Bash Arithmetic Operations

Bash Shell enables you to perform arithmetic operations with both integers and floating-point numbers easily. But the way of performing arithmetic operations is very different from other programming languages like C, C++, Java, etc.

There are four ways of performing arithmetic operations in Bash-

1. **Using double parenthesis $(())**
2. **Using square bracket $[]**
3. **Bash expr command**
4. **Bash bc command**

> **Note:** Double parenthesis, square bracket and expr support integer arithmetic. If you want to perform floating-point arithmetic then use bash bc command.

Arithmetic Operations using double parenthesis

- Double Parenthesis is a built-in arithmetic feature of Bash shell.
- It is used to perform integer arithmetic operations.
- In this, you do not have to use $variable to access a value of the variable inside it.
- It does not misinterpret multiplication and other symbols.
- It ignores the importance of spaces.

Syntax: Working with double parenthesis

```
$((expression))
```

Example: Working with double parenthesis

```
#Bash Script to check the working of double parenthesis
num1=10
num2=4
add=$(( num1+num2 ))
sub=$((num1 - num2))
multi=$(( num1*num2))
div=$((num1/num2 ))
echo "Additon of $num1 and $num2 is $add"
echo "Subtraction of $num1 and $num2 is $sub"
echo "Multiplication of $num1 and $num2 is $multi"
echo "Division of $num1 and $num2 is $div"
```

Output of the above program

```
Additon of 10 and 4 is 14
subtraction of 10 and 4 is 6
Multiplication of 10 and 4 is 40
Division of 10 and 4 is 2
```

As you can see above that the result of division must be 2.5 but we are getting 2. It is because double parenthesis is suitable for integer arithmetic operations.

Mathematical Operators used inside Double Parenthesis are: +, -, *, /, %, +=, -=, *=, /=, %=, ++, --, **

> **Note**:- You can also use double parenthesis in if-else and loop construct.

Arithmetic Operations using square brackets

- You can only perform integer arithmetic operations using square brackets.
- In this, it is not necessary to prefix a variable with $ to access a value of the variable inside it.
- The advantage of using square brackets is that you do not need to use escape character to prevent misinterpretation of multiply and other symbols.
- It also ignores the importance of spaces.

Syntax: Working with square brackets

```
$[expression]
```

Example: Working with square brackets

```
num1=10
num2=4
add=$[ $num1+num2 ]
sub=$[num1 - $num2]
multi=$[ $num1*num2]
div=$[num1/$num2 ]
echo "Additon of $num1 and $num2 is $add"
echo "Subtraction of $num1 and $num2 is $sub"
echo "Multiplication of $num1 and $num2 is $multi"
echo "Division of $num1 and $num2 is $div"
```

Output of the above program

```
Additon of 10 and 4 is 14
subtraction of 10 and 4 is 6
Multiplication of 10 and 4 is 40
Division of 10 and 4 is 2
```

As you can see above that the result of division must be 2.5 but we are getting 2. It is because square parenthesis is suitable for integer arithmetic operations.

Bash expr command

- Bash expr command only performs integer arithmetic operations.
- The problem with expr command is that it misinterprets multiplication and other symbols. To get around with this issue, you have to use the escape character(\) to prevent any misinterpretations.
- To use expr command in shell script, you have to backtick character(`).

```
num1=10
num2=4
add=`expr $num1 + $num2`
multi=`expr $num1 \* $num2`
echo "Additon of $num1 and $num2 is $add"
echo "Multiplication of $num1 and $num2 is $multi"
```

Output of the above program

```
Addition of 10 and 4 is 14
Multiplication of 10 and 4 is 40
```

As you can see above, we have used the escape character to prevent syntax error while performing multiplication.

Problem with Double Parenthesis, Square Bracket and expr command

All of the above three methods are suitable for performing integer operations. When you try to perform floating point operations, you will either get an error or only integer result.

```
num1=10.5
num2=2
add=$((num1+num2))
echo "Addition of $num1 and $num2 is $add"
```

When you run the above program, you will get syntax error. To perform floating-point arithmetic operations, use Bash bc command.

Check out these related Bash Shell Tutorial