (sequence of 'n' number of commands), then you can store this sequence of command to text file and have the shell execute this text file instead of entering the commands. This is known as ***shell script***.

Shell script defined as:"*Shell Script is****series of commands****written****in plain text file****. Shell script is just like a batch file is MS-DOS but more powerful than the MS-DOS batch file.*"

**Why to Write Shell Script?**

* Shell script can take input from the user, file and output them on to screen or a file.
* Useful to create our own commands.
* Saves lots of time.
* To automate some tasks of day to day life.
* System Administration tasks can be automated.

For our **first shell script**, we'll write a "Hello World" script.

Create a file **first.sh** (using any editor of your choice) as follows:

[first.sh](https://www.shellscript.sh/eg/first.sh.txt)

#!/bin/sh

# This is a comment!

echo Hello World # This is a comment, too!

The first line tells Unix that the file is to be executed by /bin/sh. This is the standard location of the Bourne shell on just about every Unix system. If you're using GNU/Linux, /bin/sh is normally a symbolic link to bash (or, more recently, dash).

The second line begins with a special symbol: **#**. This marks the line as a comment, and it is ignored completely by the shell.

**Department of Computer Science and Engineering** **Page 1** **Aug – Dec 2020**

The only exception is when the *very first* line of the file starts with **#!** - as ours does. This is a special directive which Unix treats specially. It means that even if you are using csh, ksh, or anything else as your interactive shell, that what follows should be interpreted by the Bourne shell.  
Similarly, a Perl script may start with the line **#!/usr/bin/perl**to tell your interactive shell that the program which follows should be executed by perl. For Bourne shell programming, we shall stick to **#!/bin/sh.**

The third line runs a command: **echo**, with two parameters, or arguments - the first is **"Hello"**; the second is **"World"**.  
Note that **echo** will automatically put a single space between its parameters.  
The **#** symbol still marks a comment; the # and anything following it is ignored by the shell.

Now run **chmod 755 first.sh** to make the text file executable, and run **./first.sh**.  
Your screen should then look like this:

$ chmod 755 first.sh  
$ ./first.sh  
Hello World  
$

You will probably have expected that! You could even just run at the shell prompt:

$ echo Hello World  
Hello World  
$

**How to write shell script?**

Following steps are required to write shell script:

(1) Use any editor like vi, gedit or mcedit to write shell script.

(2) After writing shell script set execute permission for your script as follows  
*syntax:*  
chmod permission your-script-name

*Examples:*  
$ chmod +x your-script-name  
$ chmod 755 your-script-name

*Note:* This will set read, write & execute permission (7 or binary 111) for owner

Read and execute permission only (5 or binary 101) for group and other.

**Department of Computer Science and Engineering** **Page 2** **Aug – Dec 2020**

**How to** **Execute your script**?  
*syntax:*  
bash your-script-name  
sh your-script-name  
./your-script-name

*Examples:*  
$ bash bar  
$ sh bar  
$ ./bar

*NOTE* In the last syntax **./** means current directory, But only **.** (dot) means execute given command file in current shell without starting the new copy of shell, The syntax for **.** (dot) command is as follows  
*Syntax:*  
**.** command-name

*Example:*  
$ **.** foo

Now you are ready to write a shell script that will print "Knowledge is Power" on screen. See the [common vi command list](http://www.freeos.com/guides/lsst/misc.htm#commonvi) , if you are new to vi.

|  |
| --- |
| $ vi second # # My first shell script # clear echo "Knowledge is Power" |

After saving the above script, you can run the script as follows:  
$ ./second

This will not run script since we have not set execute permission for our script *second*; to do this type command  
$ chmod 755 second  
$ ./second

First screen will be clear, then Knowledge is Power is printed on screen.

[**How Shell Locates the file**](http://www.freeos.com/guides/lsst/misc.htm#howshell) (My own bin directory to execute script)

***Tip:*** For shell script file try to give file extension such as .sh, which can be easily identified by you as shell script.

**Department of Computer Science and Engineering** **Page 3** **Aug – Dec 2020**

**Exercise 1:**  
1) Write the following shell script, save it, execute it and note down the output.

|  |
| --- |
| # Script to print user information who currently login, current date & time # Enter the following commands in a file  # clear echo "Hello $USER" echo "Today is ";date echo "Number of user login : " ; who | wc -l echo "Calendar" cal exit 0 |

At the end, why statement exit 0 is used? What is the meaning of $? See [exit status](http://www.freeos.com/guides/lsst/ch02sec09.html) for more information.

