# 

# 

# 

# 

# 

# **Bash seq and shuf commands documentation**

# **along with examples, usage and tests**

# **Bash Range: How to iterate over sequences generated on the shell**

There is a possibility to iterate the sequence of numbers in bash by two ways. One is by using **seq** command and another is by specifying range in for loop. In **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 upper limit then it decrements by one in each step. Normally, all numbers are interpreted as floating point but if the sequence starts from integer then the list of decimal integers will print. If seq command can execute successfully then it returns 0, otherwise it returns any non-zero number. The iteration of the sequence of numbers can also be done by using for loop with range.

## 

## **The options of seq command:**

The **seq** command can be provided 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 option is used to print number with particular format. Floating number can be formatted by using %f, %g and %e as conversion characters. %g is used as default.

-s string

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

### **Examples of seq command:**

The **seq** command can be applied by three ways. Using only upper limit or upper and lower limit or upper and lower limit with increment or decrement value of each step .

#### **1: seq command without option**

When only upper limit is used then 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

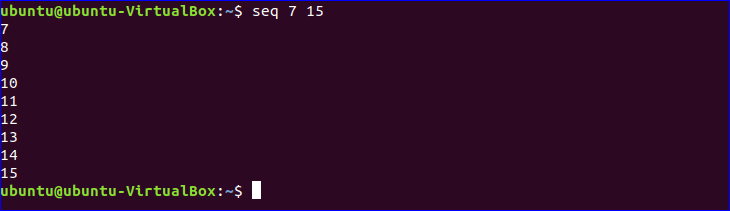
**Output:**

****

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

*$* seq 7 15

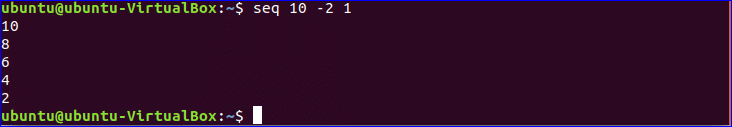
**Output:**

****

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

*$* seq 10 -2 1

**Output:**

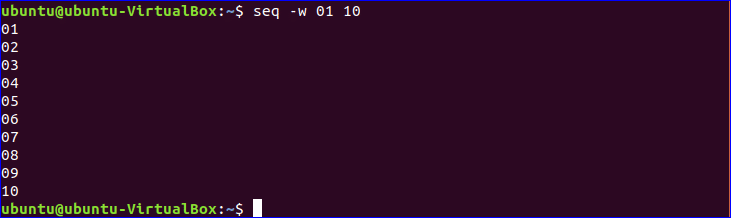
****

#### **2: seq with –w option**

The following command will print the output by adding leading zero for the number from 1 to 9.

*$* seq -w 0110

**Output:**

****

#### **3: seq with –s option**

The following command uses “-“ as separator for each sequence number. The sequence of numbers will print by adding “-“ as separator.

*$* seq -s - 8

**Output:**

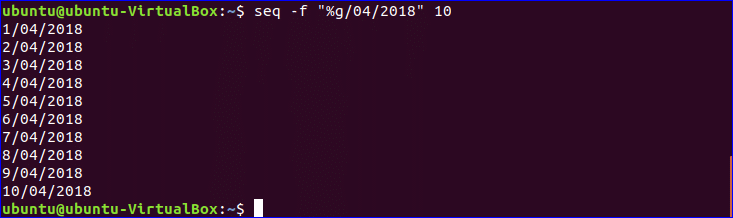
****

#### **4: seq with -f option**

The following command will print 10 date values starting from 1. Here, “%g” option is used to add sequence number with other string value.

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

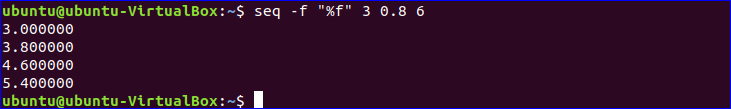
**output:**

****

The following command is used to generate the sequence of floating point number 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:**

****

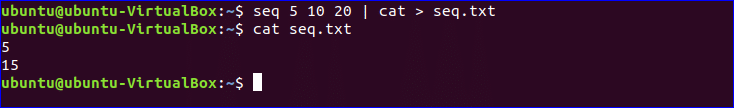
#### **5: Write the sequence in a file**

If the sequence of number needs to be saved into a file without printing in the console then it is more appropriate to 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 is used to view the content of “**seq.txt”** file.

