



# How to Use Bash For Loop with Examples in Linux

James Kiarie | Last Updated: June 6, 2023 | Read Time: 4 mins | [Bash Shell, Shell Scripting](#) | [1 Comment](#)

In programming languages, Loops are essential components and are used when you want to repeat code over and over again until a specified condition is met.

In Bash scripting, loops play much the same role and are used to [automate repetitive tasks](#) just like in programming languages.

In Bash scripting, there are 3 types of loops: for loop, while loop, and until loop. The three are used to iterate over a list of values and perform a given set of commands.

In this guide, we will focus on the Bash For Loop in Linux.

## Table of Contents



Bash For Loop Syntax

Bash For Loop Example

Bash For Loop with Ranges

Bash For Loops with Arrays

Bash C Style For Loop

Bash C-styled For Loops With Conditional Statements

Use the 'Continue' statement with Bash For Loop

Use the 'break' statement with Bash For Loop

Conclusion

## Bash For Loop Syntax

As mentioned earlier, the for loop iterates over a range of values and executes a [set of Linux commands](#).

For loop takes the following syntax:

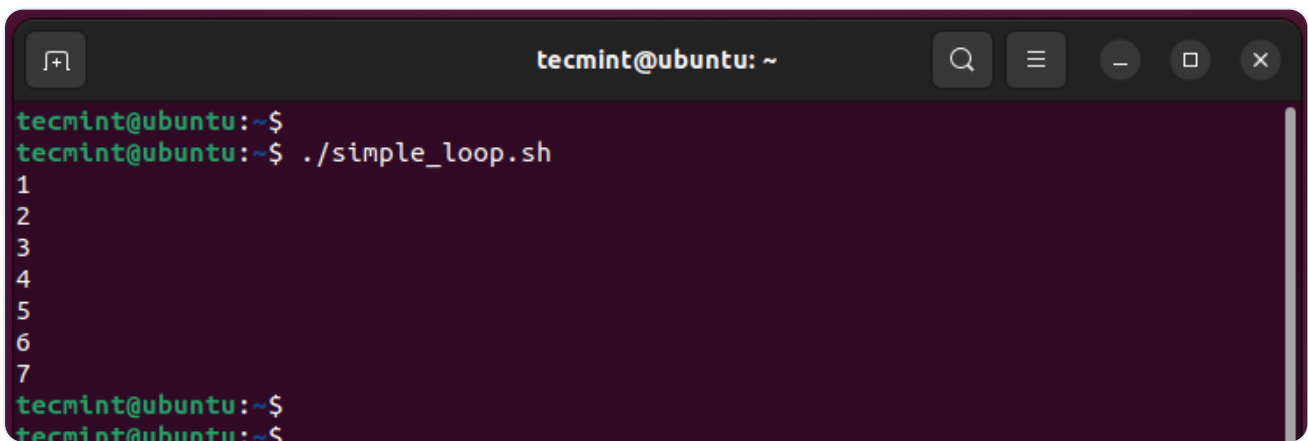
```
for variable_name in value1 value2 value3 .. n
do
    command1
    command2
    commandn
done
```

Let us now check a few example usages of the bash for loop.

## Bash For Loop Example

In its simplest form, the for loop takes the following basic format. In this example, the variable `n` iterates over a group of numerical values enclosed in curly braces and prints out their values to stdout.

```
for n in {1 2 3 4 5 6 7};
do
    echo $n
done
```

A terminal window titled 'tecmint@ubuntu: ~' with search, menu, and window control icons. It shows the execution of a script named 'simple\_loop.sh'. The script uses a for loop to iterate over the values 1 through 7, printing each value on a new line. The prompt 'tecmint@ubuntu:~\$' is shown before and after the script execution.

```
tecmint@ubuntu:~$
tecmint@ubuntu:~$ ./simple_loop.sh
1
2
3
4
5
6
7
tecmint@ubuntu:~$
tecmint@ubuntu:~$
```

Bash For Loop Example

## Bash For Loop with Ranges

In the previous examples, we explicitly listed the values to be iterated by the `for` loop, which works just fine. However, you can only imagine how cumbersome and time-consuming a task it would be if you were to iterate over, for example, a hundred values. This would compel you to type all the values from 1 to 100.

