BASH Programming

Bash Range

1 year ago • by Fahmida Yesmin

You can iterate the sequence of numbers in bash in two ways. One is by using the **seq** command, and another is by specifying the range in for loop. In **the seq** command, the sequence starts from one, the number increments by one in each step, and print each number in each line up to the upper limit by default. If the number starts from the upper limit, then it decrements by one in each step. Normally, all numbers are interpreted as a floating-point, but if the sequence starts from an integer, the decimal integers will print. If the seq command can execute successfully, then it returns 0; otherwise, it returns any non-zero number. You can also iterate the sequence of numbers using for loop with range. Both **seq** command and for loop with range are shown in this tutorial by using examples.

RELATED LINUX HINT POSTS
Bash Set Command
How to Run a Bash Script
Delete a Folder in Bash
Bash read command
Different Examples of Checking Whether a Sting Contains a Substring or Not
How to Use $IFS in Bash?
Creating Bash Infinite Loop by Example Scripts

The options of seq command:

You can use **the seq** command by using the following options.
-w

This option is used to pad the numbers with leading zeros to print all numbers with equal width.
-f format

This c

forma

-s str

This option is used to separate the numbers with string. The default value is a newline ('\n').

Examples of seq command:

You can apply the seq command in three ways. You can use only the upper limit or upper and lower limit or upper and lower limit with increment or decrement value of each step. Different uses of the seq command with options are shown in the following examples.

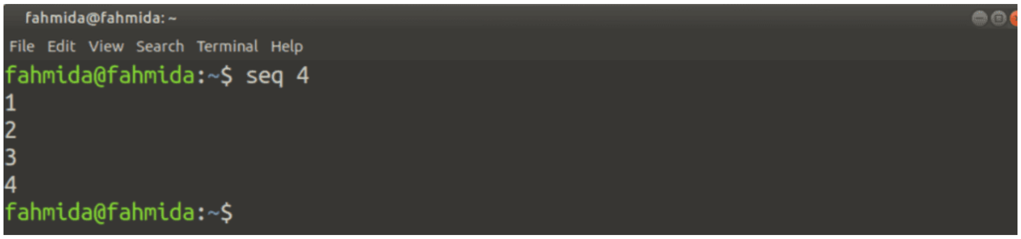
Example-1: seq command without the option

When the only upper limit is used, the number will start from 1 and increment by one in each step. The following command will print the number from 1 to 4.

```
$ seq 4
```

Output:

The following output will appear after executing the above command.



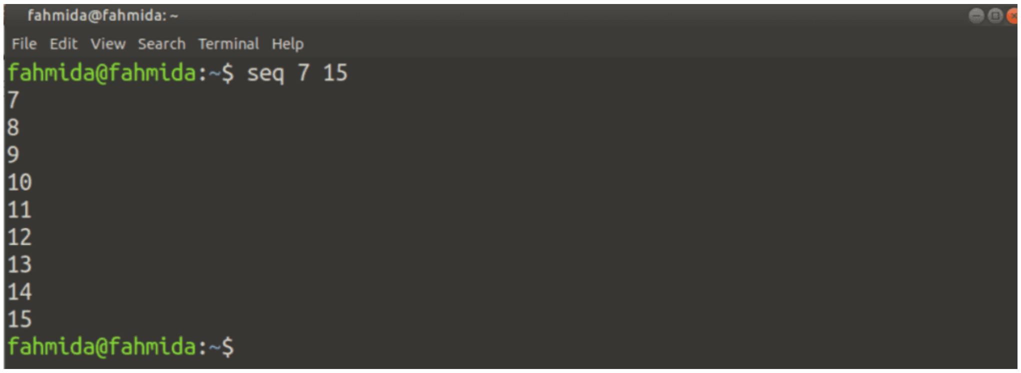
```
fahmida@fahmida:~$ seq 4
1
2
3
4
fahmida@fahmida:~$
```

When the two values are used with the seq command, the first value will be used as the starting number, and the second value will be used as the ending number. The following command will print the number from 7 to 15.

```
$ seq 7 15
```

Output:

The following output will appear after executing the above command.



