

The topics:

- Variables in Shell
- *Shell Arithmetic*
- The Read Statement
- if condition
- test command or [expr]
- if ...else...fi

In Linux, there are two types of variable:

(1) System variables :

Created and maintained by Linux itself. This type of variable defined in CAPITAL LETTERS.

Ex. \$HOME \$BASH \$PWD

(2) User defined variables (UDV) :

Created and maintained by user. This type of variable defined in lower letters.

How to define and print User Defined Variables:

Syntax to define UDV: ***variable name=value***

Syntax to print or access value of UDV : ***\$variablename***

Example:

- To define variable called 'vech' having value Bus and print contains of variable 'vech'

```
$ vech=Bus  
$ echo $vech
```

- To define variable called n having value 10 and print contains of variable 'n'

```
$ n=10  
$ echo $n
```

Rules for Naming variable:

- 1 begin with Alphanumeric character or underscore character (_), followed by one or more Alphanumeric character

```
HOME  
vech
```

- 2 Don't put spaces on either side of the equal sign

```
$ no=10  
$no =10 →wrong  
$no= 10 →wrong  
$ no = 10 →wrong
```

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

```
$ no=10  
$ No=11
```

- 4 You can define NULL variable as follows:

```
$ vech=  
$ vech=""
```

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

Shell Arithmetic :

Syntax:

expr op1 math-operator op2

expr 1 + 3

expr 2 - 1

expr 10 / 2

expr 20 % 3

expr 10 * 3

```
$ echo 6 + 3 → will print 6+3  
$ echo `expr 6 + 3`  
$ expr 6+3 → will not work because no space between number and operator
```

- define two variable x=20, y=5 and then to print division of x and y :

```
$x=20  
$ y=5  
$ expr $x / $y
```

- store division of x and y to variable called z :

```
$ x=20  
$ y=5  
$ z=`expr $x / $y`  
$ echo $z
```

The Read Statement :

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

Syntax:

read variable1, variable2,...variableN

Example:

```
$ vi sayH  
#  
#Script to read your name from key-board  
#  
echo "Your first name please:"  
read fname  
echo "Hello $fname, Lets be friend! "
```

Run it as follows:

```
$ chmod 755 sayH  
$ ./sayH
```

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...
fi
```

```
$ vi showfile
#Script to print file
#
if cat $1
then
echo -e "\n\nFile $1, found and successfully echoed"
fi
```

```
$ chmod 755 showfile
$ ./showfile file-name
```

```
$ vi trmif
# Script to test rm command and exist status
#
if rm $1
then
echo "$1 file deleted"
fi
```

```
$ chmod 755 trmif
$ ./trmif file-name
```

test command or [expr] :

--True → return zero(0)

Is used to see if an *expression* is true

--False → returns nonzero

Syntax:

test expression OR [expression]

```
$ vi ispositive
# Script to see whether argument is positive
#
if test $1 -gt 0
then
echo "$1 number is positive"
fi
```

Run it as follows:

```
$ chmod 755 ispositive
$ ./ispositive 5
```

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

String Comparisons	
Operator	Meaning
string1 = string2	string1 is equal to string2
string1 != string2	string1 is NOT equal to string2
string1	string1 is NOT NULL or not defined
-n string1	string1 is NOT NULL and does exist
-z string1	string1 is NULL and does exist

Logical Operators	
Operator	Meaning
! expression	Logical NOT
expression1 -a expression2	Logical AND
expression1 -o expression2	Logical OR

if ...else...fi

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

Syntax:

if *condition*

then

condition

else

if condition is not true then execute all commands up to fi

fi

\$ vi isnump_n
Script to see whether argument is positive or negative
if [\$# -eq 0]
then
echo "\$0 : You must give/supply one integers"
exit 1
fi
if test \$1 -ge 0
then
echo "\$1 number is positive"
else
echo "\$1 number is negative"
fi
\$ chmod 755 isnump_n
\$./isnump_n 5

See : <http://www.freeos.com/guides/oldsst/getting.htm>