[](https://linuxsimply.com/for-loop-in-shell-script-examples/)

16 Examples of For Loop in Shell Script [Free Downloads]

Loops are introduced in programming languages to run tasks in a repetitive manner. It iterates a set of statements within a limit depending on conditions. Like every other programming language, [**Bash**](https://linuxsimply.com/bash-in-linux/)also supports the **for loop** to repeat code executions within a range or a limit. In addition, **for loop** in bash scripting can also iterate over command output. In this article, you will get to learn all about the **for loop in Shell Script** with the help of practical examples.

[For Loop Syntax in Bash Scripting 1](#_zc2dvy2e8osb)

[**Shell Script Examples Using For Loop 2**](#_eo69xpby0oyg)

[Basic Shell Scripts with For Loop 3](#_jn7wzfla0qzf)

[Example 1: Print Numbers from 5 to 1 3](#_sro895m239lp)

[Example 2: Print Even Numbers From 1 to 10 3](#_acqteb433mfb)

[Example 3: Print the Multiplication Table of a Number 4](#_wyxtax7yxzbs)

[Example 4: Loop Through a String Character-by-Character 5](#_ut7stw1zwq37)

[Example 5: Loop Through Array Elements 5](#_syxla5nphs8t)

[Example 6: Calculate the Factorial of a Number 5](#_3t4hcgdrbj82)

[Example 7: Calculate the Sum of the First “n” Numbers 6](#_h4ihld5zj0zn)

[Example 8: Find the Smallest and Largest Elements in an Array 6](#_6l5eya813pl9)

[Example 9: Calculate the Average of an Array of Numbers 7](#_pklm0oa04moq)

[Task-Specific Shell Scripts with For Loop 7](#_nt5qkagozsh5)

[Example 1: Take Multiple Filenames and Prints their Contents 8](#_chcygckmqs1h)

[Example 2: Read Lines from a File 8](#_lri6imn4402z)

[Example 3: Loop Through Files with a Specific Extension 9](#_w8wpk6r5ubbu)

[Example 4: Loop Through Files in Multiple Directories 9](#_4fifqkxnqwb4)

[Example 5: Organizes Files in a Directory Based on their File Types 10](#_duq1vkjauoc5)

[Example 6: Loop Through Command Output 11](#_8g2ie1v3s8jq)

[Example 7: Kill All Processes That are Consuming More Than a Certain Amount of CPU 11](#_moamkfeyi7y1)

[**Conclusion 12**](#_nv7xygda73d1)

## **For Loop Syntax in Bash Scripting**

Similar to all the programming languages, the **for loop** in **Shell Scripting** also follows a certain syntax. However, this syntax can vary depending on the purpose of the user. This is because **Bash** provides different **for loop** syntaxes for accessing data stored in different data structures. You can run a **for loop** over a range of values, a list of items, or an array of elements. Syntaxes for each of the cases are given below:

1. **for loop Syntax for Numeric Range**

| for i in {INITIAL\_VAL..TERMINATING \_VAL} do  #code to execute done |
| --- |

OR,

| for (( i=INITIAL\_VAL; i<=TERMINATING \_VALUE; i++ )) do  #code to execute done |
| --- |

OR,

| for i in {INITIAL\_VAL..TERMINATING \_VAL..INCREMENT} do  #code to execute done |
| --- |

1. **for loop Syntax for List of Elements**

| for item in item1 item2 item3 ..itemN do  #code to execute done |
| --- |

1. **for loop Syntax for Array**

| for element in "${arr[@]}" do  #code to execute done |
| --- |

1. **for loop Syntax for Command Output**

| for output in $(LINUX\_COMMAND) do  #code to execute done |
| --- |

## **Shell Script Examples Using For Loop**

The **for loop** in Shell Scripting can be used to achieve numerous tasks similar to other programming languages. This section covers some of the frequent applications of the **for loop**. For your convenience, I have categorized the examples into two groups: **Basic Shell Scripts** and **Task-Specific Shell Scripts**. Follow the categories below to learn more about the applications of **for loop**.

