

# Лабораторная работа 5 #2

## Bash-скрипты, часть 2: циклы

### Циклы for

Оболочка bash поддерживает циклы for, которые позволяют организовывать перебор последовательностей значений. Вот какова базовая структура таких циклов:

```
for var in list
do
команды
done
```

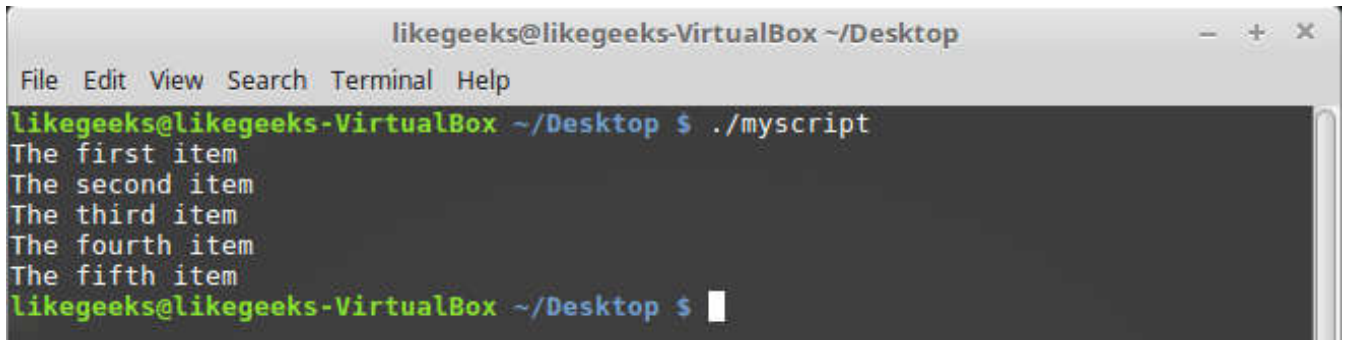
В каждой итерации цикла в переменную var будет записываться следующее значение из списка list. В первом проходе цикла, таким образом, будет задействовано первое значение из списка. Во втором — второе, и так далее — до тех пор, пока цикл не дойдёт до последнего элемента.

### Перебор простых значений

Пожалуй, самый простой пример цикла for в bash-скриптах — это перебор списка простых значений:

```
#!/bin/bash
for var in first second third fourth fifth
do
echo The $var item
done
```

Ниже показаны результаты работы этого скрипта. Хорошо видно, что в переменную `$var` последовательно попадают элементы из списка. Происходит так до тех пор, пока цикл не дойдёт до последнего из них.

A screenshot of a terminal window titled 'likegeeks@likegeeks-VirtualBox ~/Desktop'. The terminal shows the command `./myscript` being executed, which outputs five lines: 'The first item', 'The second item', 'The third item', 'The fourth item', and 'The fifth item'. The prompt `likegeeks@likegeeks-VirtualBox ~/Desktop $` is visible at the bottom.

```
likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
The first item
The second item
The third item
The fourth item
The fifth item
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

### *Простой цикл for*

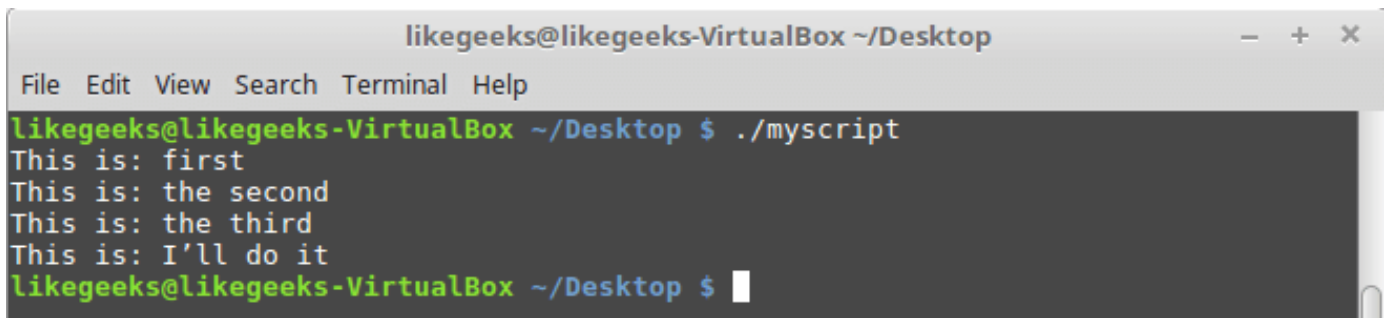
Обратите внимание на то, что переменная `$var` сохраняет значение при выходе из цикла, её содержимое можно менять, в целом, работать с ней можно как с любой другой переменной.

## Перебор сложных значений

В списке, использованном при инициализации цикла `for`, могут содержаться не только простые строки, состоящие из одного слова, но и целые фразы, в которые входят несколько слов и знаков препинания. Например, всё это может выглядеть так:

