

## **BASH ARITHMETIC OPERATORS**

In this topic, we will understand how to use arithmetic operators in Bash.

Depending on what type of result we want through our scripts, we may need to apply arithmetic operators at some point. Like variables, they are reasonably easy to apply. In the bash script, we can perform arithmetic operations on numeric values to get the desired result.

There are 11 arithmetic operators which are supported by Bash Shell.

Look at the following table demonstrating the syntax, description and examples for each of the arithmetic operators:

Operator	Description	Examples
+	Addition, measures addition of numbers (operands)	<i><code>\$(( 10 + 3 ))</code>, result=13</i>
-	Subtraction, measures subtraction of second operand from first	<i><code>\$(( 10 - 3 ))</code>, result=7</i>
*	Multiplication, measures the multiplication of operands.	<i><code>\$(( 10 * 3 ))</code>, result=30</i>
/	Division, measures the division of first operand by second operand and return quotient.	<i><code>\$(( 10 / 3 ))</code>, result=3</i>
**	Exponentiation, measures the result of second operand raised to the power of first operand.	<i><code>\$(( 10 ** 3 ))</code>, result=1000</i>

%	Modulo, measures remainder when the first operand is divided by second operand.	<i>\$(( 10 % 3 )), result=1</i>
+=	Increment Variable by Constant- used to increment the value of first operand by the constant provided.	<i>x=10 let "x += 3" echo \$x result=13</i>
-=	Decrement Variable by Constant- used to decrement the value of first operand by the constant provided.	<i>x=10 let "x -= 3" echo \$x result=7</i>
*=	Multiply Variable by Constant- used to multiply the value of the first operand by the constant provided.	<i>x=10 let "x *= 3" echo \$x result=30</i>
/=	Divide Variable by Constant- used to calculate the value of (variable / constant) and store the result back to variable.	<i>x=10 let "10 /= 3" echo \$x result=3</i>
%=	Remainder of Dividing Variable by Constant- used to calculate the value of (variable %	<i>x=10 let "10 %= 3" echo \$x</i>

constant) and store the result      *result=1*  
back to variable.

## **Performing Arithmetic Operations in Bash**

There are many options to perform arithmetic operations on the bash shell. Some of the options are given below that we can adopt to perform arithmetic operations:

### **Double Parentheses**

Double parentheses is the easiest mechanism to perform basic arithmetic operations in the Bash shell. We can use this method by using double brackets with or without a leading \$.

### **Syntax**

1. ((expression))

We can apply four different ways to achieve the required objectives. Look at the methods given below to understand how the double parentheses mechanism can be used (**Assuming that we want to add the numbers 10 and 3**) :

### **Method 1**

1. Sum=\$((10+3))

2. echo "Sum = \$Sum"

### **Method 2**

1. ((Sum=10+3))

2. echo "Sum = \$Sum"

### **Method 3**

1. Num1=10
2. Num2=3
3. ((Sum=Num1+Num2))
4. echo "Sum = \$Sum"

#### Method 4

1. Num1=10
2. Num2=3
3. Sum=\$((Num1+Num2))
4. echo "Sum = \$Sum"

All of these methods will provide the same output as:

*Sum = 13*

Following is an example demonstrating the use of double parentheses for arithmetic operations in a Bash shell script:

#### Bash Script

- #!/bin/bash
- x=8
- y=2
- echo "x=8, y=2"
- echo "Addition of x & y"
- echo \$(( \$x + \$y ))
- echo "Subtraction of x & y"
- echo \$(( \$x - \$y ))
- echo "Multiplication of x & y"
- echo \$(( \$x \* \$y ))

```
➤ echo "Division of x by y"
➤ echo $(( $x / $y ))
➤ echo "Exponentiation of x,y"
➤ echo $(( $x ** $y ))
➤ echo "Modular Division of x,y"
➤ echo $(( $x % $y ))
➤ echo "Incrementing x by 5, then x= "
➤ (( x += 5 ))
➤ echo $x
➤ echo "Decrementing x by 5, then x= "
➤ (( x -= 5 ))
➤ echo $x
➤ echo "Multiply of x by 5, then x="
➤ (( x *= 5 ))
➤ echo $x
➤ echo "Dividing x by 5, x= "
➤ (( x /= 5 ))
➤ echo $x
➤ echo "Remainder of Dividing x by 5, x="
➤ (( x %= 5 ))
➤ echo $x
```

## Output

```
Terminal
File Edit View Search Terminal Help
javatpoint@javatpoint:~$ ./example.sh
./example.sh: line 1: Bash: command not found
x=8, y=2
Addition of x & y
10
Subtraction of x & y
6
Multiplication of x & y
16
Division of x by y
4
Exponentiation of x,y
64
Modular Division of x,y
0
Incrementing x by 5, then x=
13
Decrementing x by 5, then x=
8
Mulitply of x by 5, then x=
40
Dividing x by 5, x=
8
Remainder of Dividing x by 5, x=
3
javatpoint@javatpoint:~$
```