### **Basic Shell Scripts with For Loop**

This section covers a list of examples considered as the basic applications of **for loop**. These examples focus on the utilization of **for loop** in different computational scenarios. Therefore, go through the examples below to learn more about the conceptual usage of **for loop**.

#### **Example 1: Print Numbers from 5 to 1**

You can use the **for loop** in bash to print a number sequence. In this case, specify the condition to stop the loop inside “**for (( ))**”.

**Code:**

| #!/bin/bash  for ((i=5; i>=1; i--)) do  echo $i done |
| --- |

**Output:**

| 5 4 3 2 1 |
| --- |

#### **Example 2: Print Even Numbers From 1 to 10**

To print the even number in a range, check the even number condition inside the for loop before printing the number.

**Code:**

| #!/bin/bash for (( i=1; i<=10; i++ )) do  if [ $((i%2)) == 0 ]  then  echo $i  fi done |
| --- |

**Output:**

| Enter a number: 12 12 x 1 = 12 12 x 2 = 24 12 x 3 = 36 12 x 4 = 48 12 x 5 = 60 12 x 6 = 72 12 x 7 = 84 12 x 8 = 96 12 x 9 = 108 12 x 10 = 120 |
| --- |

#### **Example 3: Print the Multiplication Table of a Number**

Use the simple [**echo command**](https://linuxsimply.com/echo-command-in-linux/) inside a “**for**” loop to display the Multiplication Table of a number.

**Code:**

| #!/bin/bash read -p "Enter a number: " num for (( i=1; i<=10; i++ )) do  echo "$num x $i = $((num\*i))" done |
| --- |

**Output:**

| Enter a number: 12 12 x 1 = 12 12 x 2 = 24 12 x 3 = 36 12 x 4 = 48 12 x 5 = 60 12 x 6 = 72 12 x 7 = 84 12 x 8 = 96 12 x 9 = 108 12 x 10 = 120 |
| --- |

#### **Example 4: Loop Through a String Character-by-Character**

You can use the **for loop** to print a string character by character. For this, the loop needs to initiate from 0 till the length of the string. While moving from the 0th character use echo to print each character.

**Code:**

| #!/bin/bash read -p "Enter a string: " str for ((i=0; i<${#str}; i++)); do  echo ${str:i:1} done |
| --- |

**Output:**

| Enter a string: Linux L i n u x |
| --- |

#### **Example 5: Loop Through Array Elements**

For accessing each array element you can use the **for loop** in the following manner. Indicate the desired array using "**${ARRAY\_NAME[@]}**" and access each item stored in the array.

**Code:**

| #!/bin/bash arr=("mango" "grape" "apple" "cherry" "orange") for item in "${arr[@]}"; do  echo $item done |
| --- |

**Output:**

| **mango grape apple cherry orange** |
| --- |

#### **Example 6: Calculate the Factorial of a Number**

Calculate the factorial of a number by running multiplications inside a “**for**” loop:

**Code:**

| #!/bin/bash read -p "Enter a number: " num temp=1 for (( i=1; i<=$num; i++ )) do  temp=$((temp\*i)) done echo "The factorial of $num is: $temp" |
| --- |

**Output:**

| Enter a number: 6 The factorial of 6 is: 720 |
| --- |

#### **Example 7: Calculate the Sum of the First “n” Numbers**

To calculate the sum of the first n numbers run a for loop and addition operation till n:

**Code:**

| #!/bin/bash read -p "Enter a number: " num sum=0 for (( i=1; i<=$num; i++ )) do  sum=$((sum + i)) done echo "Sum of first $num numbers: $sum" |
| --- |

**Output:**

| Enter a number: 100 Sum of first 100 numbers: 5050 |
| --- |

#### **Example 8: Find the Smallest and Largest Elements in an Array**

For finding the smallest and largest element in a given array, first initialize a small and a large number. Then compare the array elements with these numbers inside any loop.