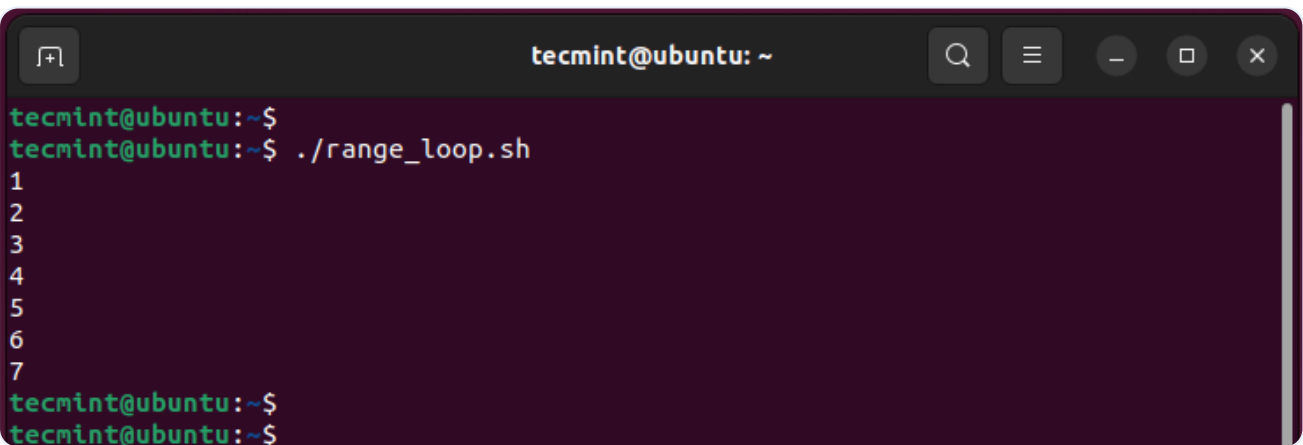
To address this issue, specify a range. To do so, specify the number to start and stop separated by two periods.

In this example, 1 is the first value whilst 7 is the last value in the range.

```
#!/bin/bash

for n in {1..7};
do
    echo $n
done
```

Once the shell script is executed, all the values in the range are listed, similar to what we had in simple loops.

A terminal window titled 'tecmint@ubuntu: ~' with search, menu, and window control icons. The prompt is 'tecmint@ubuntu:~\$'. The user enters './range\_loop.sh'. The output shows the numbers 1 through 7, each on a new line. The prompt returns to 'tecmint@ubuntu:~\$' twice.

```
tecmint@ubuntu:~$
tecmint@ubuntu:~$ ./range_loop.sh
1
2
3
4
5
6
7
tecmint@ubuntu:~$
tecmint@ubuntu:~$
```

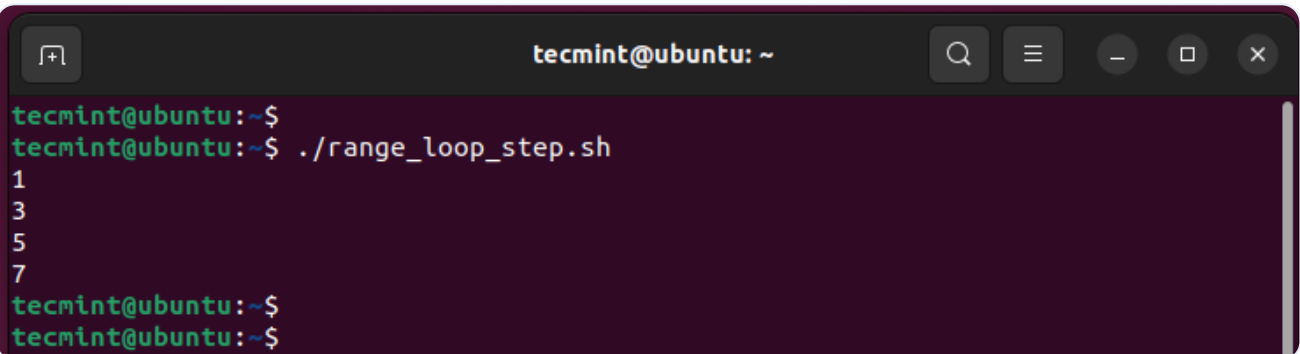
Bash For Loop with Ranges Example

Additionally, we can include a value at the end of the range that is going to cause the `for` loop to iterate through the values in incremental steps.

The following bash script prints the values between 1 and 7 with 2 incremental steps between the values starting from the first value.

```
#!/bin/bash

for n in {1..7..2};
do
    echo $n
done
```

A terminal window titled 'tecmint@ubuntu: ~' with search, menu, and window control icons. The prompt is 'tecmint@ubuntu:~\$'. The user enters './range\_loop\_step.sh'. The script outputs '1', '3', '5', and '7' on separate lines. The prompt returns to 'tecmint@ubuntu:~\$' twice.

```
tecmint@ubuntu:~$
tecmint@ubuntu:~$ ./range_loop_step.sh
1
3
5
7
tecmint@ubuntu:~$
tecmint@ubuntu:~$
```

Bash For Loop Incremented Values

From the above example, you can see that the loop incremented the values inside the curly braces by 2 values.