```
#!/bin/bash
for var in first "the second" "the third" "I'll do it"
do
echo "This is: $var"
done
```

Вот что получится после того, как этот цикл пройдёт по списку. Как видите, результат вполне ожидаем.



```
likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
This is: first
This is: the second
This is: the third
This is: I'll do it
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

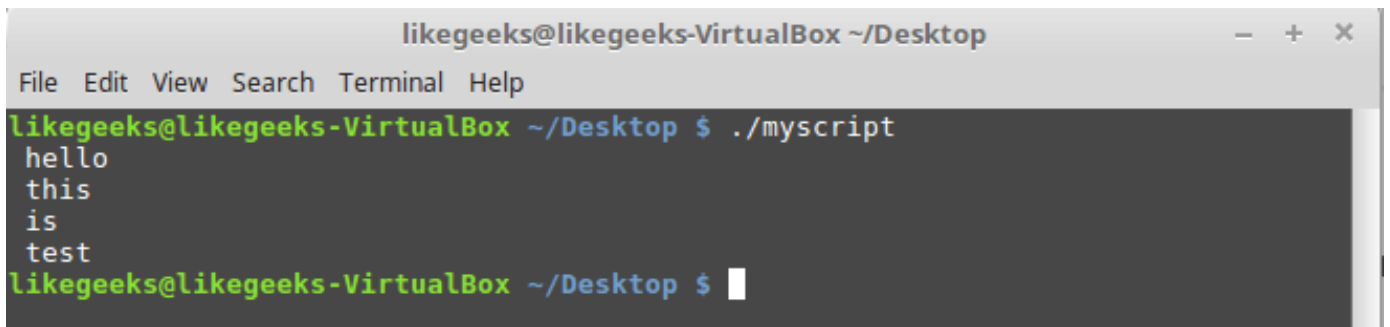
*Перебор сложных значений*

## Инициализация цикла списком, полученным из результатов работы команды

Ещё один способ инициализации цикла `for` заключается в передаче ему списка, который является результатом работы некоей команды. Тут используется подстановка команд для их исполнения и получения результатов их работы.

```
#!/bin/bash
file="myfile"
for var in $(cat $file)
do
echo " $var"
done
```