**Code:**

| #!/bin/bash arr=(24 27 84 11 99) echo "Given array: ${arr[\*]}" s=100000 l=0 for num in "${arr[@]}" do  if [ $num -lt $s ]  then  s=$num  fi  if [ $num -gt $l ]  then  l=$num  fi done echo "The smallest element: $s" echo "The largest: $l" |
| --- |

**Output:**

| Given array: 24 27 84 11 99 The smallest element: 11 The largest: 99 |
| --- |

#### **Example 9: Calculate the Average of an Array of Numbers**

Find the sum of array elements using a “**for**” loop and divide it by the number of elements i.e. **${#arr[@]}**.

**Code:**

| #!/bin/bash echo "Enter an array of numbers (separated by space):" read -a arr sum=0 for i in "${arr[@]}" do  sum=$((sum+i)) done avg=$((sum/${#arr[@]})) echo "Average of the array elements: $avg" |
| --- |

**Output:**

| Enter an array of numbers (separated by space): 23 45 11 99 100 Average of the array elements: 55 |
| --- |

### **Task-Specific Shell Scripts with For Loop**

In addition to the conceptual bash scripts, in this section, you will find some task-specific script examples. These scripts are mostly related to the regular process that you run on your system. Hence, follow the examples below to improve your experience with **Shell Scripting**.

#### **Example 1: Take Multiple Filenames and Prints their Contents**

The below script is for reading the contents of multiple files. It will take the file names as user input and display their contents on the screen. If any filename does not exist, it will show a separate error message for that file.

**Code:**

| #!/bin/bash read -p "Enter the file names: " files IFS=' ' read -ra array <<< "$files" for file in "${array[@]}" do if [ -e "$file" ]; then  echo "Contents of $file:"  cat "$file"  else  echo "Error: $file does not exist" fi done |
| --- |

**Output:**

| Enter the file names: message.txt passage.txt Contents of message.txt: "Merry Christmas! May your happiness be large and your bills be small." Contents of passage.txt: The students told the headmaster that they wanted to celebrate the victory of the National Debate Competition. |
| --- |

#### **Example 2: Read Lines from a File**

The following script can be used to read and display each line from a file. Here, a filename is taken as user input and the **IFS**(Internal Field Separator) is set to **New Line (\n)** which enables the **for loop** to recognize each line individually inside the file.

**Code:**

| #!/bin/bash read -p "Enter a filename: " file echo Lines: IFS=$'\n'  for line in $(cat "$file"); do  echo "$line" done |
| --- |

**Output:**

| Enter a filename: textfile.txt Lines: I wandered lonely as a cloud That floats on high o'er vales and hills, When all at once I saw a crowd, A host, of golden daffodils; |
| --- |

#### **Example 3: Loop Through Files with a Specific Extension**

The given script takes a file extension as user input and looks for the files with that extension within the current directory using **for loop**. Inside the loop, it prints each file name.

**Code:**

| #!/bin/bash read -p "Enter a file extension (i.e. txt, jpg, ..): " ext for file in \*.$ext; do  echo $file done |
| --- |

**Output:**

| Enter a file extension (i.e. txt, jpg, ..): txt file1.txt file2.txt textfile.txt urls.txt |
| --- |

#### **Example 4: Loop Through Files in Multiple Directories**

The following script takes a list of directory names from the user. The **for loop** goes through each file of every directory and prints out its name.

**Code:**

| #!/bin/bash read -p "Enter a list of directories: " directories for dir in $directories; do  for file in $dir/\*; do  echo $file  done done |
| --- |

**Output:**

| Enter a list of directories: Documents Pictures Documents/list1.txt Documents/list2.txt Documents/message.txt Documents/packets.pcap Documents/ping.txt Documents/poem.txt Pictures/Screenshots Pictures/ss1.png Pictures/ss2.png |
| --- |

#### **Example 5: Organizes Files in a Directory Based on their File Types**

The script given below organizes files in a directory depending on their type. The user needs to give a destination directory path to organize the files along with the source directory path.