$ seq 5 10 20 | cat &gt; seq.txt

$ cat seq.txt

**Output:**

****

### **6: Using seq in for loop**

If there is need 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*

for i in `seq 10`

do

touch fn.$i

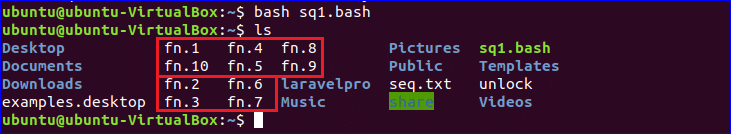
done

**Output:**

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

$ bash sq1.bash

$ ls



### **For loop with range:**

#### **7: For loop with range**

The alternative of seq command is range. Range in for loop is used to generate sequence of numbers like seq. After writing the following code in a bash file named “**sq2.bash**”, the loop will iterate for 5 times and print the square root of each number in each step.

*#!/bin/bash*

for n in {1..5}

do

((result=n\*n))

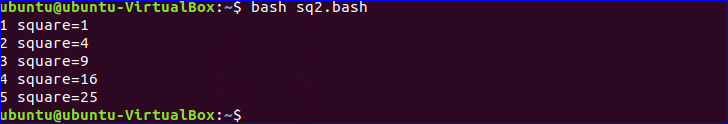
echo $n square=$result

done

**Output:**

Run the command to execute the script of the file.

*$* bash sq2.bash



### **8: For loop with range and increment value**

By default, the number is increment by one in each step in range like seq. The increment value in range can also be changed. After writing the following code in a bash file named “**sq3.bash**”, the for loop in the script will iterate for 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"

for i in {1..10..2}

do

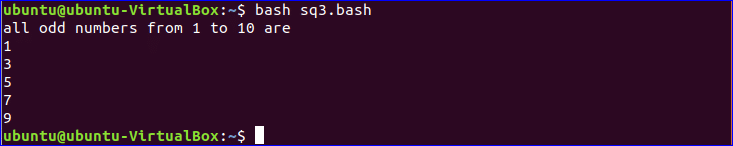
echo $i;

done

**Output:**

Run the command to execute the script of the file.

*$* bash sq3.bash



# **Bash shuf command**

Shuf is one of those commands that most bash programmers have not heard. For those who have, the experience itself is often eerie, like a whisper coming from a dark distant terminal where long-forgotten commands end up. Those that venture further find shuf and are never the same again.

# **What is shuf?**

shuf is a command-line utility like sort included in Coreutils.

## **Where to find help on shuf**

shuf comes with an –help long option.

**Command**

*# shuf --help*

Usage: shuf [OPTION]... [FILE]

or: shuf -e [OPTION]... [ARG]...

or: shuf -i LO-HI [OPTION]...

Write a random permutation of the input lines to standard output.

With no FILE, or when FILE is -, read standard input.

Mandatory arguments to long options are mandatory for short options too.

-e, --echo treat each ARG as an input line

-i, --input-range=LO-HI treat each number LO through HI as an input line

-n, --head-count=COUNT output at most COUNT lines

-o, --output=FILE write result to FILE instead of standard output

--random-source=FILE get random bytes from FILE

-r, --repeat output lines can be repeated

-z, --zero-terminated line delimiter is NUL, not newline

--help display this help and exit

--version output version information and exit

## **The Three Ways of shuf**

There are three ways to use the shuf command which are:

1. file shuf
2. list shuf
3. range shuf

Each way has its advantages. Knowledge of all the ways beforehand may reduce the need to use other external commands in conjunction with the shuf command.

### **file shuf**

File shuf is the most common way shuf is used in command line. When the -e or -i option are not included in the options, shuf will operate as file shuf. That is, the input to be shuffled will be a file whether standard input or any given file. The last word in the parameter list may be a filename. In case this parameter is omitted file is taken to be standard input from the shell or piped. A – may be included following the convention that standard input is inferred.

