TutorialKart

Bash IF – Syntax and Examples

Bash IF

Bash IF statement is used for conditional branching in the sequential flow of execution of statements.

We shall learn about the syntax of if statement and get a thorough understanding of it with the help of examples.

Options for IF statement in Bash Scripting

If statement can accept options to perform a specific task. These options are used for file operations, string operations, etc. In this topic, we shall provide examples for some mostly used options.

Syntax of Bash If

Bash If statement syntax is

```
if [ expression ];
# ^ ^          ^      please note these spaces
then
    statement(s)
fi
```

Note : Observe the mandatory spaces required, in the first line, marked using arrows. Also the semicolon at the end of first line. And **if conditional statement** ends with fi

The syntax to include multiple conditions with AND operator is

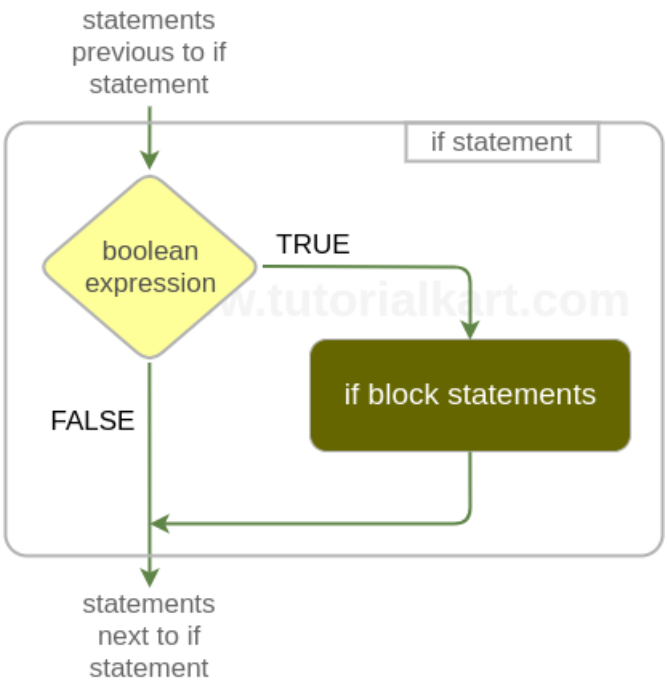
```
if [ expression ] && [ expression_2 ];
then
    statement(s)
fi
```

The syntax to include multiple conditions with OR operator is

```
if [ expression ] || [ expression_2 ];
then
statement(s)
fi
```

For compound expressions, following **if** syntax is allowed. Please observe that the condition has double square brackets.

```
if [[ expression_1 && expression_2 || expression_3 ]];
then
    statement(s)
fi
```



Example 1 – Bash IF

In the following example, we demonstrate the usage of **if statement** with a simple scenario of comparing two strings.

Bash Script File

```
#!/bin/bash
```

```
#!/bin/bash

# if condition is true
if [ "hello" == "hello" ];
then
    echo "hello equals hello"
fi

# if condition is false
if [ "hello" == "bye" ];
then
    echo "hello equals bye"
fi
```

Note : In bash, respect each token/literal. Observe the spaces provided after `if` `[` string literal “hello” and `==`

When you run the above bash if example script file in a shell program like Terminal, the result would be

Output

```
~$ ./bash-if-example
hello equals hello
```

Example 2 – Bash IF – Compare Numbers

In the following example, we will compare numbers using if statement.

Bash Shell Script

```
#!/bin/bash

# if condition (greater than) is true
if [ 8 -gt 7 ];
then
    echo "is 8 greater than 7 : true "
fi

# if condition (greater than) is false
if [ 7 -gt 8 ];
then
    echo "is 7 greater than 8 : false "
fi

# if condition (less than) is true
if [ 7 -lt 8 ];
then
    echo "is 7 lesser than 8 : true "
fi

# if condition (lesser than) is false
if [ 8 -lt 7 ];
then
    echo "is 8 lesser than 7 : false "
fi
```

```
# if condition (equal to) is true
if [ 8 -eq 8 ];
then
    echo "is 8 equals 8 : true "
fi

# if condition (equal to) is false
if [ 7 -eq 8 ];
then
    echo "is 7 equals 8 : false "
fi
```

When you run the above bash script file in shell program like Terminal, the result would be

Output

```
~$ ./bash-if-example-2
is 8 greater than 7 : true
is 7 lesser than 8 : true
is 8 equals 8 : true
```

Example 3 – Using AND in IF Expression

In this example, we shall learn to use AND operator `&&` to combine multiple conditions and form an expression (compound condition).