## Let Construction

Let is a built-in command of Bash that allows us to perform arithmetic operations. It follows the basic format:

### Syntax

1. `let <arithmetic expression>`

An example is given below explaining how to use let command in a Bash script:

### Bash script

```
➤ #!/bin/bash
```

```
➤
```

```
➤ x=10
```

```
➤ y=6
```

```
➤ z=0
```

➤ echo "Addition"

➤ let "z =  $\$( (x + y) )$ "

➤ echo "z= \$z"

➤

➤ echo "Substraction"

➤ let "z =  $\$( (x - y) )$ "

➤ echo "z= \$z"

➤

➤ echo "Multiplication"

➤ let "z =  $\$( (x * y) )$ "

➤ echo "z = \$z"

➤

➤ echo "Division"

➤ let "z =  $\$( (x / y) )$ "

➤ echo "z = \$z"

➤

➤ echo "Exponentiation"

➤ let "z =  $\$( (x ** y) )$ "

➤ echo "z = \$z"

➤

➤ echo "Modular Division"

➤ let "z =  $\$( (x \% y) )$ "

➤ echo "z = \$z"

➤

➤ `let "x += 5"`

➤ `echo "Incrementing x by 5, then x= "`

➤ `echo $x`

➤

➤ `let "x -= 5"`

➤ `echo "Decrementing x by 5, then x= "`

➤ `echo $x`

➤

➤ `let "x *=5"`

➤ `echo "Multiply of x by 5, then x="`

➤ `echo $x`

➤

➤ `let "x /= 5"`

➤ `echo "Dividing x by 5, x= "`

➤ `echo $x`

➤

➤ `let "x %= 5"`

➤ `echo "Remainder of Dividing x by 5, x="`

➤ `echo $x`

## Output



```
Terminal
File Edit View Search Terminal Help
javatpoint@javatpoint:~$ ./let_example.sh
Addition
z= 16
Substraction
z= 4
Multiplication
z = 60
Division
z = 1
Exponentiation
z = 1000000
Modular Division
z = 4
Incrementing x by 5, then x=
15
Decrementing x by 5, then x=
10
Multiply of x by 5, then x=
50
Dividing x by 5, x=
10
Remainder of Dividing x by 5, x=
0
javatpoint@javatpoint:~$
```

## BACKTICKS

In bash scripting, an arithmetic expansion can also be performed using **backticks** and `expr` (known as all-purpose expression evaluator). The ``expr`` is similar to `'let,'` but it does not save the result to a variable. It directly prints the result. Unlike `let`, we don't need to enclose the expression within the quotes. We are required to use spaces between the items of the expression. It is important to note that we should use `'expr`` within command substitution to save the output to a variable.

We can also use ``expr`` without `'backticks'`.

### Syntax

➤ ``expr item1 operator item2``

or

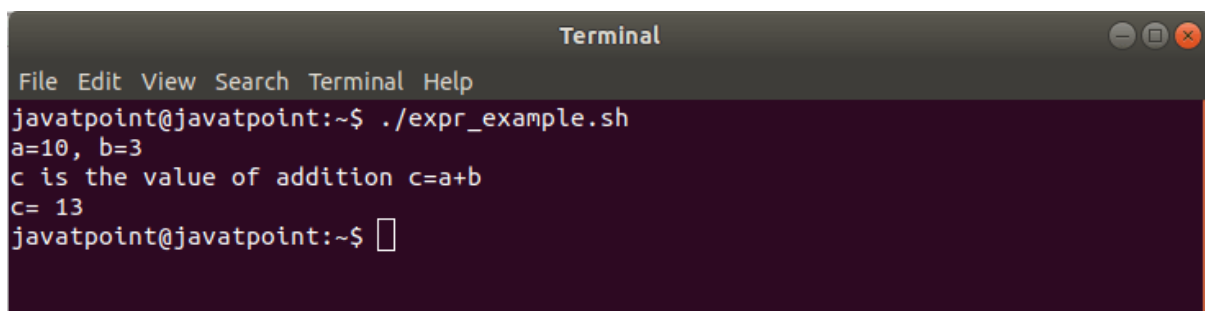
➤ `expr item1 operator item2`

An example is given below explaining how to use backticks and expr in a Bash script:

### Bash Script Program

```
➤ #!/bin/bash
➤ #Basic arithmetic using expr
➤
➤ echo "a=10, b=3"
➤ echo "c is the value of addition c=a+b"
➤ a=10
➤ b=3
➤ echo "c= `expr $a + $b`"
```

### Output

A screenshot of a terminal window titled "Terminal". The window has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal content shows the execution of a script: "javatpoint@javatpoint:~\$ ./expr\_example.sh", followed by the script's output: "a=10, b=3", "c is the value of addition c=a+b", and "c= 13". The prompt "javatpoint@javatpoint:~\$" is shown again at the end.

```
Terminal
File Edit View Search Terminal Help
javatpoint@javatpoint:~$ ./expr_example.sh
a=10, b=3
c is the value of addition c=a+b
c= 13
javatpoint@javatpoint:~$
```