Faculty of Information Technology
University of Moratuwa
Batch 21 – Level 2 Semester
Operating System Lab session03

OBJECTIVE: To study shell script and menu driven program

Variables in Linux Sometimes to process our data/information, it 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, there are two types of variable1) **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.

In every **programming** language **variables** plays an important role , in Linux shell scripting we are using two types of variables : **System Defined Variables** & **User Defined Variables**

User defined variables

Shells also allow you to create your own variables for use within scripts (local variables) and to pass between scripts (global variables). User variables are traditionally created using lower-case characters.

To create a variable merely choose a lower-case name for the variable and give it a value using an equal (=) sign. Make certain that there are **no spaces on either side of the equal sign**. Here are some examples.

Creating user variables name is myname and assigned value as 'Terry Clark'

To define UDV use following syntax,
Syntax:

variable name=value

$ myname='Terry Clark'

$ echo myname

myname

To print or access UDV use following syntax,
$ echo $myname

Output:
Terry Clark

To define variable called number having value 10
$ number=10

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

Output:
10

About Quotes:

Quotes	Name	Meaning
"	Double Quotes	"Double Quotes" - Anything enclose in double quotes removed meaning of that characters (except \ and $).
'	Single quotes	'Single quotes' - Enclosed in single quotes remains unchanged.
`	Back quote	`Back quote` - To execute command

Example:
$ echo "Today is date"
Can't print message with today's date.

$ echo "Today is `date`".

Rules for Naming variable name (Both UDV and System Variable)

(1) Variable name must begin with Alphanumeric character or underscore character (_), followed by one or more Alphanumeric character. For e.g. Valid shell variable are as follows:

HOME
SYSTEM_VERSION

(2) Don't put spaces on either side of the equal sign when assigning value to variable. For e.g. In following variable declaration there will be no error

$ number =10

Note that,
But there will be problem for any of the following variable declaration:

$ number =10
$ number = 10
$ number = 10

(3) Variables are case-sensitive, just like filename in Linux.

(4) You can define NULL variable as follows (NULL variable is variable which has no value at
the time of definition) For e.g.

$ var1=
$ var2=""
Try to print it's value by issuing following command

$ echo $var1
Nothing will be shown because variable has no value i.e. NULL variable.

(5) Do not use ?,* etc, to name your variable names.

The variables created above are local variables available only to the current shell. We must export variables if we wish them to become **global** and can be referenced by other shell scripts.

Creating global user variables

$ myname=Terry

$ export myname

Or

$ export myname=Terry

System Defined Variables

These are the variables which are created and maintained by **Operating System(Linux) itself.** Generally these variables are defined in **CAPITAL LETTERS**. We can see these variables by using the command "**$ set**". Some of the system defined variables are given below :

System Defined Variables	Meaning
BASH=/bin/bash	Shell Name
BASH_VERSION=4.1.2(1)	Bash Version
COLUMNS=80	No. of columns for our screen
HOME=/home/linuxtechi	Home Directory of the User
LINES=25	No. of columns for our screen
LOGNAME=LinuxTechi	LinuxTechi Our logging name
OSTYPE=Linux	OS type
PATH=/usr/bin:/sbin:/bin:/usr/sbin	Path Settings
PS1=[\u@\h \W]\$	Prompt Settings
PWD=/home/linuxtechi	Current Working Directory
SHELL=/bin/bash	Shell Name
USERNAME=linuxtechi	User name who is currently login to system

To Print the value of above variables, use **echo command** as shown below :

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

Arithmetic Expansion in Bash Shell

Arithmetic expansion and evaluation is done by placing an integer expression using the following format:

Syntax:

expr op1 math-operator op2

```
$((expression))
$(( n1+n2 ))
$(( n1/n2 ))
$(( n1-n2 ))
```

Examples

Add two numbers on fly using the echo command:

```
echo $(( 10 + 5 ))
```

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.

Add two numbers using x and y variable. Create a shell program called add.sh using a text editor:

Examples:

1)

```
#!/bin/bash
x=5
y=10
ans=$(( x + y ))
echo "$x + $y = $ans"
```

Save and close the file. Run it as follows:

Excute add.sh file first to set permission for your script add.sh
chmod +x add.sh

Run add.sh file:
./add.sh

Sample Outputs:

```
5 + 10 = 15
```

Create an interactive program using the read command called add1.sh using a text editor:

2)

```
#!/bin/bash
read -p "Enter two numbers : " x y
ans=$(( x + y ))
echo "$x + $y = $ans"
```

Save and close the file. Run it as follows:

chmod +x add1.sh
./add1.sh

Sample Outputs:

Enter two numbers : 20 30
20 + 30 = 50

3)

#!/bin/bash
a=2
echo $((a+3))
b=$((2* ++a))
echo $b
echo $((a*b))

Save and close the file. Run it as follows:

chmod +x ex3.sh
./ex3.sh

4) Read- read input from terminal assign to variable

#!/bin/bash
Echo “Enter name:”
read name
echo “Entered name: $name”

- **Multiple names**

#!/bin/bash
echo “Enter name:”
read name1 name2 name3
echo “Name: $name1, $name2, $name3”

- **Extract multiple names in a array**

#!/bin/bash
echo “Enter name:”
read –a names
echo “Name: {$name[0], $name[1], $name[2]}”

Mathematical Operators With Integers

Operator	Description	Example	Evaluates To
+	Addition	echo $((20 + 5))	25
-	Subtraction	echo $((20 - 5))	15
/	Division	echo $((20 / 5))	4
*	Multiplication	echo $((20 * 5))	100
%	Modulus	echo $((20 % 3))	2
++	post-increment (add variable value by 1)	x=5 echo $((x++)) echo $((x++))	5 6
--	post-decrement (subtract variable value by 1)	x=5 echo $((x--))	4
**	Exponentiation	x=2 y=3 echo $((x ** y))	8

For Mathematics, use following operator in Shell Script

Mathematical Operator in Shell Script	Meaning	Normal Mathematical Statements	But in Shell	
			For test statement with if command	For [expr] statement with if command
-eq	is equal to	5 == 6	if test 5 -eq 6	if [5 -eq 6]
-ne	is not equal to	5 != 6	if test 5 -ne 6	if [5 -ne 6]
-lt	is less than	5 < 6	if test 5 -lt 6	if [5 -lt 6]
-le	is less than or equal to	5 <= 6	if test 5 -le 6	if [5 -le 6]
-gt	is greater than	5 > 6	if test 5 -gt 6	if [5 -gt 6]
-ge	is greater than or equal to	5 >= 6	if test 5 -ge 6	if [5 -ge 6]