#### **Usage**

Usage: shuf [OPTION]... [FILE]

#### **Implicit file as standard input**

In this way file is omitted from the arguments of the shuf command. Following the convention, it can opt to include a – in place of file to indicate the file to be taken as standard input.

**Commands**

{

seq 3 | shuf

}

**Output**

1

3

2

#### **Explicit file as standard input**

**Commands**

{

seq 3 | shuf -

}

**Output**

3

1

2

**Notes**

(1) Adding a – at the end of shuf serves as a visual cue improving readability in bash scripts.

#### **File as the name of file**

In this way, a filename is specified as file in the arguments of the shuf command.

**Shuffle input lines from the terminal**

**Commands**

{

shuf /dev/fd/1

}

asdf

sdf

df

f

Ctrl-D

**Output**

df

f

asdf

sdf

**Notes**

(1) The above shuf command shuf /dev/fd/1 is equivalent to shuf –

(2) Termination of input lines through Ctrl-D is required

#### **Shuffle lines in file**

**Commands**

{

seq 3 > file;

shuf file;

rm -f file

}

**Output**

2

1

3

### **list shuf**

In the last way to shuf, it can be operated on a file or input piped into the shuf command. In this way to shuf, there is allowing of input lines to be specified as arguments of the shuf command using the -e option, forcing shuf to operate as list shuf.

#### **Usage**

Usage: shuf -e [OPTION]... [ARG]...

#### **Ways to specify list args**

#### **Type input as args**

**Commands**

{

shuf -e 1 2 3

}

**Output**

1

3

2

**Notes**

**(1) The above shuf command shuf -e 1 2 3 is equivalent to seq 3 | shuf –**

#### **Variable as args**

**Commands**

{

var="1 2 3";

shuf -e *${var}*

}

<strong>Output</strong>

[cc lang="bash"]

3

1

2

#### **Parameter expansion as args**

**Commands**

{

shuf -e {1..3}

}

**Output**

1

2

3

#### **Command substitution as args**

**Commands**

{

shuf -e $( seq 3 )

}

**Output**

3

2

1

### **range shuf**

This last way to shuf is unlike the previous ways introduced. Instead of specifying a file or args in the command line, it requires a range of integers. The -i option, forces shuf to operate as range shuf.

Range shuf produces a range of integers in random order.

#### **Usage**

Usage: shuf -i LO-HI [OPTION]...

#### **Ways to specify range**

#### **The one way: LO-HI**

**Commands**

{

shuf -i 1-3

}

**Output**

2

3

1

**Notes**

(1) The shuf command shuf -i 1-3 is equivalent to all previous command using the sequence 1 2 3

## **Advanced shuf options**

### **Limit number of output lines**

To limit the number of lines in the output, there is using of the -n option followed by an integer as follows.

**Commands**

{

shuf -i 1-3 -n 1

}

**Output**

3

**Notes**

* The shuf command shuf -i 1-3 -n 1 is equivalent to shuf -i 1-3 | head -1

### **Specify a file to write output lines**

To specify a file to write output lines, there is using of the -o option followed by a filename as follows.

**Commands**

{

shuf -i 1-3 -n 1 -o file;

cat file;

rm -f file

}

**Output**

1

**Notes**

(1) The shuf command shuf -i 1-3 -n 1 -o file is equivalent to shuf -i 1-3 -n 1 > file using I/O redirection

### **Stream output lines**

To create a continuous stream of output lines, there is using of the -r option as follows.

**Commands**

{

shuf -e {0,1} -r | xargs -i echo -n "{}"

}

**Output**

000101101010101101010110000101111010001010111001110…

### **Use the zero byte instead of newline as line delimiter**

To use zero-terminated lines, there is using of the -z option as follows.

**Commands**

{

seq 3 | tr '\n' '\0' | shuf -z

}

**Output**

213

**Notes**

(1) The output contains non-printing zero byte between digits

## **How to shuf in bash the easy way**

The easy way to shuf is to use the shuf command as discussed above.