

Bash While Loop

In this topic, we have demonstrated how to use while loop statement in Bash Script.

The **bash while loop** can be defined as a control flow statement which allows executing the given set of commands repeatedly as long as the applied condition evaluates to true. For example, we can either run echo command many times or just read a text file line by line and process the result by using while loop in Bash.

Syntax of Bash While Loop

Bash while loop has the following format:

1. while [expression];
2. do
3. commands;
4. multiple commands;
5. done

The above syntax is applicable only if the expression contains a single condition.

Backward Skip 10sPlay VideoForward Skip 10s

If there are multiple conditions to include in the expression, then the syntax of the while loop will be as follows:

1. while [expressions];
2. do
3. commands;
4. multiple commands;
5. done

The while loop one-liner syntax can be defined as:

1. while [condition]; do commands; done
2. while control-command; do Commands; done

There are some key points of 'while loop' statement:

- The condition is checked before executing the commands.
- The 'while' loop is also capable of performing all the work as for 'loop' can do.
- The commands between 'do' and 'done' are repeatedly executed as long as the condition evaluates to true.
- The arguments for a 'while' loop can be a boolean expression.

How it works

The while loop is a restricted entry loop. It means that the condition is checked before executing the commands of the while loop. If the condition evaluates to true, the set of commands following that condition are executed. Otherwise, the loop is terminated, and the program control is given to the other command following the 'done' statement.

Bash While Loop Examples

Following are some examples of bash while loop:

While Loop with Single Condition

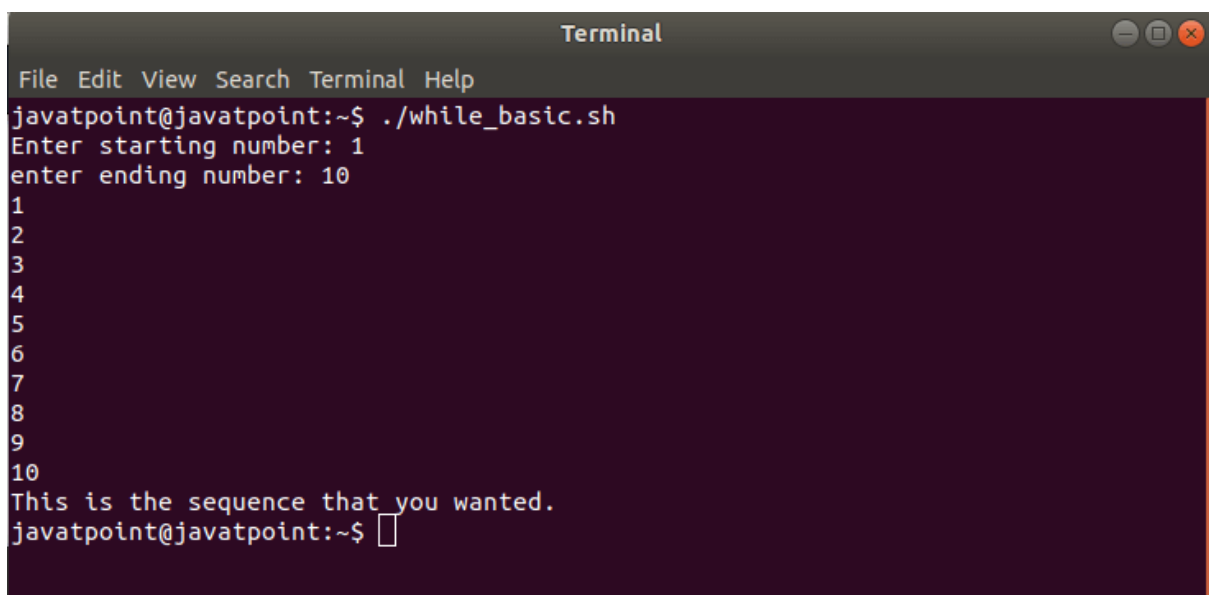
In this example, the while loop is used with a single condition in expression. It is the basic example of while loop which will print series of numbers as per user input:

Example

1. `#!/bin/bash`
2. `#Script to get specified numbers`

- 3.
4. `read -p "Enter starting number: " snum`
5. `read -p "Enter ending number: " enum`
- 6.
7. `while [[$snum -le $enum]];`
8. `do`
9. `echo $snum`
10. `((snum++))`
11. `done`
- 12.
13. `echo "This is the sequence that you wanted."`