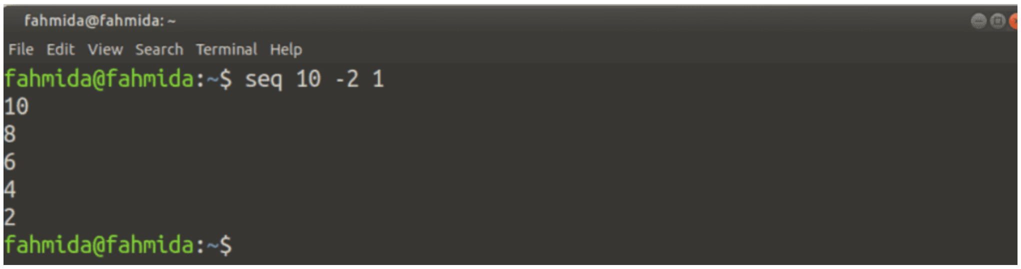
```
fahmida@fahmida:~$ seq 7 15
7
8
9
10
11
12
13
14
15
fahmida@fahmida:~$
```

When using three values with the seq command, the second value will be used as an increment or decrement value for each step. For the following command, the starting number is 10, the ending number is 1, and each step will be counted by decrementing 2.

```
$ seq 10 -2 1
```

Output:

The following output will appear after executing the above command.

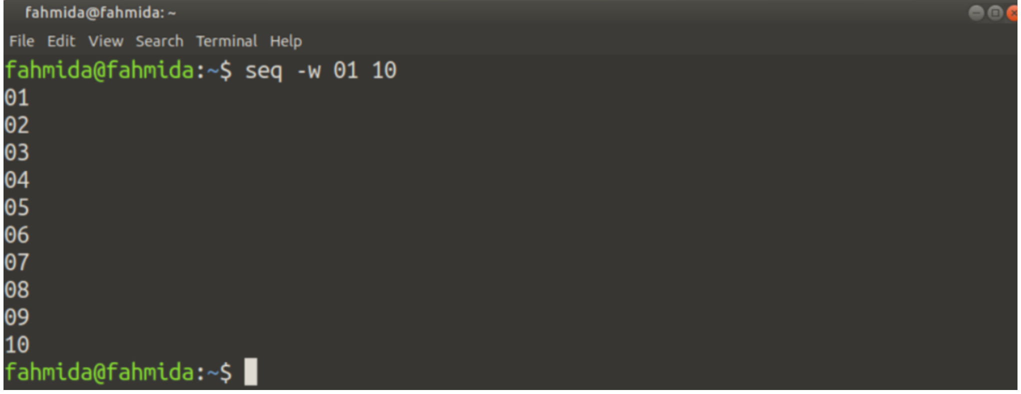


```
fahmida@fahmida:~$ seq 10 -2 1
10
8
6
4
2
fahmida@fahmida:~$
```

Example-2: seq with –w option

The following command will print the output by adding leading zero for the number from 1 to 10.

```
$ seq -w 01 10
```



```
fahmida@fahmida:~$ seq -w 01 10
01
02
03
04
05
06
07
08
09
10
fahmida@fahmida:~$
```

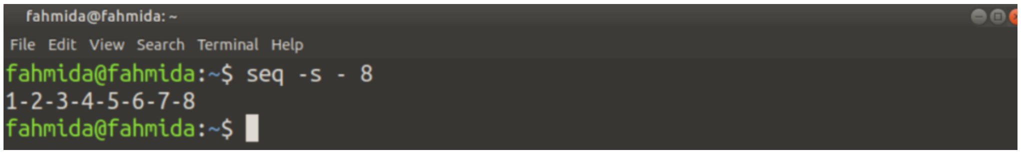
Example-3: seq with –s option

The following command uses "-" as a separator for each sequence number. The sequence of numbers will be printed by adding "-" as the separator.

```
$ seq -s - 8
```