В этом примере задействована команда `cat`, которая читает содержимое файла. Полученный список значений передаётся в цикл и выводится на экран. Обратите внимание на то, что в файле, к которому мы обращаемся, содержится список слов, разделённых знаками перевода строки, пробелы при этом не используются.



```
likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
hello
this
is
test
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

*Цикл, который перебирает содержимое файла*

Тут надо учесть, что подобный подход, если ожидается построчная обработка данных, не сработает для файла более сложной структуры, в строках которого может содержаться по несколько слов, разделённых пробелами. Цикл будет обрабатывать отдельные слова, а не строки. Что, если это совсем не то, что нужно?

## Разделители полей

Причина вышеописанной особенности заключается в специальной переменной окружения, которая называется IFS (Internal Field Separator) и позволяет указывать разделители полей. По умолчанию оболочка bash считает разделителями полей следующие символы:

- Пробел
- Знак табуляции
- Знак перевода строки

Если bash встречает в данных любой из этих символов, он считает, что перед ним — следующее самостоятельное значение списка.

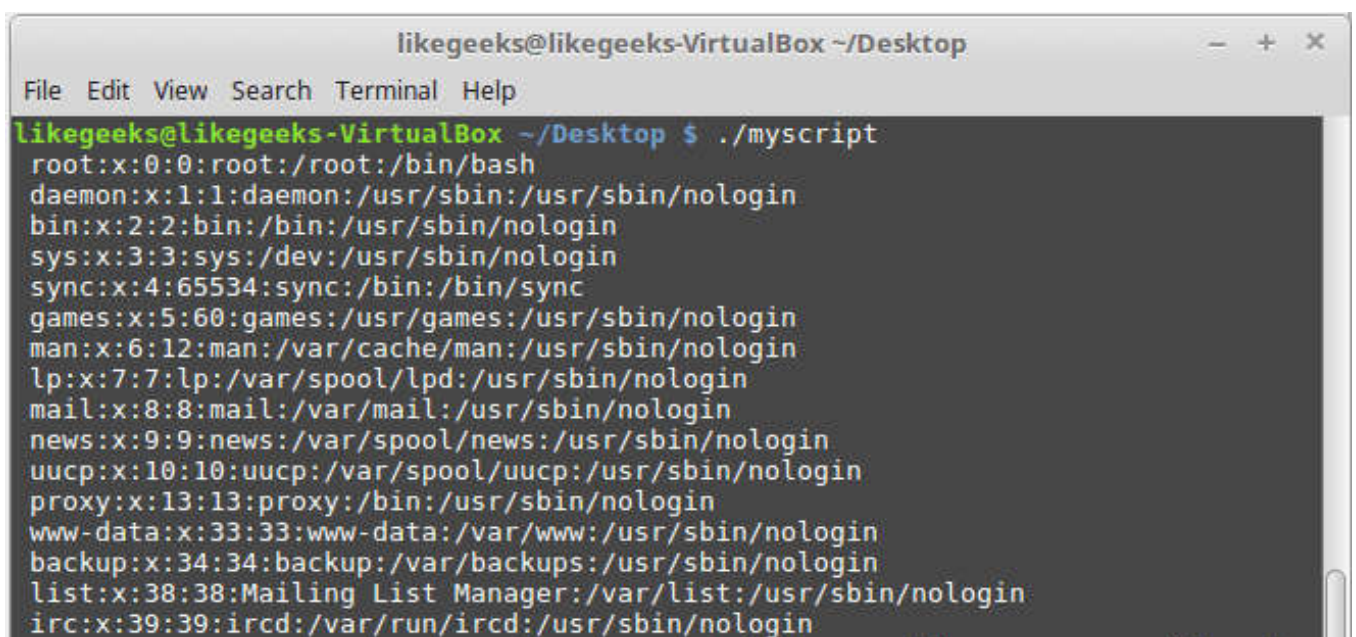
Для того чтобы решить проблему, можно временно изменить переменную среды IFS. Вот как это сделать в bash-скрипте, если исходить из предположения, что в качестве разделителя полей нужен только перевод строки:

```
IFS=$'\n'
```

После добавления этой команды в bash-скрипт, он будет работать как надо, игнорируя пробелы и знаки табуляции, считая разделителями полей лишь символы перевода строки.

```
#!/bin/bash
file="/etc/passwd"
IFS=$'\n'
for var in $(cat $file)
do
echo " $var"
done
```

Если этот скрипт запустить, он выведет именно то, что от него требуется, давая, в каждой итерации цикла, доступ к очередной строке, записанной в файл.

A screenshot of a terminal window titled 'likegeeks@likegeeks-VirtualBox ~/Desktop'. The terminal shows the execution of a script './myscript' which outputs the contents of the /etc/passwd file, one line at a time. The output lines are: root:x:0:0:root:/root:/bin/bash, daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin, bin:x:2:2:bin:/bin:/usr/sbin/nologin, sys:x:3:3:sys:/dev:/usr/sbin/nologin, sync:x:4:65534:sync:/bin:/bin/sync, games:x:5:60:games:/usr/games:/usr/sbin/nologin, man:x:6:12:man:/var/cache/man:/usr/sbin/nologin, lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin, mail:x:8:8:mail:/var/mail:/usr/sbin/nologin, news:x:9:9:news:/var/spool/news:/usr/sbin/nologin, uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin, proxy:x:13:13:proxy:/bin:/usr/sbin/nologin, www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin, backup:x:34:34:backup:/var/backups:/usr/sbin/nologin, list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin, and irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin.