Output

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is "javatpoint@javatpoint:~\$". The user runs the command ". /while_basic.sh". The script prompts "Enter starting number: 1" and "enter ending number: 10". It then prints the numbers 1 through 10, followed by the message "This is the sequence that you wanted." and returns to the shell prompt.

```
Terminal
File Edit View Search Terminal Help
javatpoint@javatpoint:~$ ./while_basic.sh
Enter starting number: 1
enter ending number: 10
1
2
3
4
5
6
7
8
9
10
This is the sequence that you wanted.
javatpoint@javatpoint:~$
```

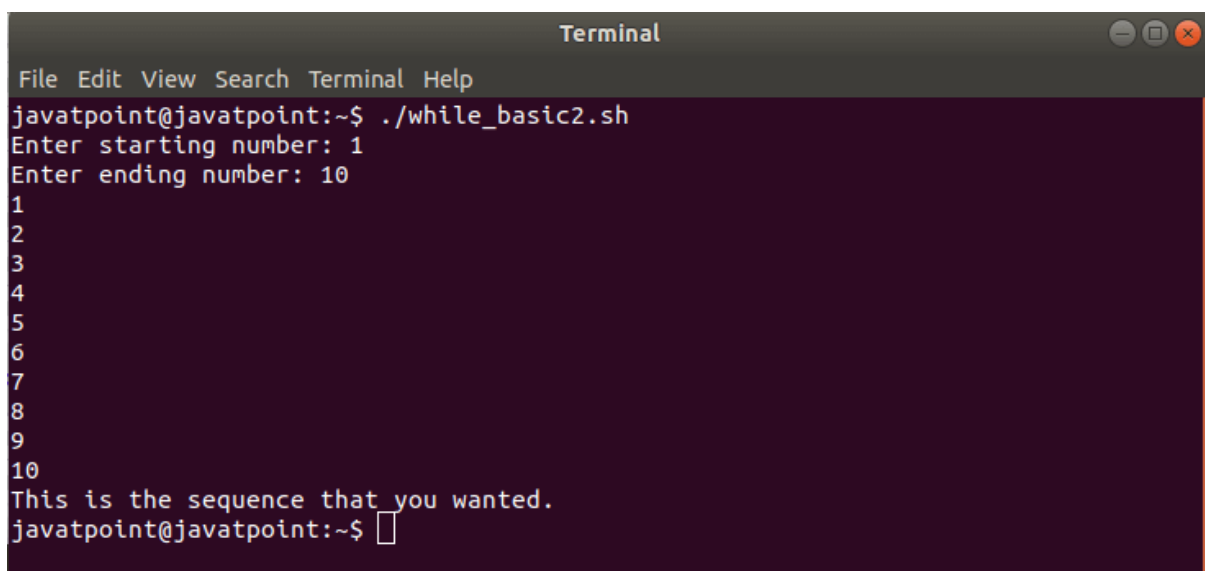
While Loop with Multiple Conditions

Following is an example of while loop with multiple conditions in the expression:

Example

1. `#!/bin/bash`
2. `#Script to get specified numbers`
- 3.
4. `read -p "Enter starting number: " snum`
5. `read -p "Enter ending number: " enum`
- 6.
7. `while [[$snum -lt $enum || $snum == $enum]];`
8. `do`
9. `echo $snum`
10. `((snum++))`
11. `done`
- 12.
13. `echo "This is the sequence that you wanted."`

Output

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is "javatpoint@javatpoint:~\$". The user runs the command "./while_basic2.sh". The script prompts "Enter starting number: 1" and "Enter ending number: 10". It then outputs the numbers 1 through 10, each on a new line, followed by the message "This is the sequence that you wanted." and returns to the shell prompt.

```
Terminal
File Edit View Search Terminal Help
javatpoint@javatpoint:~$ ./while_basic2.sh
Enter starting number: 1
Enter ending number: 10
1
2
3
4
5
6
7
8
9
10
This is the sequence that you wanted.
javatpoint@javatpoint:~$
```

Infinite While Loop

An infinite loop is a loop that has no ending or termination. If the condition always evaluates to true, it creates an infinite loop. The loop will execute continuously until it is forcefully stopped using CTRL+C :