This script will create five directories: 1) Documents, 2) Images, 3) Music, 4) Videos, and 5) Others only if they do not already exist on the destination path. Then, it will check all the files and their extension and move them to the corresponding directory. If there is any unknown file extension, then the script will move the file to the Others Directory.

**Code:**

| #!/bin/bash  # Specify the source and destination directories read -p "Enter path to the source directory: " source\_dir read -p "Enter path to the destination directory: " dest\_dir  # Create the destination directories if they don't exist mkdir -p "${dest\_dir}/Documents" mkdir -p "${dest\_dir}/Images" mkdir -p "${dest\_dir}/Music" mkdir -p "${dest\_dir}/Videos" mkdir -p "${dest\_dir}/Others"  # Move files to the appropriate directories based on their extensions for file in "${source\_dir}"/\*; do  if [ -f "${file}" ]; then  extension="${file##\*.}"  case "${extension}" in  txt|pdf|doc|docx|odt|rtf)  mv "${file}" "${dest\_dir}/Documents"  ;;  jpg|jpeg|png|gif|bmp)  mv "${file}" "${dest\_dir}/Images"  ;;  mp3|wav|ogg|flac)  mv "${file}" "${dest\_dir}/Music"  ;;  mp4|avi|wmv|mkv|mov)  mv "${file}" "${dest\_dir}/Videos"  ;;  \*)  mv "${file}" "${dest\_dir}/Others"  ;;  esac  fi done echo "Files organized successfully!" |
| --- |

**Output:**

| Enter path to the source directory: /home/anonnya/Downloads Enter path to the destination directory: /home/anonnya/Downloads\_Organized Files organized successfully! |
| --- |

#### **Example 6: Loop Through Command Output**

You can loop through a command’s output using the **for loop**. The given script will take a command from the user input and display it as a list.

**Code:**

| #!/bin/bash read -p "Enter a command: " comm for result in $($comm); do  echo $result done |
| --- |

**Output:**

| Enter a command: ls Documents list1.txt list2.txt message.txt packets.pcap ping.txt poem.txt |
| --- |

#### **Example 7: Kill All Processes That are Consuming More Than a Certain Amount of CPU**

This script takes a CPU usage percentage as user input and terminates all the running processes that are consuming more than the entered CPU threshold. If there is no process above that threshold, then it returns a message saying there are no such processes.

**Code:**

| #!/bin/bash read -p "Enter CPU usage threshold: " threshold if [ "$(ps -eo pid,%cpu | awk -v t=$threshold '$2 > t {print $1}' | wc -c)" -gt 0 ]; then for pid in $(ps -eo pid,%cpu | awk -v t=$threshold '$2 > t {print $1}') do  kill $pid done echo "All processes consuming more than $threshold% CPU killed." else echo "There are no processes consuming more than $threshold% CPU." fi |
| --- |

**Output:**

| Enter CPU usage threshold: 10 There are no processes consuming more than 10% CPU. |
| --- |

## **Conclusion**

This article presents hands-on examples on the topic **for loop** in [**Shell**](https://linuxsimply.com/what-is-a-shell-linux/) **Scripting**. It covers all possible **for loop** syntaxes that can be useful to a user while creating loop-based scripts. Moreover, the practical examples are divided into two categories: Basic Shell Scripts and Task-Specific Shell Scripts. Therefore, users from beginner to advanced levels can utilize the provided materials to improve their experience of Bash Scripting.

|  | **Prepared By:** [**Susmit Das Gupta**](https://linuxsimply.com/susmit-das-gupta/) | **Web View:** [**16 Examples of For Loop in Shell Script [Free Downloads]**](https://linuxsimply.com/for-loop-in-shell-script-examples/)  **Copyright ©2024 linuxsimply.com| All rights reserved** |
| --- | --- | --- |