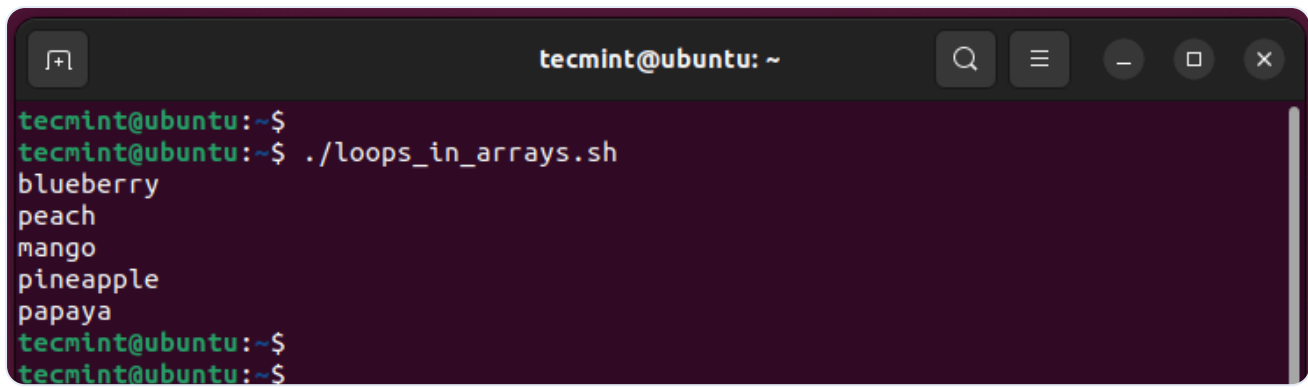
## Bash For Loops with Arrays

You can also easily iterate through values defined in an array using a for Loop. In the following example, the for loop iterates through all the values inside the fruits array and prints them to stdout.

```
#!/bin/bash

fruits=("blueberry" "peach" "mango" "pineapple" "papaya")

for n in ${fruits[@]};
do
    echo $n
done
```



```
tecmin@ubuntu: ~  
tecmin@ubuntu:~$  
tecmin@ubuntu:~$ ./loops_in_arrays.sh  
blueberry  
peach  
mango  
pineapple  
papaya  
tecmin@ubuntu:~$  
tecmin@ubuntu:~$
```

Bash For Loop Array Example

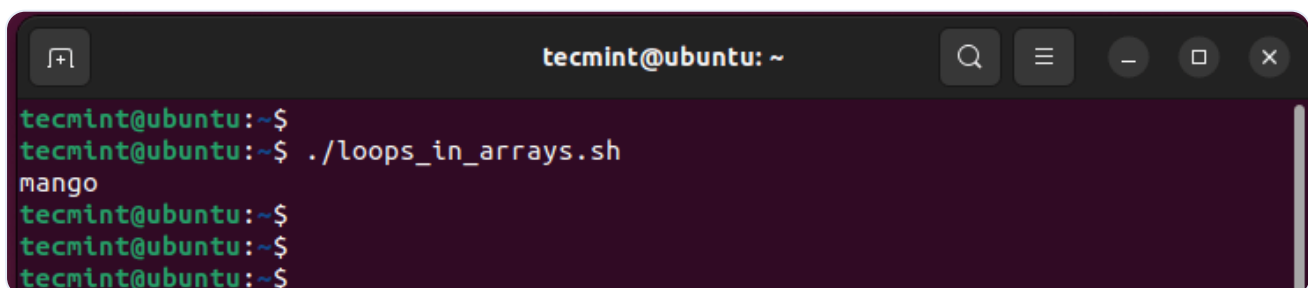
The `@` operator accesses or targets all the elements. This makes it possible to iterate over all the elements one by one.

In addition, you can access a single element by specifying its position within the array.

For example to access the “mango” element, replace the `@` operator with the position of the element in the array (the first element starts at 0, so in this case, “mango” will be denoted by 2).

This is what the for loop looks like.

```
#!/bin/bash  
  
fruits=("blueberry" "peach" "mango" "pineapple" "papaya")  
  
for n in ${fruits[2]};  
do  
    echo $n  
done
```



```
tecmin@ubuntu: ~  
tecmin@ubuntu:~$  
tecmin@ubuntu:~$ ./loops_in_arrays.sh  
mango  
tecmin@ubuntu:~$  
tecmin@ubuntu:~$  
tecmin@ubuntu:~$
```