```
likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
```

*Построчный обход содержимого файла в цикле for*

Разделителями могут быть и другие символы. Например, выше мы выводили на экран содержимое файла /etc/passwd. Данные о пользователях в строках

разделены с помощью двоеточий. Если в цикле нужно обрабатывать подобные строки, IFS можно настроить так:

```
IFS=:
```

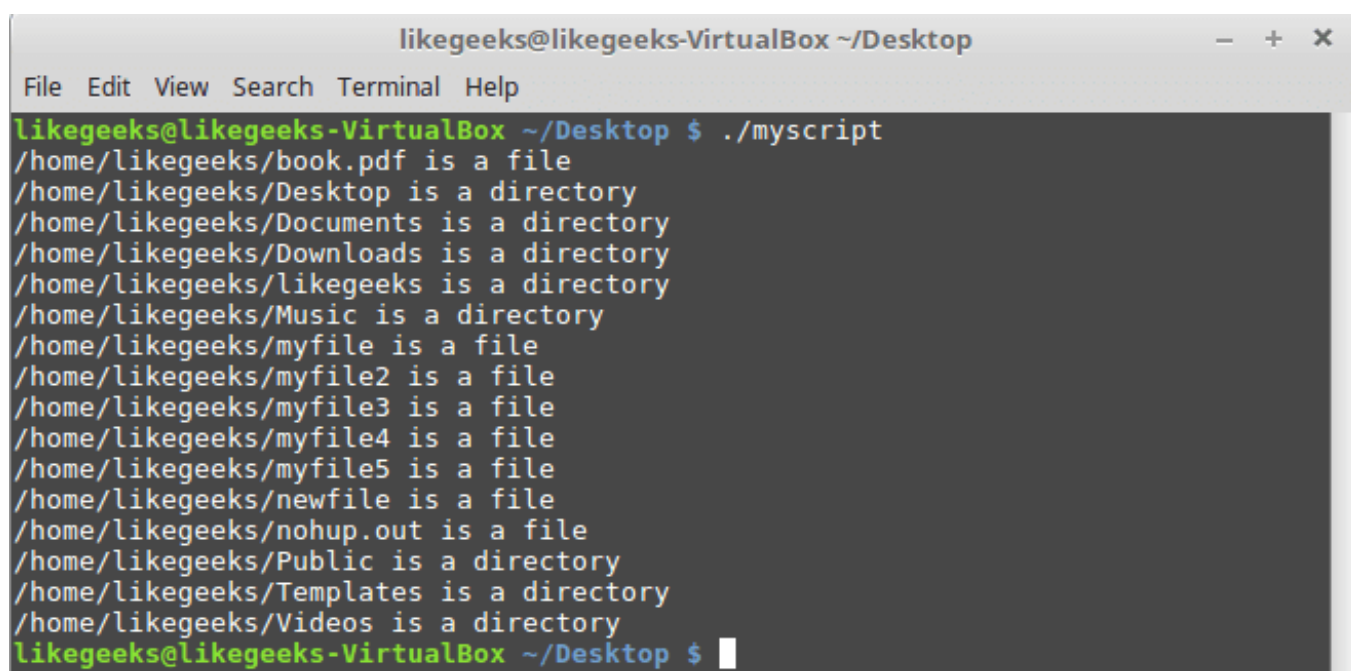
## Обход файлов, содержащихся в директории

Один из самых распространённых вариантов использования циклов `for` в `bash`-скриптах заключается в обходе файлов, находящихся в некоей директории, и в обработке этих файлов.

Например, вот как можно вывести список файлов и папок:

```
#!/bin/bash
for file in /home/likegeeks/*
do
if [ -d "$file" ]
then
echo "$file is a directory"
elif [ -f "$file" ]
then
echo "$file is a file"
fi
done
```

Вот что выведет скрипт:

A screenshot of a terminal window titled 'likegeeks@likegeeks-VirtualBox ~/Desktop'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows the command './myscript' being executed, which outputs a list of files and directories in the current directory, each followed by 'is a file' or 'is a directory'. The output is as follows:

