**EXPR COMMAND EXAMPLES IN UNIX / LINUX TUTORIALS**

This continuation to my previous post on [bc command - calculator](http://www.folkstalk.com/2012/09/bc-command-examples-in-unix-linux.html). In this article we will see how to use the expr command in unix or linux system for doing arithmetic operations.  
  
The syntax of expr command is 

expr [expression]

Let see how to use the expr command in unix or linux with examples. Most of the examples are straightforward. I will provide description as and when required.  
  
  
**Note**: You have to provide the space between the values and the operands. Otherwise the expr command may throw error or print them as a string.   
  
**Arithmetic Operator Examples**:  
  
1. Sum of numbers 

$ expr 5 + 3

8

$ expr 1 + 2 + 3

6

$ expr 5+3

5+3

Here in the third expr command, space is not provided between the literals. The expr command treated it as a string and printed on the terminal.   
  
2. Difference between two numbers 

$ expr 10 - 6

4

3. Multiplying numbers 

$ expr 7 \\* 9

63

Here the \* is shell builtin operator, that is why it needs to escaped with backslash.   
  
4. Dividing numbers 

$ expr 6 / 4

1

The division operator returns only the arithmetic quotient.   
  
5. Remainder or modulus 

$ expr 6 % 4

2

**Comparision or Relational Operator Examples**:   
  
You can use the following comparision operators with the expr command:

* Val1 < Val2 : Returns 1 if val1 is less than val2. otherwise zero.
* Val1 <= Val2 : Returns 1 if val1 is less than or equal to val2. otherwise zero.
* Val1 > Val2 : Returns 1 if val1 is greater than val2. otherwise zero.
* Val1 >= Val2 : Returns 1 if val1 is greater than or equal to val2. otherwise zero.
* Val1 = Val2 : Returns 1 if val1 is equal to val2. otherwise zero.
* Val1 != Val2 : Returns 1 if val1 is equal to val2. otherwise zero.
* val1 | val2 : Returns val1 if val1 is neither null nor zero. Otherwise val2.
* val1 & val2 : Returns val1 if both val1 and val2 is neither null nor zero. Otherwise 0.

**Note**: You have to escape most of the operators with backslash as they are shell built in. 

$ expr 1 \< 2

1

$ expr 1 \<= 1

1

$ expr 2 \> 5

0

$ expr 2 \>= 5

0

$ expr 7 = 7

1

$ expr 9 != 18

1

$ expr 2 \| 5

2

$ expr 0 \| 5

5

$ expr 2 \& 5

2

$ expr 6 \& 3

6

$ expr 6 \& 0

0

$ expr 0 \& 3

0

**String Function Examples**:   
  
1. Length of string  
  
The length function is used to find the number of characters in a string. 

$ expr length linux

5

$expr length linux\ system

12

$expr length "linux system"

If you have spaces in your string escape them with backslash or quote them with double quotes.   
  
2. Find Substring   
  
You can extract a portion of the string by using the substr function. The syntax of substr function is

substr string position length

Here position is the character position in the string. length is the number of chracters to extract from the main string. An example is shown below:

$ expr substr unixserver 5 6

server

3. Index of the substring   
  
You can find the position of a string in the main string using the index function. The syntax of index function is shown below:

index string chars

If the chars string is found in the main string, then the index function returns the position of the chars. Otherwise it returns 0. See the following examples:

$ expr index linux nux

3

$expr index linux win

0

4. Matching a regexp   
  
The match function is used to find anchored pattern match of regexp in the string. The syntax of match function is shown below:

match string pattern

The match function returns the number of characters in the pattern is a match is found. Otherwise, it returns 0. Alternative synatx is

string : pattern

The following examples shows how to use the match function:

$ expr match linuxserver lin

3

$ expr match linuxserver server

0

Here in the second expr, the pattern (server) exists in the main string. However the pattern does not start from the beggining of the main string. Thats why the match function returns 0.