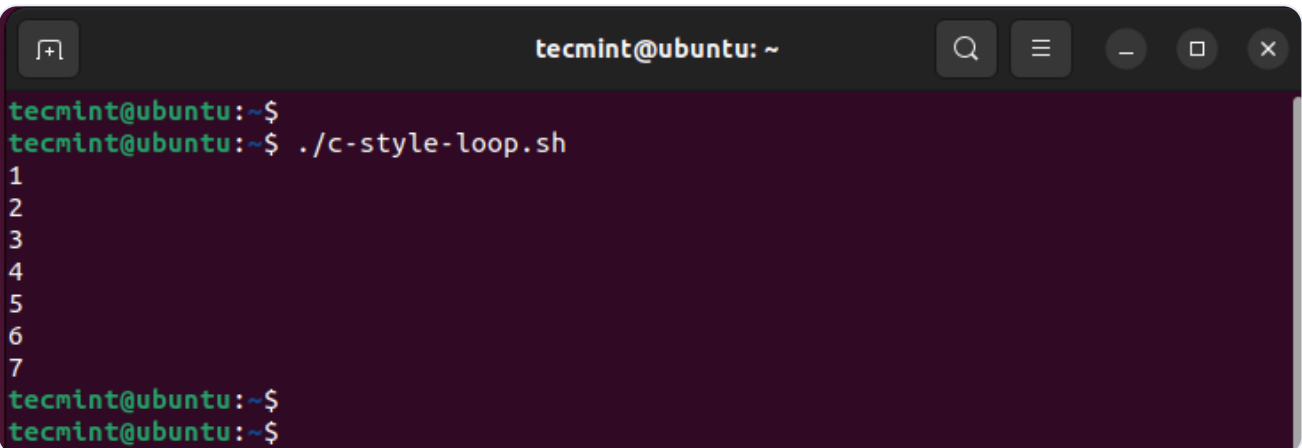
Bash For Loops with Array Elements

## Bash C Style For Loop

You can use variables inside loops to iterate over a range of elements. This is where C-styled for loops come in. The following example illustrates a C-style for loop that prints out a list of numerical values from 1 to 7.

```
#!/bin/bash

n=7
for (( n=1 ; n<=$n ; n++ ));
do
    echo $n
done
```

A terminal window titled 'tecmint@ubuntu: ~' with search, menu, and window control icons. It shows the execution of a script named 'c-style-loop.sh'. The script prints numbers 1 through 7, one per line. The prompt returns to the user after the script finishes.

```
tecmint@ubuntu:~$
tecmint@ubuntu:~$ ./c-style-loop.sh
1
2
3
4
5
6
7
tecmint@ubuntu:~$
tecmint@ubuntu:~$
```

Bash C-styled For Loops Example

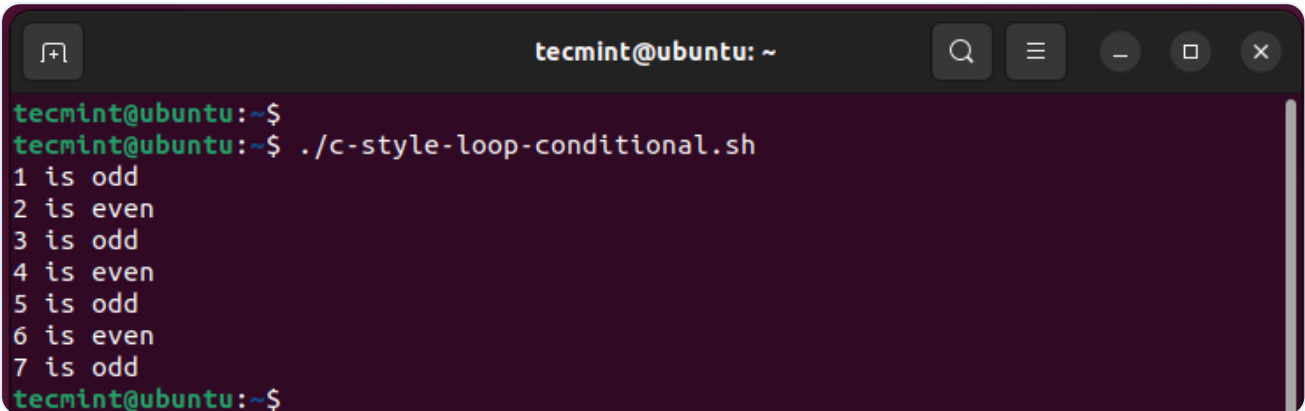
## Bash C-styled For Loops With Conditional Statements

You can include conditional statements inside C-styled for loops. In the following example, we have included an if-else statement that checks and prints out even and odd numbers between 1 and 7.

```
#!/bin/bash

for (( n=1; n<=7; n++ ))
do
    # Check if the number is even or not
    if (( $n%2==0 ))
```

```
then
    echo "$n is even"
else
    echo "$n is odd"
fi
done
```