```
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
/home/likegeeks/book.pdf is a file
/home/likegeeks/Desktop is a directory
/home/likegeeks/Documents is a directory
/home/likegeeks/Downloads is a directory
/home/likegeeks/likegeeks is a directory
/home/likegeeks/Music is a directory
/home/likegeeks/myfile is a file
/home/likegeeks/myfile2 is a file
/home/likegeeks/myfile3 is a file
/home/likegeeks/myfile4 is a file
/home/likegeeks/myfile5 is a file
/home/likegeeks/newfile is a file
/home/likegeeks/nohup.out is a file
/home/likegeeks/Public is a directory
/home/likegeeks/Templates is a directory
/home/likegeeks/Videos is a directory
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

*Вывод содержимого папки*

Обратите внимание на то, как мы инициализируем цикл, а именно — на подстановочный знак «\*» в конце адреса папки. Этот символ можно воспринимать как шаблон, означающий: «все файлы с любыми именами». Он позволяет организовать автоматическую подстановку имён файлов, которые и соответствуют шаблону.

При проверке условия в операторе `if`, мы заключаем имя переменной в кавычки. Сделано это, потому что имя файла или папки может содержать пробелы.

## Циклы `for` в стиле C

Если вы знакомы с языком программирования C, синтаксис описания `bash`-циклов `for` может показаться вам странным, так как привыкли вы, очевидно, к такому описанию циклов:

```
for (i = 0; i < 10; i++)  
{  
printf("number is %d\n", i);  
}
```

В bash-скриптах можно использовать циклы for, описание которых выглядит очень похожим на циклы в стиле C, правда, без некоторых отличий тут не обошлось. Схема цикла при подобном подходе выглядит так:

```
for (( начальное значение переменной ; условие окончания цикла;  
изменение переменной ))
```

На bash это можно написать так:

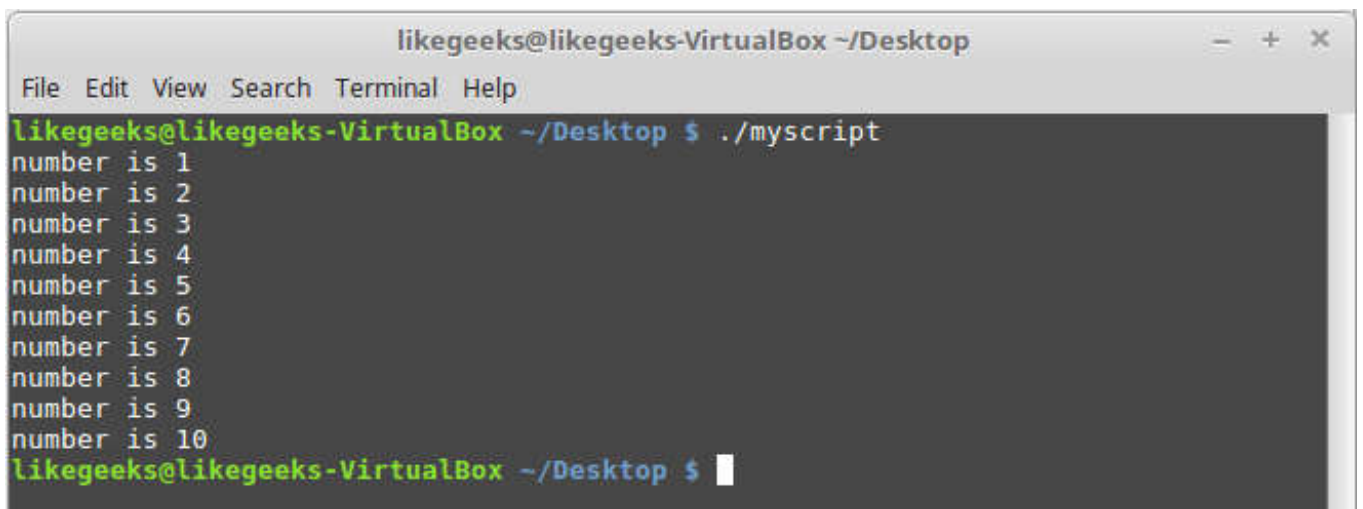
```
for (( a = 1; a < 10; a++ ))
```

А вот рабочий пример:

```
#!/bin/bash  
for (( i=1; i <= 10; i++ ))  
do  
echo "number is $i"  
done
```



Этот код выведет список чисел от 1 до 10.