Output:

The following output will appear after executing the above command.



```
fahmida@fahmida:~$ seq -s - 8
1-2-3-4-5-6-7-8
fahmida@fahmida:~$
```

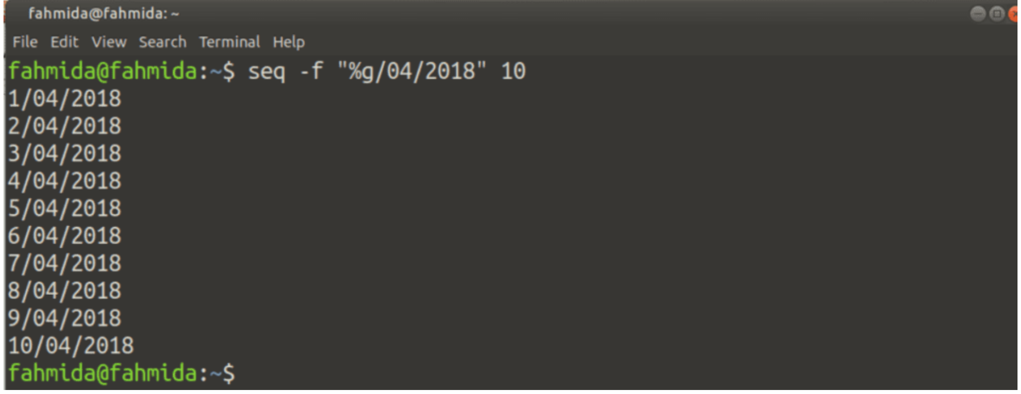
Example-4: seq with -f option

The following command will print 10 date values starting from 1. Here, the "%g" option is used to add sequence numbers with other string values.

```
$ seq -f "%g/04/2018" 10
```

Output:

The following output will appear after executing the above command.



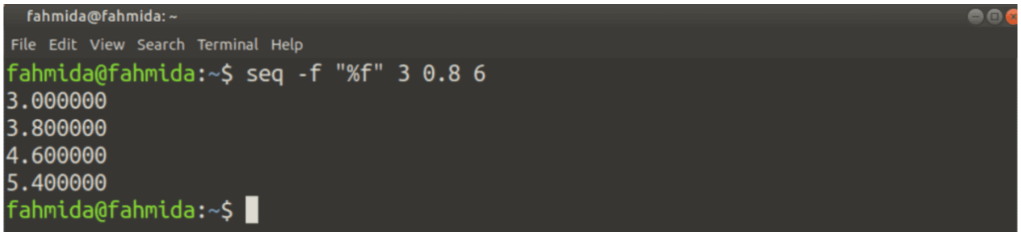
```
fahmida@fahmida:~$ seq -f "%g/04/2018" 10
1/04/2018
2/04/2018
3/04/2018
4/04/2018
5/04/2018
6/04/2018
7/04/2018
8/04/2018
9/04/2018
10/04/2018
fahmida@fahmida:~$
```

The following command is used to generate the sequence of floating-point numbers using "%f". Here, the number will start from 3 and increment by 0.8 in each step, and the last number will be less than or equal to 6.

```
$ seq -f "%f" 3 0.8 6
```

Output:

The following output will appear after executing the above command.



