DHA Suffa University
CS 206 – Operating Systems – Lab
Fall 2017

Lab 03 – Shell Programming using vi in Linux

Twitter: @oslabatdsu

Objective(s):

- Writing Shell Script
- Running a Shell Script
- Running Arithmetic Operations in Shell Script
- User Defined Variables
- Conditional Statements
- Nested Conditional Statements
- Multi-level Conditional Statements

What is Shell Script ?

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

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

Why to Write Shell Script ?

- Shell script can take input from user, file and output them on screen.
- Useful to create our own commands.
- Save lots of time.
- To automate some task of day today life.
- System Administration part can be also automated.

How to write shell script

Following steps are required to write shell script:
(1) Use any editor like vi or mcedit to write shell script.
(2) Save your script as *filename*.sh
(3) Execute your script as
syntax:
bash *filename*.sh

Example:

```
bash bar.sh
```

Variables in Shell

To process our data/information, data must be kept in computers RAM memory. RAM memory is divided into small locations, and each location had unique number called memory location/address, which is used to hold our data. Programmer can give a unique name to this memory location/address called memory variable or variable (Its a named storage location that may take different values, but only one at a time).

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.

You can see system variables by giving command like **$ set**, 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/vivek	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/students/Common	Our current working directory
SHELL=/bin/bash	Our shell name
USERNAME=vivek	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:

```
$ echo $USERNAME
$ echo $HOME
```

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 'vech' having value Bus

```
$ vech=Bus
```

To define variable called n having value 10

```
$ n=10
```

How to print or access value of UDV (User defined variables)

To print or access UDV use following syntax

Syntax:

$variablename

Define variable vech and n as follows:

```
$ vech=Bus
$ n=10
```

To print contains of variable 'vech' type

```
$ echo $vech
```

It will print 'Bus',To print contains of variable 'n' type command as follows

```
$ echo $n
```

Caution: Do not try **$ echo vech**, as it will print vech instead its value 'Bus' and **$ echo n**, as it will print n instead its value '10', You must *use $ followed by variable name*.

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.

-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"**

Shell Arithmetic

Use to perform arithmetic operations.

Syntax:
expr op1 math-operator op2

Examples:

```
$ expr 1 + 3
$ expr 2 - 1
$ expr 10 / 2
$ expr 20 % 3
$ expr 10 \* 3
$ echo `expr 6 + 3`
```

Note:
expr 20 %3 - Remainder read as 20 mod 3 and remainder is 2.
expr 10 * 3 - Multiplication use * and not * since its wild card.

For the last statement not the following points

(1) First, before expr keyword we used ` (back quote) sign not the (single quote i.e. ') sign. Back quote is generally found on the key under tilde (~) on PC keyboard OR to the above of TAB key.
(2) Second, expr is also end with ` i.e. back quote.
(3) Here expr 6 + 3 is evaluated to 9, then echo command prints 9 as sum
(4) Here if you use double quote or single quote, it will NOT work
For e.g.
$ echo "expr 6 + 3" # It will print expr 6 + 3
$ echo 'expr 6 + 3' # It will print expr 6 + 3

The read Statement

Use to get input (data from user) from keyboard and store (data) to variable.

Syntax:
read variable1, variable2,...variableN

Following script first ask user, name and then waits to enter name from the user via keyboard. Then user enters name from keyboard (after giving name you have to press ENTER key) and entered name through keyboard is stored (assigned) to variable fname.

if condition

if condition which is used for decision making in shell script, If given condition is true then command1 is executed.

Syntax:

```
if condition
then
        command1 if condition is true or if exit status
        of condition is 0 (zero)
        ...
fi
```

Condition is defined as: "*Condition is nothing but comparison between two values.*"
For compression you can use test or [expr] statements or even exist status can be also used.
Relational Operators

Operator	Description	Example
-eq	Checks if the value of two operands are equal or not; if yes, then the condition becomes true.	[$a -eq $b] is not true.
-ne	Checks if the value of two operands are equal or not; if values are not equal, then the condition becomes true.	[$a -ne $b] is true.
-gt	Checks if the value of left operand is greater than the value of right operand; if yes, then the condition becomes true.	[$a -gt $b] is not true.
-lt	Checks if the value of left operand is less than the value of right operand; if yes, then the condition becomes true.	[$a -lt $b] is true.
-ge	Checks if the value of left operand is greater than or equal to the value of right operand; if yes, then the condition becomes true.	[$a -ge $b] is not true.
-le	Checks if the value of left operand is less than or equal to the value of right operand; if yes, then the condition becomes true.	[$a -le $b] is true.

Note: Every command i.e. variable value access, relational operator, etc should be separated by a whitespace.

test command or [expr]

test command or [expr] is used to see if an expression is true, and if it is true it return zero(0), otherwise returns nonzero for false.

Syntax:

test expression OR [expression]

if...else...fi

If given condition is true then command1 is executed otherwise command2 is executed.

Syntax:

```
if condition
then

            #execute all commands up to else statement
else

            #execute all commands up to fi
fi
```

Nested if

Syntax:

```
if condition
then
        if condition
        then
                .....
                do this
        else
                ....
                do this
        fi
else
        ...
        do this
fi
```

Multilevel if-then-else

Syntax:

```
if condition
then
            #execute all commands up to elif statement
elif condition1
then
            #execute all commands up to elif statement
elif condition2
then
            #execute all commands up to elif statement
else
            #None of the above condtion,condtion1,condtion2 are true
            #execute all commands up to fi
Fi
```

Lab Tasks:

1. **Write a Shell Script using vi to check whether a number is positive-even, positive-odd, negative-even, negative-odd. (Ignore zero. Consider user does not enter zero at any time).**

Lab Assignment 03

1. **Prompt the user for his/her year of birth and if the year is a leap year print: "You were born in a leap year", otherwise print the age of the user.**

2. **Take obtained marks of a student out of 100 as input and print the grade as per following rules.**

Obtained Marks	Grade
Between and including 0 and 50	F
Between and including 51 and 80	B
Between and including 81 and 100	A

Submission Instructions:

1. **Number your C++ files as question number e.g. Q1.cpp, Q2.cpp, etc. (Q is in upper case)**
2. **For every cpp file, there should be a word file containing at least ten snapshots on different stages of editing the code in vi.**
3. **Create a new folder named cs152abc where abc is your 3 digit roll #. e.g. cs152111.**
4. **Copy all the C++ files and word files into this folder.**
5. **Right-click on the folder you created and create a zip file by selecting the option**
   - **"Send to" and selecting "Compressed (zipped) folder" [for windows].**
   - **"Create Archive" and change option to ".zip" instead of ".tar.gz" and click on "Create". [for linux]**

   Now make sure a zip file named cs152abc.zip is created e.g. *cs152111.zip*.
6. **(A) Compose a new email, attach this zip file and send email to *oslabatdsu@gmail.com*.**
   - **The subject of the email must be:**

     4A-Lab03-cs152abc **where 4A is the section you're enrolled in and abc is your 3 digit unique roll#.**

     4A-Lab03-cs152001

   (B) The body of the email must contain:

   Full Name
   DSU Roll #
   Section
   Lab Number #
7. **To double check whether your assignment was submitted or not make sure the Sent Items folder of your email account contains the email you just sent.**
8. **Due Date for assignment submission is Sept 17, 2017.**