TutorialKart

Bash For Loop – Syntax and Examples

Bash For Loop

Bash For loop is a statement that lets you iterate specific set of statements over series of words in a string, elements in a sequence, or elements in an array.

In this tutorial, we will go through following topics.

Bash For Loop – Iterate over Elements of Array

The syntax to loop or iterate over elements of an array using Bash For loop is

```
#!/bin/bash

arr=( "element1" "element2" . . "elementN" )

for i in "${arr[@]}"
do
    echo $i
    #statement(s)
done
```

For each element in `arr` the statements from **do** till **done** are executed, and each element could be accessed as `i` within the for loop for respective iteration.

Bash For Loop Example

In this For loop example, we will take an array of three strings, and iterate over the elements of array.

Bash Script File

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

```
#!/bin/bash

# declare an array
arr=( "bash" "shell" "script" )

# for loop that iterates over each element in arr
for i in "${arr[@]}"
do
    echo $i
done
```

When the above **bash for loop** example is run in Terminal, you will get the following output.

Output

```
~$ ./bash-for-loop-example-4
bash
shell
script
```

Bash For Loop – Whitespaces in String as Word Separators

The syntax of Bash For loop to consider whitespaces in given string as word separators is

```
#!/bin/bash

for word in $str;
do
    #statement(s)
done
```

str is a string

for each **word** in **str** the statements from **do** till **done** are executed, and **word** could be accessed within the **for loop** for respective iteration.

Example

In the following example, we will take a string with words separated by white spaces, and iterate over the words using for loop.

Bash Script File

```
#!/bin/bash

str="this example
demonstrates the for loop
that considers all white-space
```

```
characters as word separators"

## now loop through the above array
for i in $str;
do
   echo "$i"
done
```

When the above **bash for loop** example is run in Terminal, you will get the following output.

Output

```
~$ ./bash-for-loop-example
this
example
demonstrates
the
for
loop
that
considers
all
white-space
characters
as
word
separators
```

Bash For Loop – Each Line in String as Element

The syntax of Bash For Loop to consider each line in string as element is

```
#!/bin/bash

for line in "$str";
do
    #statement(s)
done
```

str is a string

for each **line** that is a line in **str**, statements from **do** till **done** are executed, and **line** could be accessed within the **for loop** for respective iteration.

Note: Observe that the only difference between first type of for loop and this one is the double quotes around string variable.

Example

In the following example, we will take a string with four lines. We will use for loop to iterate over the elements in string separated by new line.

Bash Script File

```
#!/bin/bash

str="this example
demonstrates the for loop
that considers new line
characters as line separators"


## loop through each line in above string
for line in "$str"
do
   echo "$line"
done
```

When the above **bash for loop** example is run in Terminal, we will get the following output.

Output

```
~$ ./bash-for-loop-example-2
this example
demonstrates the for loop
that considers new line
characters as word separators
```

Bash For Loop – Iterate Over a Sequence

The syntax of Bash For loop to iterate over a sequence of numbers is

```
#!/bin/bash

for i in `seq m n`;
do
    #statement(s)
done
```

for loop is iterated for each element `i` in the sequence from `m` to `n` . The element in the sequence, `i` is accessible during the iteration.

Example

In the following Bash Script, we will use For Loop to iterate over a sequence of numbers.

Bash Script File

```
#!/bin/bash

for i in `seq 1 10`;
do
    echo $i
done
```

When the above bash for loop example is run in Terminal, you will get the following in the output.

Output

```
~$ ./bash-for-loop-example-3
1
2
3
4
5
6
7
8
9
10
```

Bash For Loop – Use Counter to Iterate Over Elements of Array

The syntax of Bash For Loop to use a variable for counter to iterate over elements of an array is

```
#!/bin/bash

array=( "element1" "element2" . . "elementN" )

arraylength=${#array[@]}

for (( i=1; i<${arraylength}+1; i++ ));
do
    echo $i : " ${array[$i-1]}
    #statement(s)
done
```

Length of an array could be calculated using `${#array[@]}` and the the statements from **do** till **done** are could iterated based on a condition. The syntax of for loop might look similar to that of Java For Loop.

Example

In the following Bash Script, we will use For Loop to iterate over an Array using a counter variable.

Bash Script File

```
#!/bin/bash

#array variable
arr=( "bash" "shell" "script" )

# get length of an array to a variable
arrlength=${#arr[@]}

# for loop to read all values and indexes
for (( i=1; i<${arrlength}+1; i++ ));
do
  echo $i " : " ${arr[$i-1]}
done
```

When the above **bash for loop** example is run in Terminal, we will get the following output.

Output

```
~$ ./bash-for-loop-example-5
1  :  bash
2  :  shell
3  :  script
```

Bash For Loop – Break Command

The syntax of For Loop with break command, to break the for loop abruptly even before the for condition fails is

```
#!/bin/bash

array=( "element1" "element2" . . "elementN" )

for i in "${array[@]}"
do
    #statement(s)

    if [ BREAKING_CONDITION ]; then
        break
    fi
done
```

BREAKING_CONDITION decides when to break the for loop prematurely. When the breaking condition evaluates to TRUE, then break statement is executed. **break** command breaks the loop that is immediate to the break statement.

Example

In the following Bash Script, we will write a For Loop containing break statement. The break statement is surrounded in an if statement, which executes the body when the element is “script”. In other words, we are

breaking the for loop if the element is “script”.

Bash Script File

```
#!/bin/bash

# declare an array
arr=( "bash" "shell" "script" "language" )

# for loop that iterates over each element in arr
for i in "${arr[@]}"
do
    echo $i
    # break for loop based on a condition
    if [ $i == "script" ]; then
        break
    fi
done
```

When the above **bash for loop** example is run in Terminal, we will get the following output.

Output

```
~$ ./bash-for-loop-example-6
bash
shell
script
```

Conclusion

In this Bash Tutorial – **Bash For Loop**, we have learnt to iterate specific set of statements over words, lines in a string, elements in a sequence or elements in an array with the help of Bash Script examples.

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

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