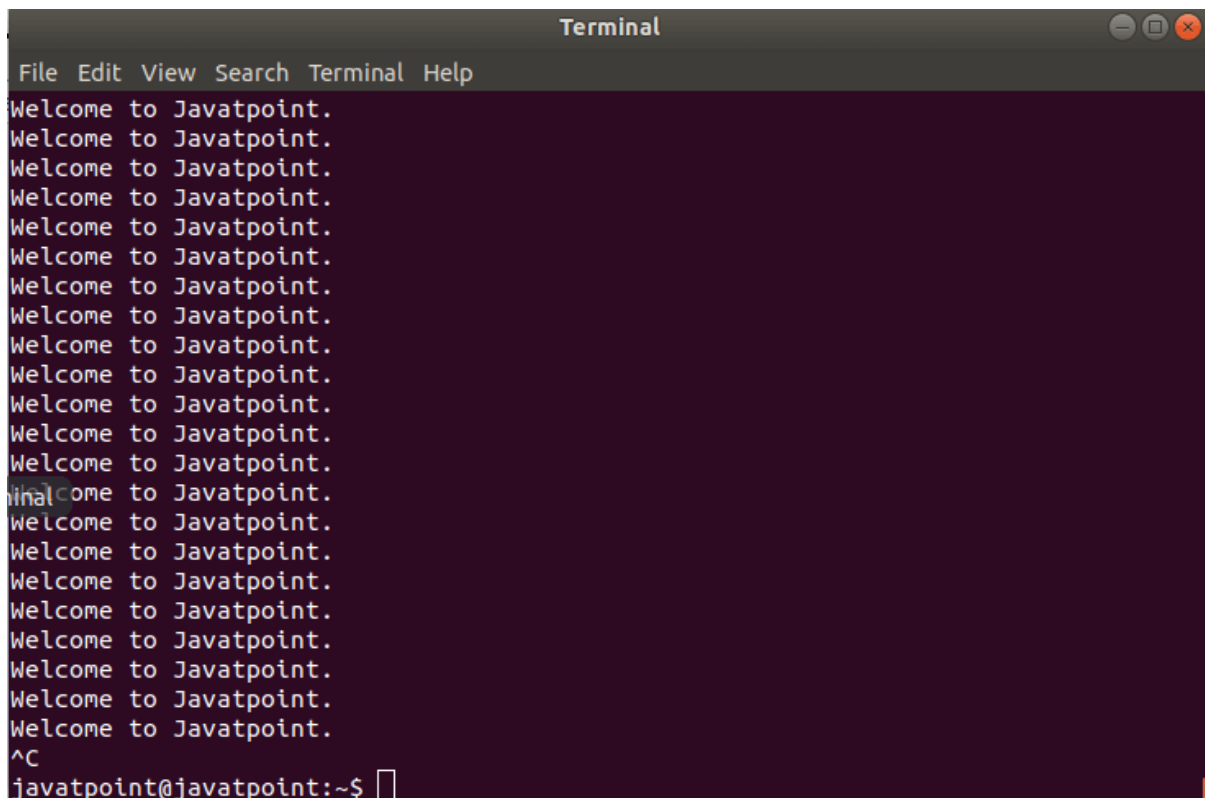
Example

1. `#!/bin/bash`
2. `#An infinite while loop`
- 3.
4. `while :`
5. `do`
6. `echo "Welcome to Javatpoint."`
7. `done`

We can also write the above script in a single line as:

1. `#!/bin/bash`
2. `#An infinite while loop`
- 3.
4. `while ;; do echo "Welcome to Javatpoint."; done`

Output

A screenshot of a macOS Terminal window titled "Terminal". The window has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal content shows a continuous loop of the text "Welcome to Javatpoint." printed on each line. At the bottom, the prompt "javatpoint@javatpoint:~\$" is visible with a cursor. Above the prompt, there are some control characters like "^C" and a partial line "inal come to Javatpoint.".

Here, we have used the built-in command `(:)` which always return true. We can also use the built-in command `true` to create an infinite loop just as below:

Example

1. `#!/bin/bash`
2. `#An infinite while loop`
- 3.
4. `while true`
5. `do`
6. `echo "Welcome to Javatpoint"`
7. `done`

This bash script will also provide the same output as an above infinite script.

