BASH Programming

Bash Script Loops Examples

5 months ago • by Prateek Jangid

A loop in programming is a control structure that allows a specific code to be executed repeatedly until a condition is met. This process is repeated until no further action is required. Loop allows you to repeat the desired set of instructions numerous times to attain the desired outcome. These recursions can be useful for all tasks that require repetitive operations or when working with data collections.

In this article, we will explore the meaning, types, and examples of loops, further gaining insight into how to use them correctly, enabling you to develop powerful, concise, and adaptable code. As we mentioned earlier, three types of loops are available in bash scripting. Let's divide this section into multiple parts to explain each loop briefly.

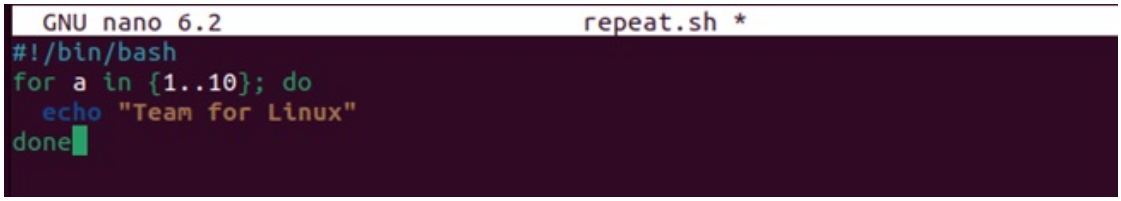
For Loop in Bash

For loop is used to execute specific commands or codes for particular times. Here is the basic syntax of the for loop:

```
for var in list
do
commands
done
```

For example, let's use the for loop to print a message for 10 times:

```
</div>
<div style="text-align: justify">!#/bin/bash
for i in {1..10}; do
echo "Hello, World!"
done
```



Once you execute the command, the system will print the message "Time for Linux" 10 times:

```
prateek@prateek:~$ ./repeat.sh
Team for Linux
Team for Linux
Team for Linux
Team for Linux
Team for Linux
Team for Linux
Team for Linux
Team for Linux
Team for Linux
Team for Linux
prateek@prateek:~$
```

While Loop

You can use the while loop to execute the commands if a specific condition is true.

```
while [ condition ]
do
commands
done
```

For instance, let's use the while loop, where the system will automatically exit whenever you press the S button on the keyboard:

```
#!/bin/bash
while :
do
read -n 1 k <&1
if [[ $k = S ]]
then
break
else
echo "You pressed '$k'"
fi
done
```

```
#!/bin/bash

while :
do
  read -n 1 k <&1
  if [[ $k = s ]]
  then
    break
  else
    echo "You Press '$k'"

  fi
done
```

In the above script, K is the variable used for all the keys except the S. Moreover, the loop will break the condition after the entry of the S key:

```
prateek@prateek:~$ ./press.sh
fYou Press 'f'
fYou Press 'f'
gYou Press 'g'
gYou Press 'g'
jYou Press 'j'
gYou Press 'g'
fYou Press 'f'
sprateek@prateek:~$
```

Until Loop

The until loop is similar to the while loop, but it only continues if the specific condition is false and terminates when the condition is true.

```
until [ condition ]
do
commands
done
```

For example, you can use the until loop to reach a specific condition, so let's create a script to print even numbers between o and 30.

```
#!/bin/bash
counter=0
until [ $counter -gt 30 ]; do
echo $counter
((counter+=2))
done
```

Once you run the script, the system will print all the even numbers until 30.

Wrapping Up

This was all about the simplest explanation behind bash script loops with examples. We have included multiple examples of various types of loops available in bash scripting. We recommend that you explore your creativity and skills to create amazing bash scripts around all the loops.

ABOUT THE AUTHOR

Prateek Jangid