Bash Script File

```
#!/bin/bash

# TRUE && TRUE
if [ "hello" == "hello" ] && [ 1 -eq 1 ];
then
    echo "if 1"
fi

# TRUE && FALSE
if [ "hello" == "hello" ] && [ 1 -gt 2 ];
then
    echo "if 2"
fi
```

Output

```
~$ ./bash-if-example-3
if 1
```

Example 4 – Using OR in IF Expression

In this example, we shall learn to use OR operator `||` to combine multiple conditions and form an expression (compound condition).

Bash Script File

```
#!/bin/bash

# TRUE || FALSE
if [ "hello" == "hello" ] || [ 1 -eq 3 ];
then
    echo "if 1"
fi

# FALSE || FALSE
if [ "hello" == "hi" ] || [ 1 -gt 2 ];
then
    echo "if 2"
fi
```

Output

```
~$ ./bash-if-example-4
if 1
```

Example 5 – Bash IF with Multiple Conditions

In this example, we shall learn to include multiple conditions combined with AND and OR forming a single expression.

Bash Script File

```
#!/bin/bash

# FALSE && TRUE || FALSE || TRUE evaluates to TRUE
if [[ 8 -eq 11 && "hello" == "hello" || 1 -eq 3 || 1 -eq 1 ]];
then
    echo "if 1"
fi

# FALSE && TRUE || FALSE evaluates to FALSE
if [[ 8 -eq 11 && "hello" == "hello" || 1 -eq 3 ]];
then
    echo "if 2"
fi
```

Output

```
~$ ./bash-if-example-5
if 1
```

Example 6 – Bash IF -z

If statement when used with optionz , returns true if the length ofthe string is zero. Following example proves the same.

Bash Script File

```
#!/bin/bash

if [ -z "" ];
then
    echo "zero length string"
fi

if [ -z "hello" ];
then
    echo "hello is zero length string"
else
    echo "hello is not zero length string"
fi
```

Example 7 – Bash IF -s

Bash If statement when used with options , returns true if size of the file is greater than zero.

Bash Script File

```
if [ -s /home/tutorialkart/sample.txt ];
then
    echo "Size of sample.txt is greater than zero"
else
    echo "Size of sample.txt is zero"
fi
```

Example 8 – Bash IF -n

Bash If statement when used with optionn , returns true if the length ofthe string is greater than zero.

Bash Script File

```
#!/bin/bash

if [ -n "learn" ];
then
    echo "learn is non-zero length string"
fi

if [ -n "hello" ];
then
```

```
    echo "hello is non-zero length string"
else
    echo "hello is zero length string"
fi
```

Example 9 – Bash IF -f

Bash If statement when used with optionf , returns true if the length ofthe string is zero. Following example proves the same.

Bash Script File

```
#!/bin/bash

if [ -f /home/tutorialkart/sample.txt ];
then
    echo "sample.txt - File exists."
else
    echo "sample.txt - File does not exist."
fi
```

Conclusion

In this Bash Tutorial, we learned conditional branching in the sequential flow of execution of statements with **bash if** statement. We learned the syntax and usage of Bash IF with example shell scripts.

Bash Shell Scripting

- Bash Tutorial
- Bash Script Example
- Bash File Extension
- Bash Echo
- Bash Comments
- Bash Variable
- Bash Command Line Arguments
- Bash Read User Input
- Bash Read Password
- Bash Date Format
- Bash Sleep

Operators

- Bash Arithmetic Operators

Conditional Statements

- **Bash If**
- Bash If Else
- Bash Else If
- Bash Case

Loops

- Bash For Loop
- Bash While Loop
- Bash Until Loop

Strings

- Bash String Manipulation Examples
- Bash String Length
- Bash If String Equals
- Bash Split String
- Bash SubString
- Bash Concatenate String
- Bash Concatenate Variables to Strings

Functions

- Bash Function
- Bash Override Buitlin Commands

Arrays

- Bash Array

Files

- Bash Write to File
- Bash Read File
- Bash Read File line by line

- Bash If File Exists
- Bash If File is Directory
- Bash If File is Readable

Bash Others

- Bash Check if variable is set