A screenshot of a terminal window titled 'likegeeks@likegeeks-VirtualBox ~/Desktop'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows the command './myscript' being executed, which outputs a list of numbers from 1 to 10, each preceded by 'number is '. The prompt 'likegeeks@likegeeks-VirtualBox ~/Desktop \$' is visible at the bottom.

```
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
number is 1
number is 2
number is 3
number is 4
number is 5
number is 6
number is 7
number is 8
number is 9
number is 10
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

*Работа цикла в стиле C*

## Цикл while

Конструкция `for` — не единственный способ организации циклов в `bash`-скриптах. Здесь можно пользоваться и циклами `while`. В таком цикле можно задать команду проверки некоего условия и выполнять тело цикла до тех пор, пока проверяемое условие возвращает ноль, или сигнал успешного завершения некоей операции. Когда условие цикла вернёт ненулевое значение, что означает ошибку, цикл остановится.

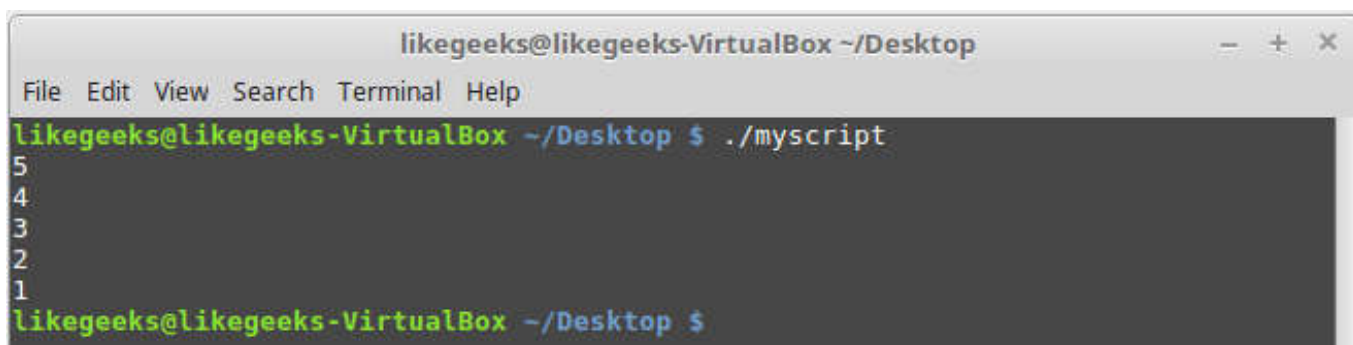
Вот схема организации циклов `while`:

```
while команда проверки условия
do
    другие команды
done
```

Взглянем на пример скрипта с таким циклом:

```
#!/bin/bash
var1=5
while [ $var1 -gt 0 ]
do
echo $var1
var1=$(( $var1 - 1 ))
done
```

На входе в цикл проверяется, больше ли нуля переменная \$var1. Если это так, выполняется тело цикла, в котором из значения переменной вычитается единица. Так происходит в каждой итерации, при этом мы выводим в консоль значение переменной до его модификации. Как только \$var1 примет значение 0, цикл прекращается.



```
likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
5
4
3
2
1
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