A terminal window titled 'tecmin@ubuntu: ~' showing the execution of a script. The prompt is 'tecmin@ubuntu:~\$' and the command is './c-style-loop-conditional.sh'. The output shows a list of numbers from 1 to 7, each followed by 'is odd' or 'is even'. The prompt returns to 'tecmin@ubuntu:~\$' after the script finishes.

```
tecmin@ubuntu:~$
tecmin@ubuntu:~$ ./c-style-loop-conditional.sh
1 is odd
2 is even
3 is odd
4 is even
5 is odd
6 is even
7 is odd
tecmin@ubuntu:~$
```

Bash C-styled For Loops Conditional Statements Example

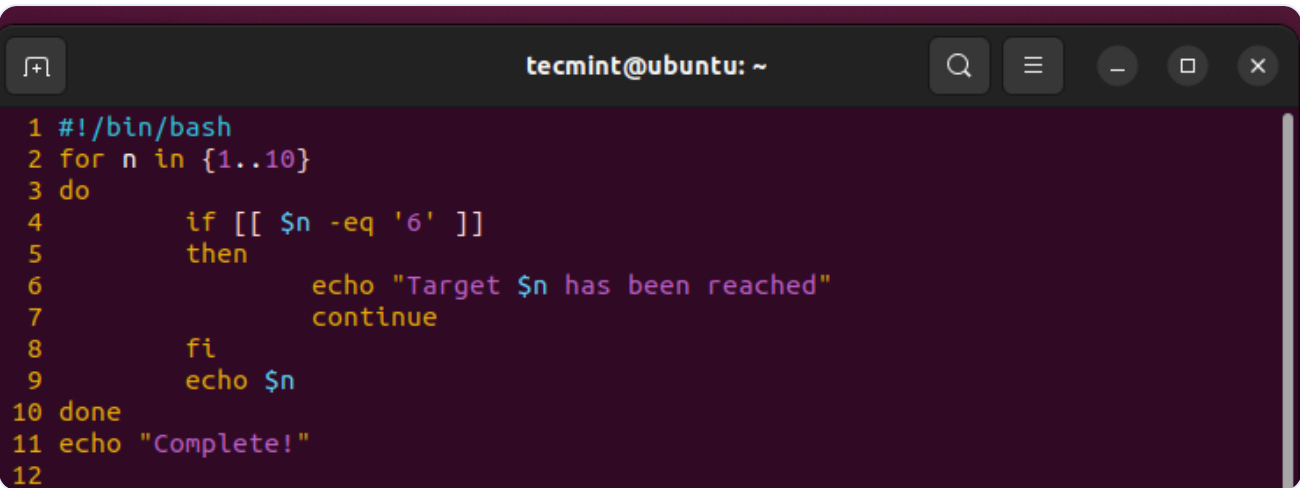
## Use the 'Continue' statement with Bash For Loop

The 'continue' statement is a built-in command that controls how a script runs. Apart from bash scripting, it is also used in programming languages such as [Python](#) and Java.

The continue statement halts the current iteration inside a loop when a specific condition is met, and then resumes the iteration.

Consider the for loop shown below.

```
#!/bin/bash
for n in {1..10}
do
    if [[ $n -eq '6' ]]
    then
        echo "Target $n has been reached"
        continue
    fi
    echo $n
done
```



```
1 #!/bin/bash
2 for n in {1..10}
3 do
4     if [[ $n -eq '6' ]]
5     then
6         echo "Target $n has been reached"
7         continue
8     fi
9     echo $n
10 done
11 echo "Complete!"
12
```

Bash For Loop Continue Statement Example

This is what the code does:

- Line 2: Marks the beginning of the for loop and iterate the variable n from 1 to 10.
- Line 4: Checks the value of n and if the variable is equal to 6, the script echoes a message to stdout and restarts the loop at the next iteration in line 2.
- Line 9: Prints the values to the screen only if the condition in line 4 is false.

The following is the expected output after running the script.



```
tecmin@ubuntu:~$
tecmin@ubuntu:~$ ./skip_and_continue_for_loop.sh
1
2
3
4
5
Target 6 has been reached
7
8
9
10
Complete!
tecmin@ubuntu:~$
tecmin@ubuntu:~$
```

