# UNIX - SHELL BASIC OPERATORS

 

There are various operators supported by each shell. Our tutorial is based on default shell *Bourne* so we are going to cover all the important Bourne Shell operators in the tutorial.

There are following operators which we are going to discuss −

- Arithmetic Operators.
- Relational Operators.
- Boolean Operators.
- String Operators.
- File Test Operators.

The Bourne shell didn't originally have any mechanism to perform simple arithmetic but it uses external programs, either **awk** or the must simpler program **expr**.

Here is simple example to add two numbers −

```
#!/bin/sh

val=`expr 2 + 2`
echo "Total value : $val"
```

This would produce following result −

```
Total value : 4
```

There are following points to note down −

- There must be spaces between operators and expressions for example 2+2 is not correct, where as it should be written as 2 + 2.
- Complete expression should be enclosed between ``, called inverted commas.

## Arithmetic Operators

There are following arithmetic operators supported by Bourne Shell.

Assume variable a holds 10 and variable b holds 20 then −

Show Examples

| 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. |

It is very important to note here that all the conditional expressions would be put inside square braces with one spaces around them, for example [ $a$ == b ] is correct where as [$a$ == b] is incorrect.

All the arithmetical calculations are done using long integers.

## Relational Operators:

Bourne Shell supports following relational operators which are specific to numeric values. These operators would not work for string values unless their value is numeric.

For example, following operators would work to check a relation between 10 and 20 as well as in between "10" and "20" but not in between "ten" and "twenty".

Assume variable a holds 10 and variable b holds 20 then −

Show Examples

| 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. | [ $a - le$b ] is true. |

It is very important to note here that all the conditional expressions would be put inside square braces with one spaces around them, for example [ $a$ <= b ] is correct where as [$a$ <= b] is incorrect.

## Boolean Operators

There are following boolean operators supported by Bourne Shell.

Assume variable a holds 10 and variable b holds 20 then −

Show Examples

| 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. |

## String Operators

There are following string operators supported by Bourne Shell.

Assume variable a holds "abc" and variable b holds "efg" then −

Show Examples

| 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. |

## File Test Operators

There are following operators to test various properties associated with a Unix file.

Assume a variable **file** holds an existing file name "test" whose size is 100 bytes and has read, write and execute permission on −

Show Examples

| 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. |

## C Shell Operators

Following link would give your brief idea on C Shell Operators.

C Shell Operators

## Korn Shell Operators

Following link would give your brief idea on Korn Shell Operators.

Korn Shell Operators