*Результат работы цикла while*

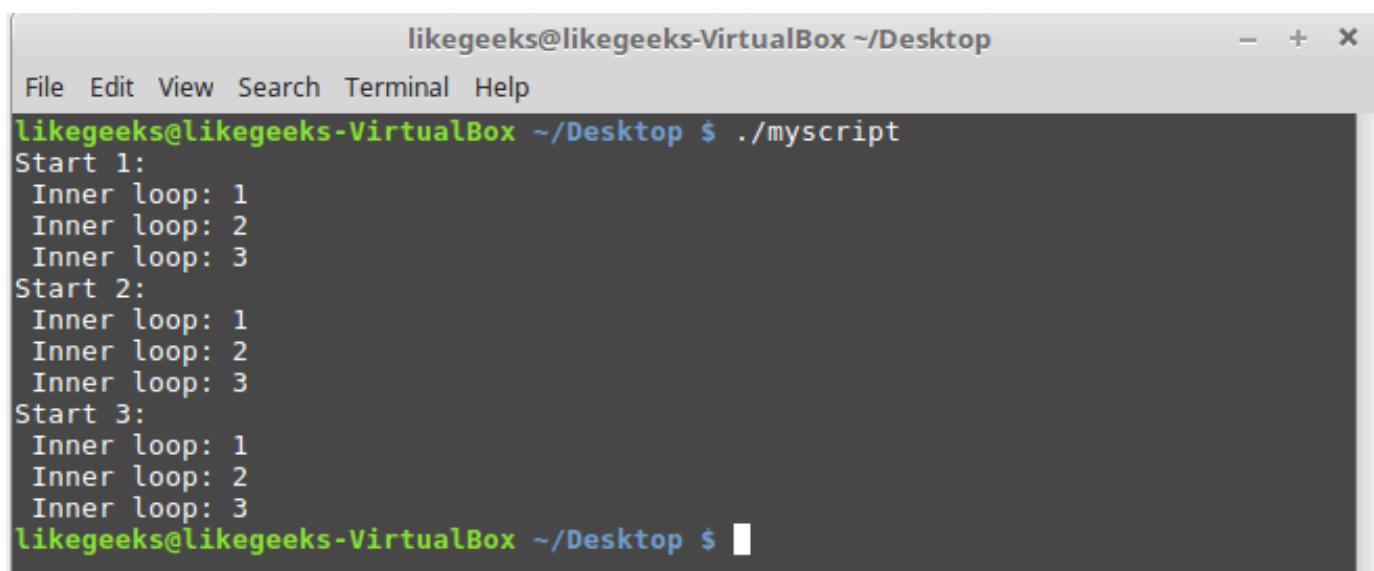
Если не модифицировать переменную \$var1, это приведёт к попаданию скрипта в бесконечный цикл.

## Вложенные циклы

В теле цикла можно использовать любые команды, в том числе — запускать другие циклы. Такие конструкции называют вложенными циклами:

```
#!/bin/bash
for (( a = 1; a <= 3; a++ ))
do
echo "Start $a:"
for (( b = 1; b <= 3; b++ ))
do
echo " Inner loop: $b"
done
done
```

Ниже показано то, что выведет этот скрипт. Как видно, сначала выполняется первая итерация внешнего цикла, потом — три итерации внутреннего, после его завершения снова в дело вступает внешний цикл, потом опять — внутренний.

A screenshot of a terminal window titled 'likegeeks@likegeeks-VirtualBox ~/Desktop'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows the execution of a script named 'myscript'. The output consists of three groups of lines, each starting with 'Start X:' where X is 1, 2, or 3. Each group is followed by three lines of 'Inner loop: Y' where Y is 1, 2, or 3. The prompt 'likegeeks@likegeeks-VirtualBox ~/Desktop \$' is visible at the bottom, followed by a cursor.

```
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
Start 1:
Inner loop: 1
Inner loop: 2
Inner loop: 3
Start 2:
Inner loop: 1
Inner loop: 2
Inner loop: 3
Start 3:
Inner loop: 1
Inner loop: 2
Inner loop: 3
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

*Вложенные циклы*

## Обработка содержимого файла

Чаще всего вложенные циклы используют для обработки файлов. Так, внешний цикл занимается перебором строк файла, а внутренний уже работает с каждой строкой. Вот, например, как выглядит обработка файла `/etc/passwd`:

```
#!/bin/bash
IFS=$'\n'
for entry in $(cat /etc/passwd)
do
echo "Values in $entry -"
IFS=:
for value in $entry
do
echo " $value"
done
done
```

В этом скрипте два цикла. Первый проходится по строкам, используя в качестве разделителя знак перевода строки. Внутренний занят разбором строк, поля которых разделены двоеточиями.