Bash For Loop Continue Statement Output

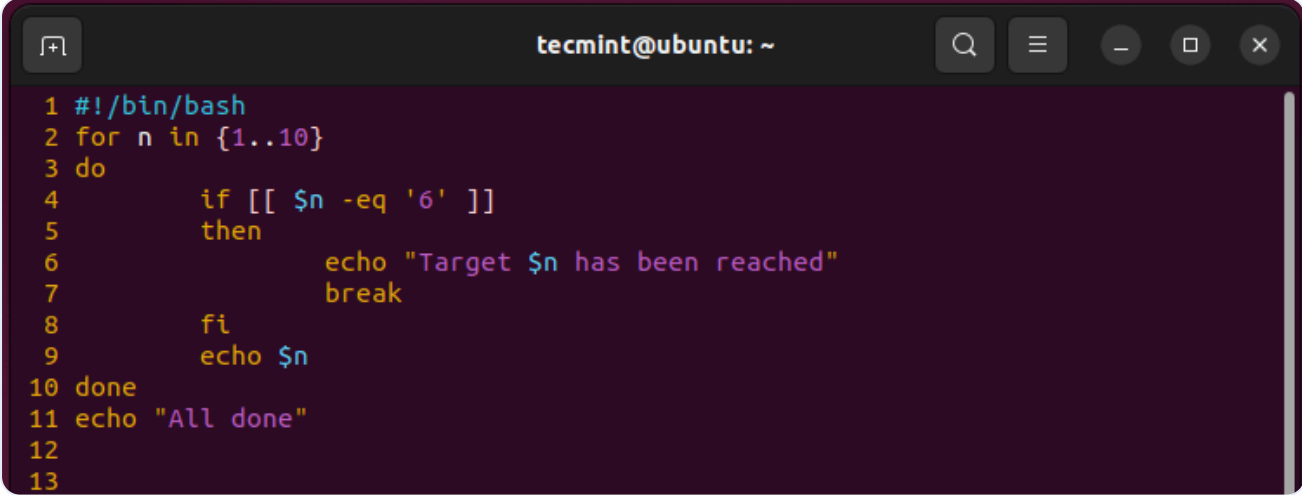
## Use the 'break' statement with Bash For Loop



The 'break' statement, as the name suggests, halts or ends the iteration when a condition is met.

Consider the For loop below.

```
#!/bin/bash
for n in {1..10}
do
    if [[ $n -eq '6' ]]
    then
        echo "Target $n has been reached"
        break
    fi
    echo $n
done
echo "All done"
```



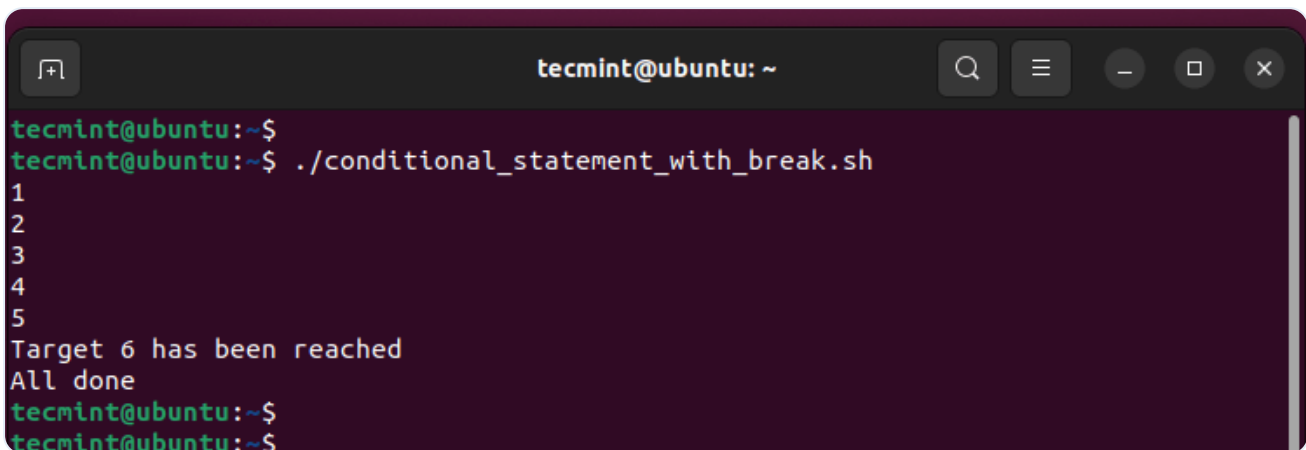
```
1 #!/bin/bash
2 for n in {1..10}
3 do
4     if [[ $n -eq '6' ]]
5     then
6         echo "Target $n has been reached"
7         break
8     fi
9     echo $n
10 done
11 echo "All done"
12
13
```

Bash For Loop Break Statement

This is what the code does:

- Line 2: Marks the beginning of the for loop and iterate the variable n from 1 to 10.
- Line 4: Checks the value of n and if the variable is equal to 6, the script echoes a message to stdout and halts the iteration.
- Line 9: Prints the numbers to the screen only if the condition in line 4 is false.

From the output, you can see that the loop stops once the variable meets the condition of the loop.