**VARIABLES IN SHELL**

In Linux (Shell), there are two types of variable:  
(1) **System variables** - Created and maintained by Linux itself. This type of variable defined in CAPITAL LETTERS.  
(2) **User defined variables (UDV)** - Created and maintained by user. This type of variable defined in lower letters. Some of the important System variables are:

|  |  |
| --- | --- |
| **System Variable** | **Meaning** |
| BASH=/bin/bash | Our shell name |
| BASH\_VERSION=1.14.7(1) | Our shell version name |
| COLUMNS=80 | No. of columns for our screen |
| HOME=/home/ubuntu | Our home directory |
| LINES=25 | No. of columns for our screen |
| LOGNAME=students | students Our logging name |
| OSTYPE=Linux | Our Os type |
| PATH=/usr/bin:/sbin:/bin:/usr/sbin | Our path settings |
| PS1=[\u@\h \W]\$ | Our prompt settings |
| PWD=/home/ubuntu/Common | Our current working directory |
| SHELL=/bin/bash | Our shell name |
| USERNAME=ubuntu | User name who is currently login to this PC |

***NOTE*** that Some of the above settings can be different in your PC/Linux environment. You can print any of the above variables contains as follows:

**Department of Computer Science and Engineering** **Page 4** **Aug – Dec 2020**

$ echo $USERNAME  
$ echo $HOME

**Exercise 2:**  
1) If you want to print your home directory location then you give command:  
a) echo $HOME

**OR**

(b)echo HOME

Which of the above command is correct & why?

**Caution:**Do not modify System variable, this can some time create problems.

**How to define User defined variables (UDV)**

To define UDV use following syntax  
*Syntax:*  
variable name=value

'value' is assigned to given 'variable name' and Value must be on right side = sign.  
  
***Example****:*$ no=10 # this is ok  
$ 10=no # Error, NOT Ok, Value must be on right side of = sign.  
To define variable called 'vehicle' having value Bus  
$ vehicle=Bus  
To define variable called n having value 10  
$ n=10

**To print or access UDV use following syntax**To print contains of variable 'vehicle' type  
$ echo $vehicle  
It will print 'Bus'

To print contains of variable 'n' type command as follows  
$ echo $n

It will print ‘10’

**echo Command**

Use echo command to display text or value of variable.

echo [options] [string, variables...]  
Displays text or variables value on screen.  
Options  
-n Do not output the trailing new line.

**Department of Computer Science and Engineering** **Page 5** **Aug – Dec 2020**

-e Enable interpretation of the following backslash escaped characters in the strings:  
\a alert (bell)  
\b backspace  
\c suppress trailing new line  
\n new line  
\r carriage return  
\t horizontal tab  
\\ backslash

For e.g. $ echo -e "An apple a day keeps away \a\t\tdoctor\n"

 How to display colourful text on screen with bold or blink effects, how to print text on any row, column on screen, [click here for more!](http://www.freeos.com/guides/lsst/misc.htm#colorfunandmore)

**Shell Arithmetic**

*Syntax:*  
expr op1 math-operator op2

**Exercise 3:**  
**What is the output of the following expressions?**  
$ expr 1 + 3  
$ expr 2 - 1  
$ expr 10 / 2  
$ expr 20 % 3  
$ expr 10 \\* 3  
$ echo `expr 6 + 3`

**Exercise 4:**

What is the meaning of Single quote (‘), Double quote (“) and Back quote (`) in shell?

**Department of Computer Science and Engineering** **Page 6** **Aug – Dec 2020**

**5th Semester - UE17CS305 – Introduction to Operating Systems Laboratory**

**Wild cards (Filename Shorthand or meta Characters)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Wild card /Shorthand** | **Meaning** | **Examples** | |
| **\*** | Matches any string or group of characters. | **$ ls \*** | will show all files |
| **$ ls a\*** | will show all files whose first name is starting with letter 'a' |
| **$ ls \*.c** | will show all files having extension .c |
| **$ ls ut\*.c** | will show all files having extension .c but file name must begin with 'ut'. |
| **?** | Matches any single character. | **$ ls ?** | will show all files whose names are 1 character long |
| **$ ls fo?** | will show all files whose names are 3 character long and file name begin with fo |
| **[...]** | Matches any one of the enclosed characters | **$ ls [abc]\*** | will show all files beginning with letters a,b,c |

# Redirection of Standard output/input i.e. Input - Output redirection

It's possible to send output to a file or to read input from a file.

