GETTING STARTED
WITH
SHELL OPERATORS

➢ SHELL OPERATORS:

- There are various operators supported by each shell. We will discuss in detail about Bourne shell (default shell).
- Let us now see how we can operate upon these values you may recall that all shell variables are string variables.

ARITHMETIC OPERATORS:

+ (Addition)
- (Subtraction)
* (Multiplication)
/ (Division)
% (Modulus)

NOTE: If we are to carry out arithmetic operators on them, we have to use the command **'expr'** which is capable of evaluating an arithmetic expression.

RELATIONAL OPERATORS:

- Relational Operators are two types:
  - Numerical Comparison Operators
  - String Comparison Operators

NUMERICAL COMPARISON OPERATORS:

-lt	(Less than)
-le	(Less than or equal to)
-gt	(Greater than)
-ge	(Greater than or equal to)
-eq	(equal to)
-ne	(Not equal to)

STRING COMPARISON OPERATORS:

<	(Less than)
>	(Greater than)
=	(Equal to)
!=	(Not equal)

ASSIGNMENT OPERATOR:

=	(equal)

LOGICAL OPERATORS (BOOLEAN OPERATORS):

-a (Logical and)
-o (Logical or)
! (Logical not)

BITWISE OPERATORS:

& : Bitwise And
| : Bitwise OR
^ : Bitwise XOR
~ : Bitwise compliment
<< : Left Shift
>> : Right Shift

NOTE: Each and every operator, should contain space before and after operator except assignment operator.

Example 1: a=10; b=4;

```
echo `expr $a + $b`
echo `expr $a - $b`
echo `expr $a \* $b`
echo `expr $a / $b`
echo `expr $a % $b` #Modular division, returns remainder.
```

Example 2: a=30 b=15 c=2 d=5

```
$echo `expr $a \* \( $b + $c \) / $d`
```

NOTE: **expr** is capable of carrying out only integer arithmetic to carry out arithmetic on real numbers or float arithmetic it is necessary to use the **'bc'** command.

Example of float vales: p=10.5; q=3.5;

```
echo `echo $a + $b` | bc
echo `echo $a - $b` | bc
echo `echo $a \* $b` | bc
echo `echo $a / $b` | bc
echo `echo $a % $b` | bc
```

NOTE: **expr $a + $b** is a legal expression whereas **bc $a + $b** isn't. hence, we have piped the result of echo to bc.

Example 2: x=10.5 y=3.5

```
c=`echo $a + $b | bc`
d=`echo $a - $b | bc`
e=`echo $a \* $b | bc`
f=`echo $a / $b | bc`
```

To get a value: $echo $c $d $e $f

➢ THE PROCESS OF USING "echo" WITH \r, \n, \t…ETC.:

THE CARRIAGE RETURN (\r):

The \r is called the carriage return. It causes the cursor to be positioned at the beginning of the current line.

$echo "I LIKE WORK...\r I CAN SIT AND WATCH IT FOR HOURS

NEW LINE(\N):

By default, every echo statement echoes the output on a fresh line. If we want that output of a single echo statement should be split across lines, we can use the newline escape sequence as shown below.

$echo "I LIKE WORK....\n I CAN SIT AND WATCH IT FOR HOURS

TAB AND BACKSPACE (\t and \b):

The function of the tab key is emulated by the sequence \t, and that of the backspace, by \b.

$echo "There is always one more \b\b\b\b\b bug. \t\t -By Law.

There is always one bug. -By Law

POSITIONING THE CURSOR (\C):

By default, after an echo statement, the cursor is placed at the beginning of the next line.

$echo "Enter your choice....\c"

The cursor waits after the ellipsis and not on the next line.