Note: Infinite loops can be terminated by using CTRL+C or by adding some conditional exit within the script.

While Loop with a Break Statement

A break statement can be used to stop the loop as per the applied condition. For example:

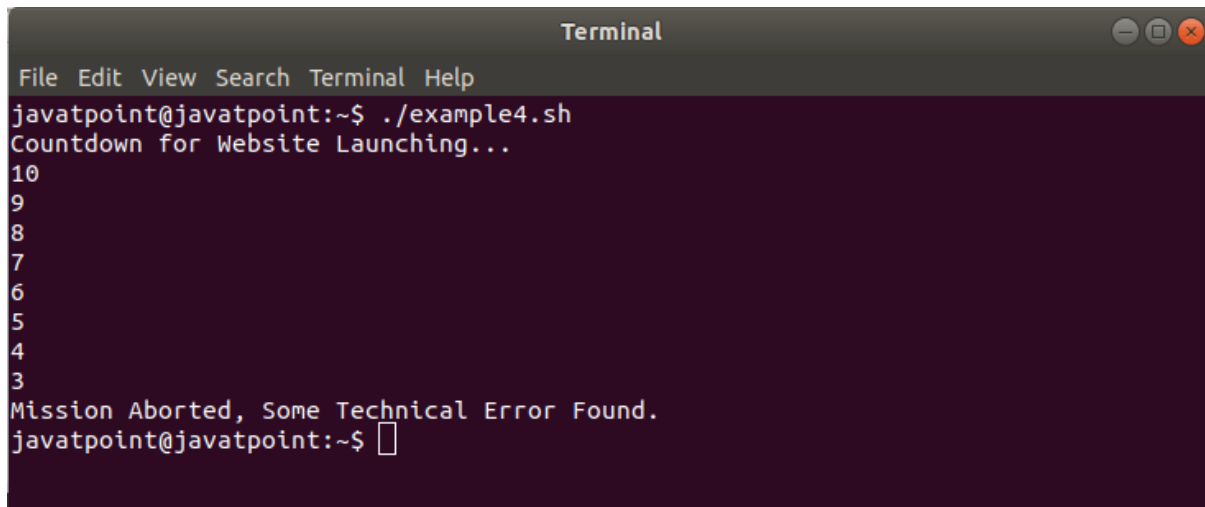
Example

```
1. #!/bin/bash
2. #While Loop Example with a Break Statement
3.
4. echo "Countdown for Website Launching..."
5. i=10
6. while [ $i -ge 1 ]
7. do
8.   if [ $i == 2 ]
9.   then
10.      echo "Mission Aborted, Some Technical Error Found."
11.      break
12.   fi
13.   echo "$i"
14.   (( i-- ))
15. done
```

Output

According to the script, the loop is assigned to iterate for ten times. But there is a condition after eight times of iteration which will break

the iteration and terminate the loop. The following output will be shown after executing the script.



```
Terminal
File Edit View Search Terminal Help
javatpoint@javatpoint:~$ ./example4.sh
Countdown for Website Launching...
10
9
8
7
6
5
4
3
Mission Aborted, Some Technical Error Found.
javatpoint@javatpoint:~$
```