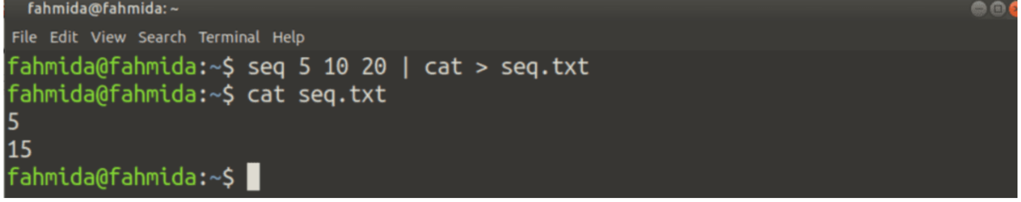
```
fahmida@fahmida:~$ seq -f "%f" 3 0.8 6
3.000000
3.800000
4.600000
5.400000
fahmida@fahmida:~$
```

Example-5: Write the sequence in a file

If you want to save the sequence of numbers into a file without printing in the console, you can use the following commands. The first command will print the numbers to a file named "**seq.txt**". The number will generate from 5 to 20 and increment by 10 in each step. The second command will print the content of the "**seq.txt**" file in the terminal.

```
$ seq 5 10 20 | cat > seq.txt
$ cat seq.txt
```

Output:



```
fahmida@fahmida:~$ seq 5 10 20 | cat > seq.txt
fahmida@fahmida:~$ cat seq.txt
5
15
fahmida@fahmida:~$
```

Example-6: Use of `seq` to create the filename

Suppose you want to create files named fn1 to fn10 using for loop with seq. Create a file named “sq1.bash” and add the following code. For loop will iterate for 10 times using `seq` command and create 10 files in the sequence fn1, fn2,fn3…..fn10.

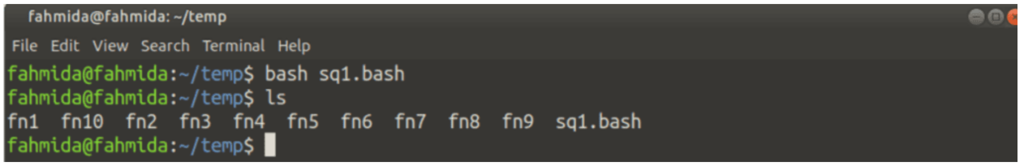
```
#!/bin/bash
# Generate 10 sequence numbers
for i in `seq 10`
do
    # Create the filename
    touch fn$i
done
```

Run the following commands to execute the bash file’s code and check whether the files are created or not.

```
$ bash sq1.bash
$ ls
```

Output:

The following output will appear after executing the above commands.



```
fahmida@fahmida:~/temp$ bash sq1.bash
fahmida@fahmida:~/temp$ ls
fn1  fn10  fn2  fn3  fn4  fn5  fn6  fn7  fn8  fn9  sq1.bash
fahmida@fahmida:~/temp$
```

Examples of for loop with range:

The alternative of the `seq` command is range. You can use range in for loop to generate the sequence of numbers like `seq`. The range expression is defined by using curly brackets and double dots. The syntax of the range expression is shown below.

Syntax:

```
{Start..Stop[..Increment]}
```

Here, the value of the **Start** and **Stop** can be any positive integer or character. These values are mandatory for defining range expression and separated by double dots. The value of the **Increment** can be any positive or negative integer, and it is optional. This value is defined after the **Stop** value with double dots. Different uses of range expression have shown in the following examples.

Example-7: Using range with Start and Stop values

Create a bash file named “**sq2.bash**” with the following code. The loop will iterate for 5 times and print the square root of each number in each iteration.

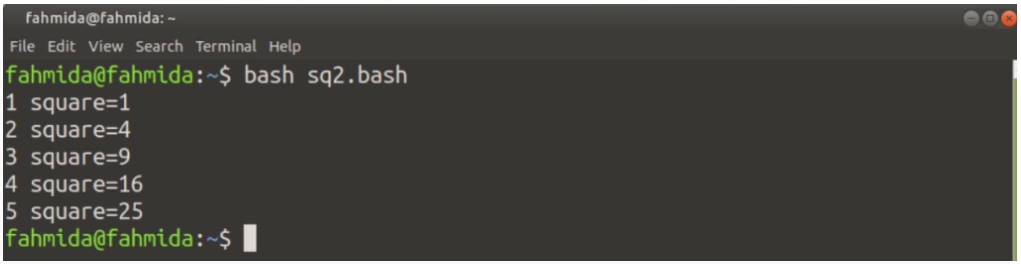
```
#!/bin/bash
# Generate the series of numbers from 1 to 5
for n in {1..5}
do
    # Calculate the square root
    ((result=n*n))
    # Print the square value
    echo $n square=$result
done
```

