Variables If-Else Loop Case Statements Operators

Sankar S

Echo Options

Options	Description
-n	do not print the trailing newline.
-e	enable interpretation of backslash escapes.
\b	backspace
\\	backslash
\n	new line
\r	carriage return
\t	horizontal tab
\v	vertical tab

Shell Variables

❖ Global Variables - environmental variables are called as global variable

❖ Local Variables - User defined variables will exists till end of the program. Variables can be exported to other shell programs

❖ **Variables by content**

1. String variables
2. Integer variables
3. Constant variables
4. Array variables

Rules for Defining variable name

(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
vech
No

(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

(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)

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

Conditional If-Else Statements

In Bash, we have the following conditional statements:

- ❖ if..then..fi statement (Simple If)
- ❖ if..then..else..fi statement (If-Else)
- ❖ if..elif..else..fi statement (Else If ladder)
- ❖ if..then..else..if..then..fi..fi..(Nested if)

Loop Statements

In Bash, we have the following loop statements:

- ❖ For do .. Done

- ❖ While do .. Done   -- executes till conditional expr is true

- ❖ Until loop -- executes till conditional expr is false

Operators

In Bash, we have the following operators:

- ❖ Arithmetic Operators
- ❖ Relational Operators
- ❖ Boolean Operators
- ❖ String Operators
- ❖ File Test Operators
- ❖ Assignment Operator

Arithmetic Operators shown here

Here a = 10 b = 10

Operator	Description	Example
+	Addition - Adds values on either side of the operator	\`expr $a + $b\` will give 30
-	Subtraction - Subtracts right hand operand from left hand operand	\`expr $a - $b\` will give -10
*	Multiplication - Multiplies values on either side of the operator	\`expr $a \\* $b\` will give 200
/	Division - Divides left hand operand by right hand operand	\`expr $b / $a\` will give 2
%	Modulus - Divides left hand operand by right hand operand and returns remainder	\`expr $b % $a\` will give 0
=	Assignment - Assign right operand in left operand	a=$b would assign value of b into a
==	Equality - Compares two numbers, if both are same then returns true.	[$a == $b] would return false.
!=	Not Equality - Compares two numbers, if both are different then returns true.	[$a != $b] would return true.

Operators

❖ Relational Operators

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

Operators

❖ Boolean Operators

Operator	Description	Example
!	This is logical negation. This inverts a true condition into false and vice versa.	[! false] is true.
-o	This is logical OR. If one of the operands is true then condition would be true.	[$a -lt 20 -o $b -gt 100] is true.
-a	This is logical AND. If both the operands are true then condition would be true otherwise it would be false.	[$a -lt 20 -a $b -gt 100] is false.

Operators

❖ String Operators

Operator	Description	Example
=	Checks if the value of two operands are equal or not, if yes then condition becomes true.	[$a = $b] is not true.
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	[$a != $b] is true.
-z	Checks if the given string operand size is zero. If it is zero length then it returns true.	[-z $a] is not true.
-n	Checks if the given string operand size is non-zero. If it is non-zero length then it returns true.	[-z $a] is not false.
str	Check if str is not the empty string. If it is empty then it returns false.	[$a] is not false.

Operators

❖ File Test Operators

Operator	Description	Example
-b file	Checks if file is a block special file if yes then condition becomes true.	[-b $file] is false.
-c file	Checks if file is a character special file if yes then condition becomes true.	[-c $file] is false.
-d file	Check if file is a directory if yes then condition becomes true.	[-d $file] is not true.
-f file	Check if file is an ordinary file as opposed to a directory or special file if yes then condition becomes true.	[-f $file] is true.
-g file	Checks if file has its set group ID (SGID) bit set if yes then condition becomes true.	[-g $file] is false.
-k file	Checks if file has its sticky bit set if yes then condition becomes true.	[-k $file] is false.
-p file	Checks if file is a named pipe if yes then condition becomes true.	[-p $file] is false.
-t file	Checks if file descriptor is open and associated with a terminal if yes then condition becomes true.	[-t $file] is false.
-u file	Checks if file has its set user id (SUID) bit set if yes then condition becomes true.	[-u $file] is false.
-r file	Checks if file is readable if yes then condition becomes true.	[-r $file] is true.
-w file	Check if file is writable if yes then condition becomes true.	[-w $file] is true.
-x file	Check if file is execute if yes then condition becomes true.	[-x $file] is true.
-s file	Check if file has size greater than 0 if yes then condition becomes true.	[-s $file] is true.
-e file	Check if file exists. Is true even if file is a directory but exists.	[-e $file] is true.

Operators & Case Statements

- ❖ Assignment operator

  Directly assign value from one variable to another

- ❖ Case Statements

  This is substitute of switch-case statement