For e.g.  
**$ ls** command gives output to screen; to send output to file of ls command use   
**$ ls > filename**  
It means put output of ls command to filename.

There are three main redirection symbols **>, >> and <**

(1) > Redirection symbol

To output Linux-commands result (output of command or shell script) to file. Note that if file already exists, it will be overwritten, else new file is created. For e.g. To send output of ls command use

**Department of Computer Science and Engineering** **Page 7** **Aug – Dec 2020**

**$ ls > myfiles**  
Now if '**myfiles**' file exist in your current directory it will be overwritten without any type of warning.

(2) >> Redirector Symbol  
*Syntax:*  
Linux-command >> filename  
To output Linux-commands result (output of command or shell script) to END of file. Note that if file exists, it will be opened and new information/data will be written to END of file, without losing previous information/data. If file does not exist, then a new file is created. For e.g. To send output of date command to already exist file use command  
**$ date >> myfiles**

(3) < Redirector Symbol  
*Syntax:*  
Linux-command < filename  
To take input to Linux-command from file instead of key-board. For e.g. To take input for cat command use command  
**$ cat < myfiles**

**Exercise 5:**

**What does the following command do?**

**$ sort < myfile > sorted\_file**

**Pipes**

A pipe is a way to connect the output of one program to the input of another program without any temporary file.  
"*A pipe is nothing but a temporary storage place where the output of one command is stored and then passed as the input for the second command. Pipes are used to run more than two commands (Multiple commands) from the same command line.*"

*Syntax:*  
command1 | command2

**Department of Computer Science and Engineering** **Page 8** **Aug – Dec 2020**

|  |  |
| --- | --- |
| *Examples:* **Command using Pipes** | **Meaning or Use of Pipes** |
| **$ ls | more** | Output of **ls** command is given as input to **more** command so that output is printed one screen (full page) at a time. |
| **$ who | sort** | Output of **who** command is given as input to **sort** command so that it will print sorted list of users |
| **$ who | sort > user\_list** | Same as above except output of sort is sent to (redirected) user\_list file |
| **$ who | wc -l** | Output of **who** command is given as input to **wc** command so that it will show number of user logged into the system |
| **$ ls -l | wc  -l** | Output of **ls** command is given as input to **wc** command so that it will print number of files in current directory. |
| **$ who | grep ubuntu** | Output of who command is given as input to **grep** command so that it will print if particular user name is logged in, if not, nothing is printed |

If-else-fi construct

Create a shell script using if-else-fi statements and execute it

osch=0  
  
echo "1. Unix (Sun OS)"  
echo "2. Linux (Red Hat)"  
echo -n "Select your os choice [1 or 2]? "  
read osch  
  
if [ $osch -eq 1 ] ; then  
  
     echo "You selected Unix (Sun OS)"  
  
else #### nested if i.e. if within if ######  
              
       if [ $osch -eq 2 ] ; then  
             echo "You selected Linux (Red Hat)"  
       else  
             echo "You don't like Unix/Linux OS."  
       fi  
fi

**Department of Computer Science and Engineering** **Page 9** **Aug – Dec 2020**

**Looping – while, until and for Loops**

**Create a shell script to print Welcome 5 times**

for ((  i = 0 ;  i <= 4;  i++  ))  
do  
  echo "Welcome $i times"  
done

**Create a shell script to display numbers 0 to 9**

#!/bin/sh

a=0

while [ $a -lt 10 ]

do

echo $a

a=`expr $a + 1`

done

**Create a Shell script that uses the until loop to display the numbers 0 to 9**

#!/bin/sh

a=0

until [ ! $a -lt 10 ]

do

echo $a

a=`expr $a + 1`

done

**Exercise 6:**

Create a shell script (using Bourne Shell or Bash) which converts all file names in the current directory to lowercase. You should execute the script and send a screenshot of the output.

**Example**:

If you have these files in the current directory: ABC, foo1.c, Test, sample, foo2.TXT, myFile.Xa

Expected output after running the script: abc, foo1.c, test, sample, foo2.txt, myfile.xa

**References:** [**http://www.freeos.com/guides/lsst/**](http://www.freeos.com/guides/lsst/)

**https://linuxhint.com/30\_bash\_script\_examples/**

**‘Man’ pages of individual commands on Linux system**

**Department of Computer Science and Engineering** **Page 10** **Aug – Dec 2020**