While Loop with a Continue Statement

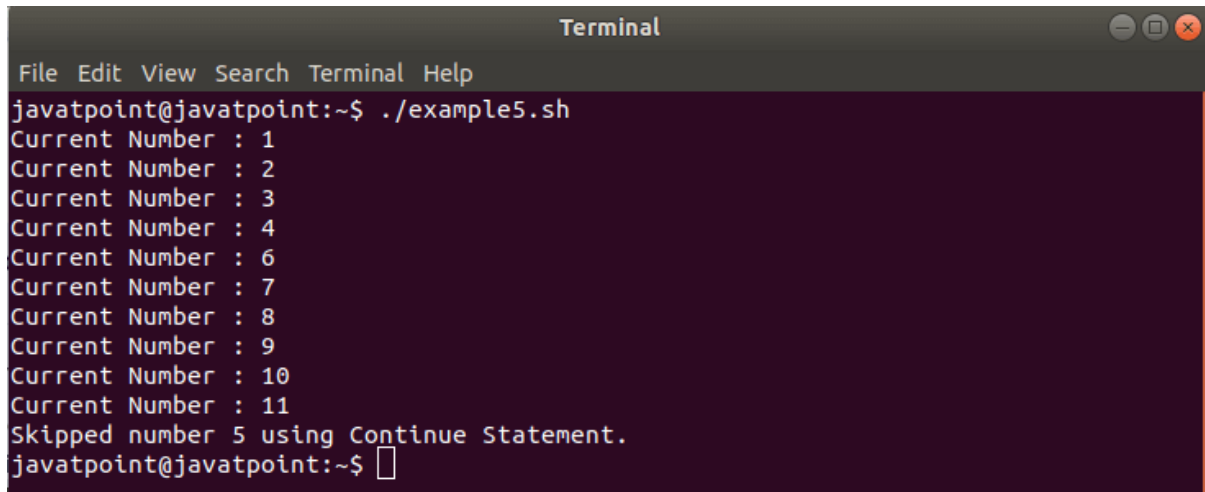
A continue statement can be used to skip the iteration for a specific condition inside the while loop.

Example

1. `#!/bin/bash`
2. `#While Loop Example with a Continue Statement`
- 3.
4. `i=0`
5. `while [$i -le 10]`
6. `do`
7. `((i++))`
8. `if [["$i" == 5]];`
9. `then`
10. `continue`
11. `fi`
12. `echo "Current Number : $i"`

13. done
- 14.
15. echo "Skipped number 5 using Continue Statement."

Output



```
Terminal
File Edit View Search Terminal Help
javatpoint@javatpoint:~$ ./example5.sh
Current Number : 1
Current Number : 2
Current Number : 3
Current Number : 4
Current Number : 6
Current Number : 7
Current Number : 8
Current Number : 9
Current Number : 10
Current Number : 11
Skipped number 5 using Continue Statement.
javatpoint@javatpoint:~$
```

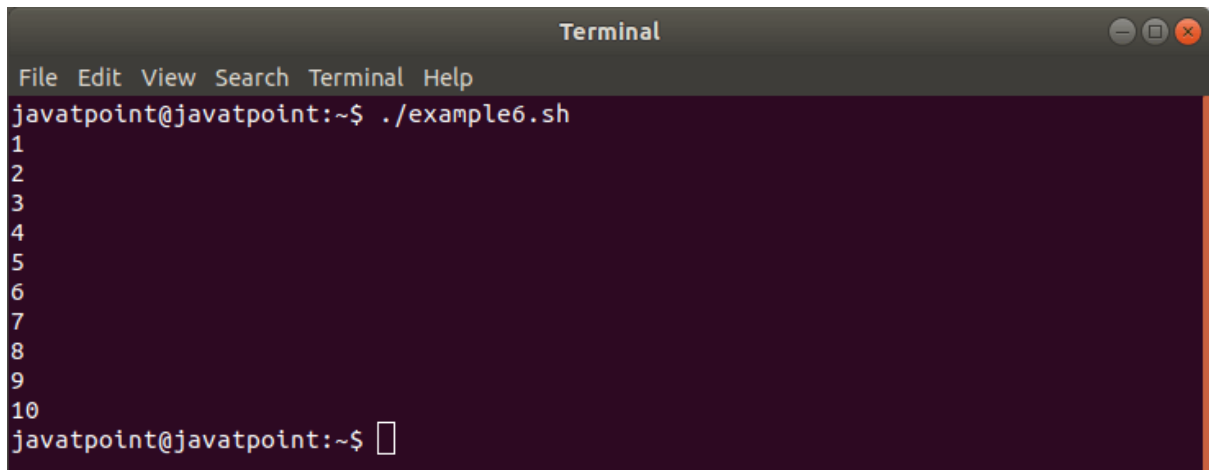
While Loop with C-Style

We can also write while loop in bash script as similar as a while loop in C programming language.

Example

1. #!/bin/bash
2. #While loop example in C style
- 3.
4. i=1
5. while((i <= 10))
6. do
7. echo \$i
8. let i++
9. done

Output



```
Terminal
File Edit View Search Terminal Help
javatpoint@javatpoint:~$ ./example6.sh
1
2
3
4
5
6
7
8
9
10
javatpoint@javatpoint:~$
```