A terminal window titled 'tecmin@ubuntu: ~' showing the execution of a script. The prompt is 'tecmin@ubuntu:~\$'. The user enters './conditional\_statement\_with\_break.sh'. The script outputs: '1', '2', '3', '4', '5', 'Target 6 has been reached', and 'All done'. The prompt returns to 'tecmin@ubuntu:~\$'.

Bash For Loop Break Statement Output

## Conclusion

That was a tutorial about Bash For loops. We hope you found this insightful. Feel free to weigh in with your feedback.

Hey TecMint readers,

Exciting news! Every month, our top blog commenters will have the chance to win fantastic rewards, like free Linux eBooks such as RHCE, RHCSA, LFCS, Learn Linux, and Awk, each worth \$20!

Learn [more about the contest](#) and stand a chance to win by [sharing your thoughts below!](#)

# GIVEAWAY!

## Win eBooks



[www.tecmint.com](http://www.tecmint.com)

PREVIOUS ARTICLE:

**[How to Fix “bash syntax error near unexpected token” in Linux](#)**

NEXT ARTICLE:

**[13 Practical Examples of Using the Gzip Command in Linux](#)**



## James Kiarie

This is James, a certified Linux administrator and a tech enthusiast who loves keeping in touch with emerging trends in the tech world. When I'm not running commands on the terminal, I'm taking listening to some cool music. taking a casual stroll or watching a nice movie.

*Each tutorial at TecMint is created by a team of experienced Linux system administrators so that it meets our high-quality standards.*

Join the [TecMint Weekly Newsletter](#) (More Than 156,129 Linux Enthusiasts Have Subscribed)

Was this article helpful? Please [add a comment](#) or [buy me a coffee](#) to show your appreciation.

## Related Posts

```
ravi@TecMint:~/Downloads$ ./table.sh
Enter the number for which you want to print the table:
10
1 x 10 = 10
2 x 10 = 20
3 x 10 = 30
4 x 10 = 40
5 x 10 = 50
6 x 10 = 60
7 x 10 = 70
8 x 10 = 80
9 x 10 = 90
10 x 10 = 100
ravi@TecMint:~/Downloads$
```

## Learn Basic Mathematical Operations in Bash

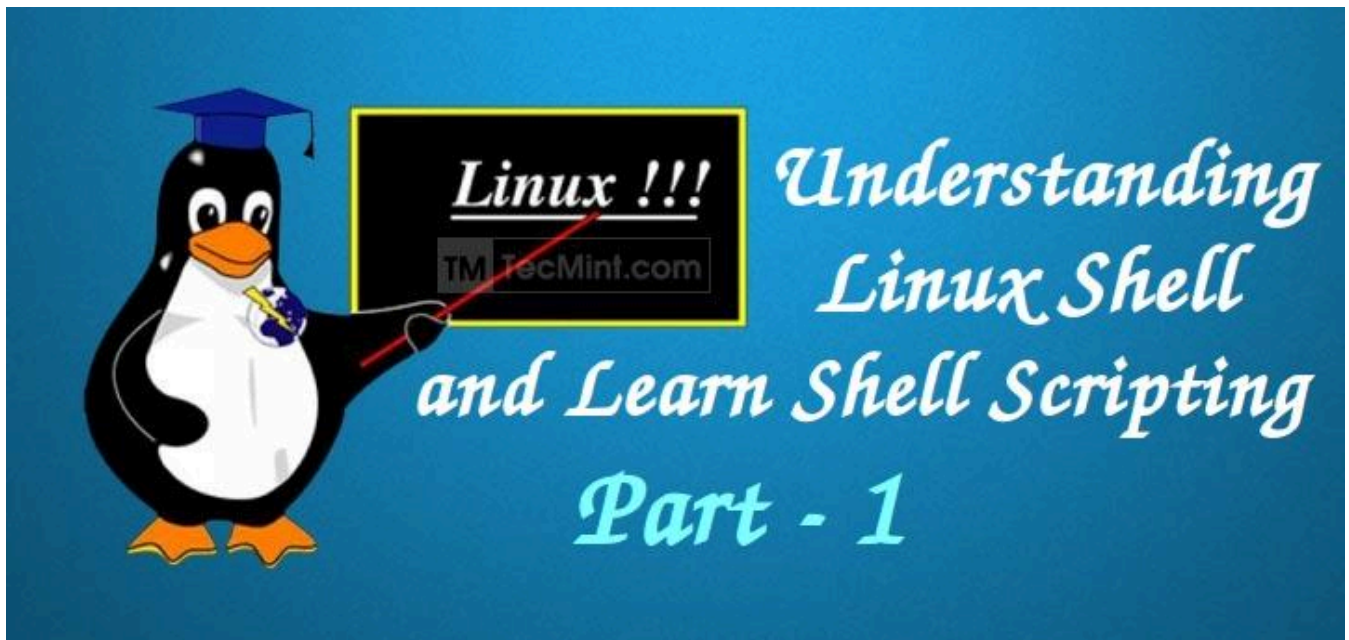
### Learn Basic Mathematical Operations in Bash Scripting – Part IV