Run the following command to execute the above script.

```
$ bash sq2.bash
```

Output:

The following output will appear after executing the script.



```
fahmida@fahmida:~$ bash sq2.bash
1 square=1
2 square=4
3 square=9
4 square=16
5 square=25
fahmida@fahmida:~$
```

Example-8: Using range with positive Increment value

By default, the number is increment by one in each step in a range like seq. You can also change the increment value in range. Write the following code in a bash file named “**sq3.bash**“. The for loop in the script will iterate 5 times; each step is incremented by 2 and print all odd numbers between 1 to 10.

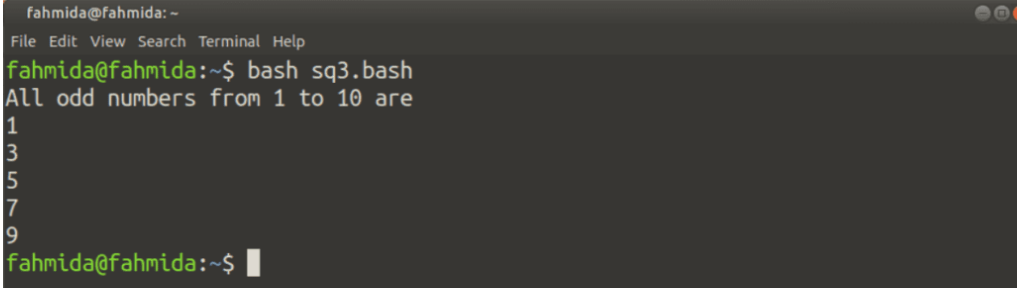
```
#!/bin/bash
echo "All odd numbers from 1 to 10 are"
# Generate odd numbers from 1 to 10
for i in {1..10..2}
do
    # Print the value
    echo $i;
done
```

Run the following command to execute the above script.

```
$ bash sq3.bash
```

Output:

The following output will appear after executing the script.



```
fahmida@fahmida:~$ bash sq3.bash
All odd numbers from 1 to 10 are
1
3
5
7
9
fahmida@fahmida:~$
```

Example-9: Using range with leading zero

The sequence of numbers with leading zero can be generated by using range also. Create a bash file named **seq4.bash** with the following script to generate five sequential numbers with leading zero from 1 to 5 by adding the string 'ID-' at the front of each number.

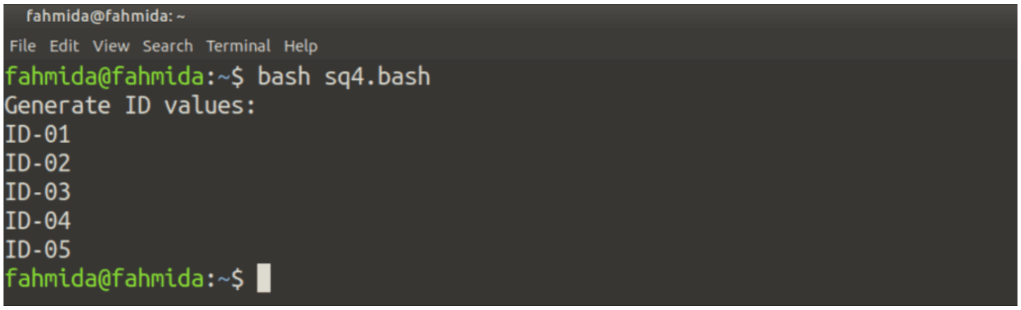
```
#!/bin/bash

echo "Generate ID values:"

# Define the start value of the range with leading zero
for i in {01..5}
do
     # Print the value with 'ID-'
     echo "ID-$i"
done
```

