
Linux™
SHELL
SCRIPTING

Loops

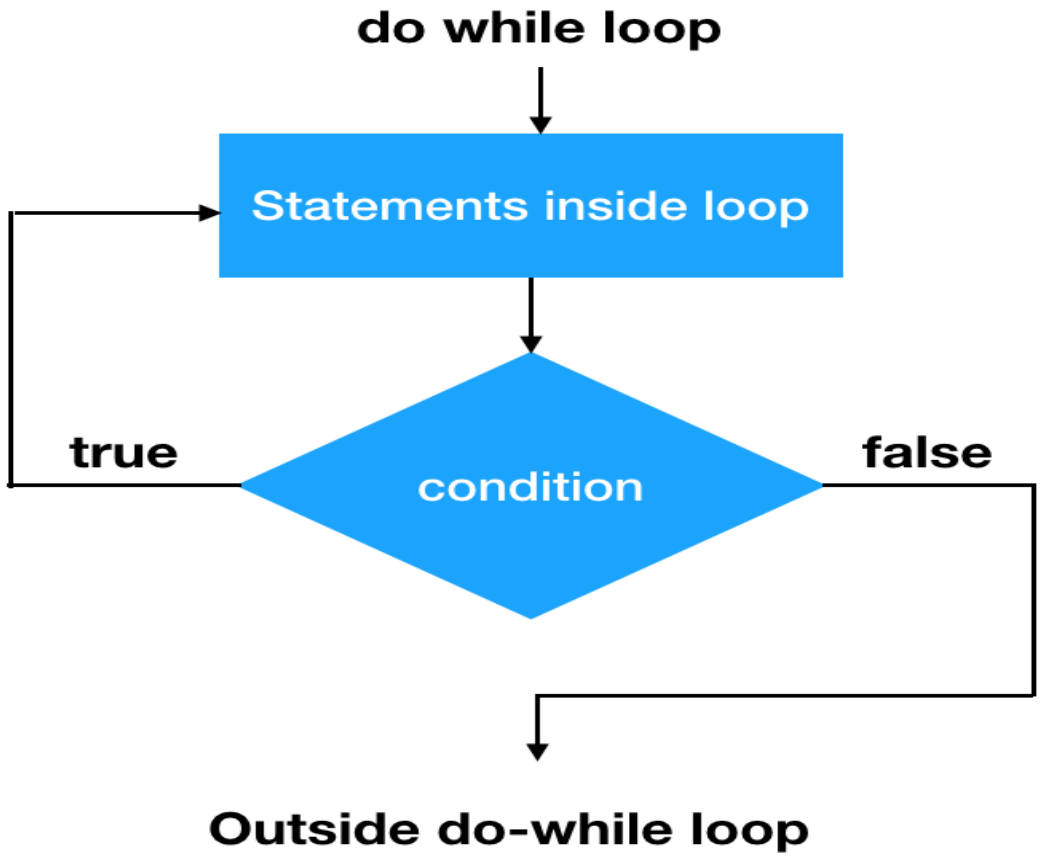
1. for statement
2. while statement
3. until statement
4. Select statement
5. functions

While loop

```
while [ condition ]
do
    command
done
```

While-loop-in-Shell-Scripting-5.png.webp

While loop



while loop

```
#!/bin/bash
 x=1
 while [ $x -le 5 ]
do
echo "Welcome $x times"
x=$(( $x + 1 ))
 done
```

until loop

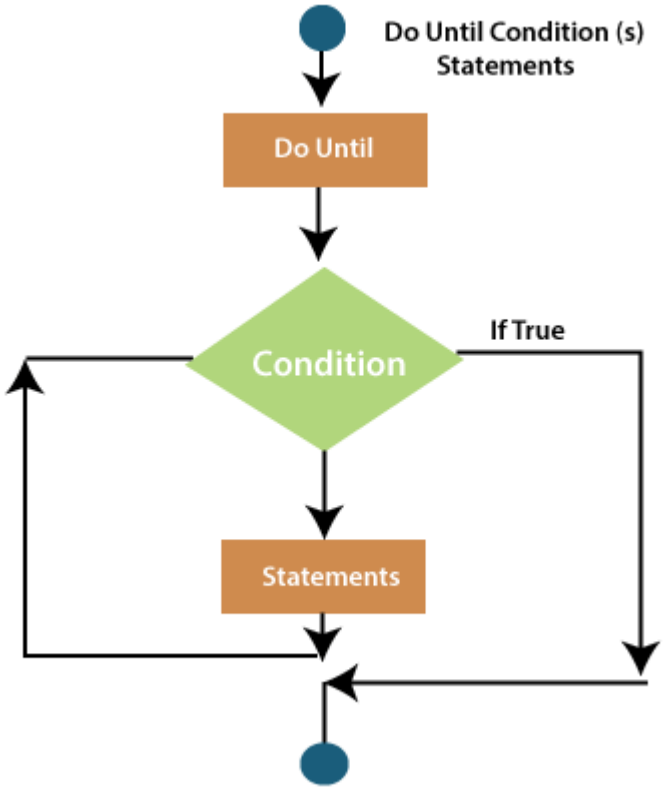
Until [condtion]

