

Chapter 3 – Operators

When creating scripts, there is a broad spectrum of operations that you may wish to perform on data within your scripts. This may include simple operations such as adding or subtracting numbers but may involve more complex operations such as IP addressing calculations.

In this chapter, we'll look at the many operators available to change, manipulate, inspect, and compare data within our scripts.

The operators available are grouped as follows:

- Arithmetic operators
- Comparison operators
- Logical operators
- Bitwise operators
- Concatenation (joining) operators
- Miscellaneous operators

Rather than creating full scripts in the chapter, the focus will be on *how* to use these operators. There are no complete scripts but rather explanations of what each operator does and brief examples of how it might be used.

Examples are provided as commands that are typed on the MikroTik CLI (command line interface) rather than shown as part of a script. This allows you to quickly and interactively test each operator to understand how it can be used. Once you have gained an understanding of how to use it, you will be able to confidently add it to your scripts.

You are strongly encouraged to execute and test the examples for yourself. Try entering your own values for each operator and see the impact of new values.

Most examples have comment statements entered on the CLI, such as:

```
[admin@RouterOS] > # Add two integers : 100 and 200
[admin@RouterOS] > :put (100 + 200)
300
```

While adding the comment "# Add two integers : 100 and 200" may seem redundant, they are helpful mental prompts when you return to this book in future and would like to quickly look up how to use a specific operator by inspecting an example.

Arithmetic Operators

Sometimes we'd like to perform simple math on the data we're looking at in our scripts. Arithmetic operators allow us to perform simple math operations you're likely already familiar with. The operators are summarised in the table below:

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo operation
-	Negation

We'll look at how to use each operator with a few simple CLI-based examples. In these examples, I've included inline comments that describe each operation. You can try these for yourself on the CLI of a MikroTik device.



Tip:

In many of the examples shown in this chapter, you will see operations enclosed with parentheses. For example:

```
:put (100 + 200)
```

Check out the "*Grouping Operator*" section of this chapter for an explanation of why these are required.

Addition

When we need to add values together, we use the traditional "+" math operator:

```
[admin@RouterOS] > # Add two integers : 100 and 200
[admin@RouterOS] > :put (100 + 200)
300
[admin@RouterOS] > # Add an integer and an integer variable
[admin@RouterOS] > :global BigNumber 10000
[admin@RouterOS] > :put (22 + $BigNumber)
10022
[admin@RouterOS] > # Add an integer to an IP network number
[admin@RouterOS] > :put (192.168.0.0 + 257)
192.168.1.1
```

The first two examples show simple addition operations: one with two numbers and the other with an integer and a variable containing an integer value. One point to note is that numerical arithmetic operations can only be performed on integer numbers (i.e. they have no decimal points). All results are also only integer values.

The final example above shows an addition of a number to an IP address. IP addresses have their own special *data type*, allowing this operation to be performed. We will look in depth at data types in chapter 4 of this book.

Subtraction

As with the addition operator, we use the traditional "-" math operator to perform subtraction operations. The same rules around data types for the addition operator also apply to subtraction.

Here are a couple of examples:

```
[admin@RouterOS] > # Subtract two integers
[admin@RouterOS] > :put (15 - 21)
-6
[admin@RouterOS] > # Perform some IP math
[admin@RouterOS] > :put (192.168.1.100 - 10)
192.168.1.90
[admin@RouterOS] >
```

Multiplication

To perform multiplication operations, we use the "*" operator. Here are a couple of simple examples to demonstrate its use:

```
[admin@RouterOS] > # Multiply 2 numbers
[admin@RouterOS] > :put (99 * 100)
9900
[admin@RouterOS] > # Multiply 3 numbers
[admin@RouterOS] > :put (2 * 3 * 4)
24
[admin@RouterOS] >
```

Division and Modulo

The division and modulo operators are closely related to each other. The operators used are "/" and "%", respectively. Both involve division math operations but return different aspects of a division calculation.

When using the division operator (i.e. "/") on a pair of numbers, the result is the whole number produced by dividing the second number into the first. Remember that RouterOS supports only integers for numeric operations. Only an integer value can be returned when the division operation is performed. The example below demonstrates this behaviour:

```
[admin@RouterOS] > :put (5 / 2)
2
[admin@RouterOS] >
```

The actual result of this calculation is 2.5. As the result *must* be an integer, then 2 is returned.

If we use the modulo operator on the same calculation, we see that it performs the same division operation and returns the *remainder* that is left over:

```
[admin@RouterOS] > # Division operator
[admin@RouterOS] > :put (5 / 2)
2
[admin@RouterOS] > # Modulo operator (remainder from division operation)
[admin@RouterOS] > :put (5 % 2)
1
[admin@RouterOS] >
```

Negation

The negation operator allows us to negate an integer. This will enable us to turn a positive integer into a negative integer or a negative integer into a positive.

This may be useful if we wish to negate the result of a calculation or a retrieved value. It uses the "-" operator (*yes, it's the same as the subtraction operator*).

Here are a couple of examples of how it might be used:

```
[admin@RouterOS] > # negate the result of adding two positive numbers
[admin@RouterOS] > :put (-(55 + 11))
-66
[admin@RouterOS] > # negate a subtraction that produces a negative result
[admin@RouterOS] > :put (-(1 - 33))
32
```

Note how we have to use nested braces in these examples to provide an order of precedence in which operations are performed. We need the result of the inner braces before we apply the final negation to the calculation performed. There are two operations performed: addition or subtraction, followed by negation.

The braces indicate the order in which operations need to be performed, with the inner bracket's operation performed before the operation in the outermost braces. This is similar to using braces to indicate precedence used in traditional math calculations.

Comparison Operators

Comparison operators test whether a particular condition exists when comparing two values. The condition could be a test of whether two numbers are equal or perhaps whether one number is larger than another.

The result of all condition operations will always be "true" or "false". This is also known as a "boolean" result (*booleans are covered in detail in chapter 4 when we look at data types*).

Comparison operators are primarily used with other scripting constructs, such as "if" statements and "loops". A check is often made for a specific condition before executing a code section. (*"If" statements and "loops" are covered in more detail in chapters 7 and 8 of this book*).

The list of available comparison operators is shown in the table below:

Operator	Description
<	Less than
>	Greater than
=	Equal to
<=	Less than or equal to
>=	Greater than or equal to
!=	Not equal to