Run the following command to execute the above script.

```
$ bash sq4.bash
```

Output:

The following output will appear after executing the script.



```
fahmida@fahmida:~$ bash sq4.bash
Generate ID values:
ID-01
ID-02
ID-03
ID-04
ID-05
fahmida@fahmida:~$
```

Example-10: Using range with negative increment value

```
#!/bin/bash

echo "Generate even numbers in descending order:"

# Define the start value of the range with a negative increment value
for i in {10..0..-2}
do
     # Print the value with 'ID-'
     echo "The value is $i"
done
```

Run the following command to execute the above script.

```
$ bash sq5.bash
```

Output:

The following output will appear after executing the script.

Example-11: Generate sequential numbers with character and number values

Create a bash file named **seq6.bash** with the following script to generate output based on two range values. The outer loop will generate three characters from A to C, and the inner loop will generate three numbers from 1 to 3.

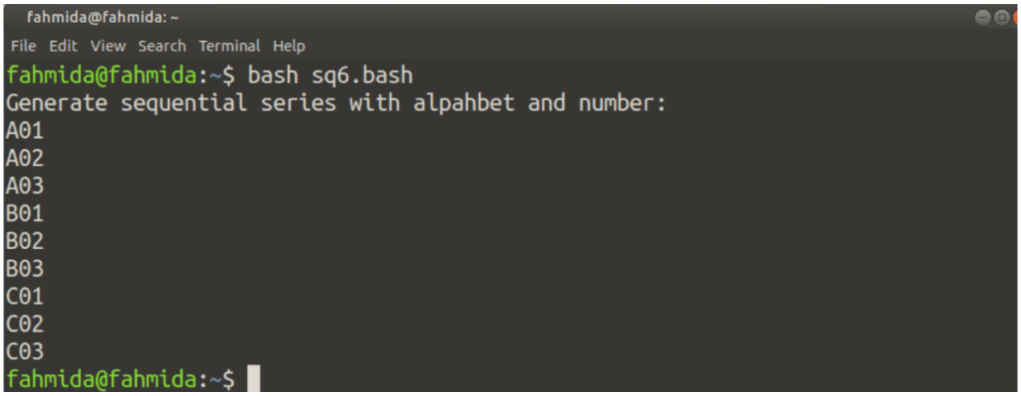
```
#!/bin/bash

echo "The sequential series with alphabet and number:"

# Define the range with alphabets
for val1 in {A..C}
do
    # Define the range with numbers
    for val2 in {01..3}
    do
        # Print the value by conctenating the alphabet and number
        echo $val1$val2
    done

done
```

Run the following command to execute the above script.

```
$ bash sq6.bash
```

Output:

The following output will appear after executing the script.



```
fahmida@fahmida:~$ bash sq6.bash
Generate sequential series with alpahbet and number:
A01
A02
A03
B01
B02
B03
C01
C02
C03
fahmida@fahmida:~$
```

Example-12: Use of range with prefix and suffix

Create a bash file named **seq7.bash** with the following script to generate 5 file names by adding prefix and suffix with the range of numbers. In the script, '**profile**' is the prefix value and '.**png**' is the suffix value.

```
#!/bin/bash

echo "The series of filenames are:"

# Generate five filenames with the extension 'png'
for name in profile{1..5}.png; do
    # Print the filename
    echo "Filename: $name"
done
```

Run the following command to execute the above script.

```
$ bash sq7.bash
```

Output:

The following output will appear after executing the script.

Conclusion:

Two ways to generate the sequence of numbers have been shown in this tutorial by using multiple examples. One way is the `seq` command, and another way is to use range with for loop. The bash users will be able to generate the sequence of numbers efficiently after practicing the examples of this tutorial.