Do

commands

done

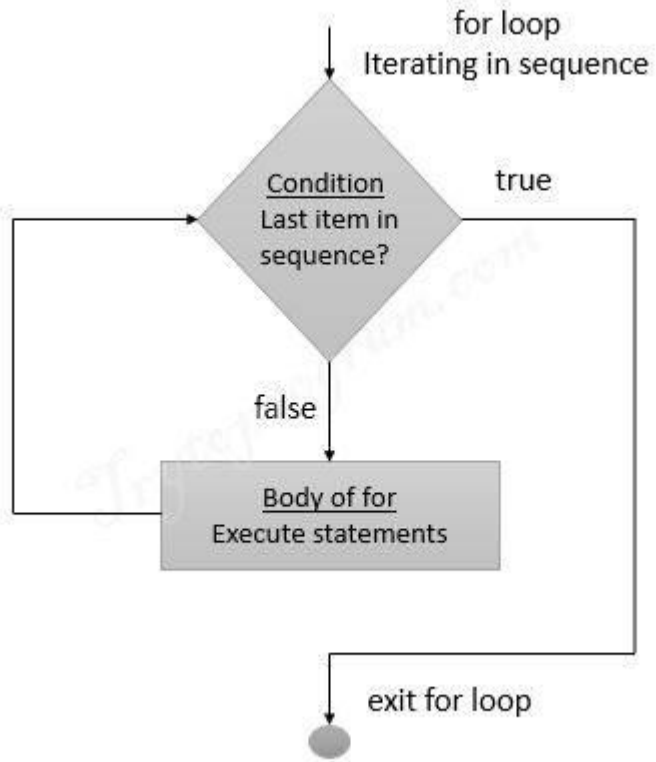
until loop


Do Until Condition (s)
Statements
Do Until
Condition
If True
Statements

until loop

```
#!/bin/bash
i=5
Until [ $i –gt 15 ]
Do
        echo “number $i”
        i=`expr $i + 1`
Done
```

for loop


for loop
Iterating in sequence
Condition
Last item in
sequence?
true
false
Body of for
Execute statements
exit for loop

for loop

for control variable

do

command

done

for loop

for control variable

do command
done

for loop

```
#!/bin/bash

for word in high on the hill was lovely mountain

do
        echo $word
done
```

which loop?

While loop: This looping process is a good choice when you are asking a question, whose answer will determine if the loop is repeated.

For loop: This loop is a good choice when the number of repetitions is known, or can be supplied by the user.

For vs While Loop

Comparison Chart

For Loop	While Loop
The for loop is used for definite loops when the number of iterations is known.	The while loop is used when the number of iterations is not known.
For loops can have their counter variables declared in the declaration itself.	There is no built-in loop control variable with a while loop.
This is preferable when we know exactly how many times the loop will be repeated.	The while loop will continue to run infinite number of times until the condition is met.
The loop iterates infinite number of times if the condition is not specified.	If the condition is not specified, it shows a compilation error.

break & continue

Interrupt for, while or until loop

The break statement

- transfer control to the statement AFTER the done statement
- terminate execution of the loop

The continue statement

- transfer control to the statement TO the done statement
- skip the test statements for the current iteration
- continues execution of the loop

break & continue

THE BREAK COMMAND

```
while [ condition ]
do
cmd-1
break
cmd-n
done
echo "done"
```

break & continue

THE CONTINUE COMMAND

```
while [ condition ]
do
cmd-1
continue
cmd-n
done
echo "done"
```

break and continue

```
for index in 1 2 3 4 5 6 7 8 9 10
do
if [ $index -le 3 ]; then
echo "continue"
continue
fi
echo $index
if [ $index -ge 8 ]; then
echo "break"
break
fi
done
```

select statement loop

- Constructs simple menu from word list
- Allows user to enter a number instead of a word
- User enters sequence number corresponding to the word

Syntax:

select WORD in LIST

do

RESPECTIVE-COMMANDS

done

functions

- Group of commands that are assigned to name
- They are like subroutines
- It helps reuse codes
- Efficient use of variables

functions

Synatx to use:

function <function name>

{

    set of commands

}