### Learn Practical BASH Scripting Projects – Part III

```
[tecmint@TecMint:~]$ ./Server-Health.sh
Wednesday 31 January 2024 02:21:34 PM IST
uptime:
 14:21:34 up 4:40, 1 user, load average: 0.78, 0.91, 0.92
Currently connected:
 14:21:34 up 4:40, 1 user, load average: 0.78, 0.91, 0.92
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WHAT
tecmint   tty7     :0              09:41    4:40m  5:14   0.69s  xfce4-session
-----
Last logins:
tecmint   tty7     Wed Jan 31 09:41    gone - no logout  :0
reboot    system boot Wed Jan 31 09:41    still running     6.2.0-39-generic
tecmint   tty7     Tue Jan 30 09:31 - 15:57 (06:26) :0
-----
Disk and memory usage:
Free/total disk: 3.2G / 3.2G
Free/total memory: 2047 / 32029 MB
```

## 5 Useful Shell Scripts for Linux Newbies – Part II



## Understand Linux Shell and Basic Shell Scripting Tips – Part I



## 10 Useful Chaining Operators in Linux with Practical Examples



## How to Setup SSH Passwordless Login in Linux [3 Easy Steps]

 **1 Comment**

[Leave a Reply](#)

**Ian Williams**

December 10, 2022 at 10:45 pm

Please test your examples. The very first example you give does not do what you say it does. I print the curly brackets, which is not what you show.

I'm going back to c++. At least that does what it says on the tin!

[Reply](#)

## Got Something to Say? Join the Discussion...

*Thank you for taking the time to share your thoughts with us. We appreciate your decision to leave a comment and value your contribution to the discussion. It's important to note that we moderate all comments in accordance with our [comment policy](#) to ensure a respectful and constructive conversation.*

*Rest assured that your email address will remain private and will not be published or shared with anyone. We prioritize the privacy and security of our users.*

☐ Save my name, email, and website in this browser for the next time I comment.

Post Comment



## Do You Enjoy My Blog?

Support from readers like YOU keeps this blog running. Buying me a cup of coffee is a simple and affordable way to show your appreciation and help keep the posts coming!

[Buy Me a Coffee](#)

## Linux Commands and Tools

[5 Commands to Manage File Types and System Time in Linux](#)

[Terminalizer – Record Your Linux Terminal and Generate Animated GIF](#)

[How to Learn dd Command in Linux \[15 Useful Examples\]](#)

[HTTP Prompt – An Interactive Command Line HTTP Client](#)

[Progress – Show Percentage of Copied Data for \(cp, mv, dd, tar\) Commands](#)

[Find Top 15 Processes by Memory Usage with 'top' in Batch Mode](#)

## Linux Server Monitoring Tools

[bmon – A Powerful Network Bandwidth Monitoring and Debugging Tool for Linux](#)

[15 Useful Performance and Network Monitoring Tools for Linux](#)

[How to Install dbWatch to Monitor MySQL Performance in Linux](#)

**[iPerf3 – Test Network Speed/Throughput in Linux](#)**

**[How to Install Nagios XI on Ubuntu 22.04](#)**

**[rtop – An Interactive Tool to Monitor Remote Linux Server Over SSH](#)**

## **Learn Linux Tricks & Tips**

**[2 Ways to Create an ISO from a Bootable USB in Linux](#)**

**[How to Transfer Files Between Two Computers using nc and pv Commands](#)**

**[How to Convert Files to UTF-8 Encoding in Linux](#)**

**[How to Set Limits on User Running Processes in Linux](#)**

**[How to Find MySQL, PHP and Apache Configuration Files](#)**

**[4 Ways to Disable Root Account in Linux](#)**

## **Best Linux Tools**

**[5 Best Open-Source Project Management Tools for Linux in 2024](#)**

**[5 Best Open Source Internet Radio Player for Linux](#)**

**[Top 5 Open-Source Productivity Tools for Linux](#)**

**[12 Best Media Server Software for Linux in 2024](#)**

**[8 Best IRC Clients for Linux in 2024](#)**

**[10 Top Open Source Artificial Intelligence Tools for Linux](#)**

Tecmint: Linux Howtos, Tutorials & Guides © 2024. All Rights Reserved.

The material in this site cannot be republished either online or offline, without our permission.

Hosting Sponsored by : [Linode Cloud Hosting](#)