```
likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
Values in root:x:0:0:root:/root:/bin/bash -
root
x
0
0
root
/root
/bin/bash
Values in daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin -
daemon
x
1
1
daemon
/usr/sbin
/usr/sbin/nologin
Values in bin:x:2:2:bin:/bin:/usr/sbin/nologin -
bin
x
2
2
bin
/bin
```

### Обработка данных файла

Такой подход можно использовать при обработке файлов формата CSV, или любых подобных файлов, записывая, по мере надобности, в переменную окружения IFS символ-разделитель.

## Управление циклами

Возможно, после входа в цикл, нужно будет остановить его при достижении переменной цикла определённого значения, которое не соответствует изначально заданному условию окончания цикла.

Надо ли будет в такой ситуации дожидаться нормального завершения цикла? Нет, конечно, и в подобных случаях пригодятся следующие две команды:

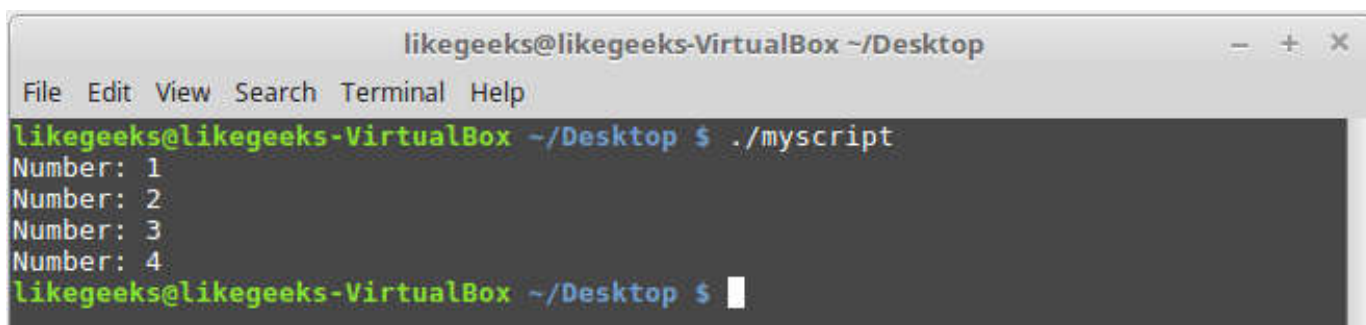
- break
- continue

## Команда break

Эта команда позволяет прервать выполнение цикла. Её можно использовать и для циклов for, и для циклов while:

```
#!/bin/bash
for var1 in 1 2 3 4 5 6 7 8 9 10
do
if [ $var1 -eq 5 ]
then
break
fi
echo "Number: $var1"
done
```

Такой цикл, в обычных условиях, пройдёт по всему списку значений из списка. Однако, в нашем случае, его выполнение будет прервано, когда переменная \$var1 будет равна 5.

A screenshot of a terminal window titled 'likegeeks@likegeeks-VirtualBox ~/Desktop'. The terminal shows the execution of a script named 'myscript'. The output of the script is 'Number: 1', 'Number: 2', 'Number: 3', and 'Number: 4'. The prompt 'likegeeks@likegeeks-VirtualBox ~/Desktop \$' is visible at the bottom of the terminal, indicating that the script has finished execution and the user is back at the shell prompt.

```
likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
Number: 1
Number: 2
Number: 3
Number: 4
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

*Досрочный выход из цикла for*

Вот — то же самое, но уже для цикла while:

```
#!/bin/bash
var1=1
while [ $var1 -lt 10 ]
do
```

```
if [ $var1 -eq 5 ]
then
break
fi
echo "Iteration: $var1"
var1=$(( $var1 + 1 ))
done
```

Команда `break`, исполненная, когда значение `$var1` станет равно 5, прерывает цикл. В консоль выведется то же самое, что и в предыдущем примере.

## Команда `continue`

Когда в теле цикла встречается эта команда, текущая итерация завершается досрочно и начинается следующая, при этом выхода из цикла не происходит. Посмотрим на команду `continue` в цикле `for`:

```
#!/bin/bash
for (( var1 = 1; var1 < 15; var1++ ))
do
if [ $var1 -gt 5 ] && [ $var1 -lt 10 ]
then
continue
fi
echo "Iteration number: $var1"
done
```

Когда условие внутри цикла выполняется, то есть, когда `$var1` больше 5 и меньше 10, оболочка исполняет команду `continue`. Это приводит к пропуску оставшихся в теле цикла команд и переходу к следующей итерации.

```
likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
Iteration number: 1
Iteration number: 2
Iteration number: 3
Iteration number: 4
Iteration number: 5
Iteration number: 10
Iteration number: 11
Iteration number: 12
Iteration number: 13
Iteration number: 14
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

*Команда continue в цикле for*

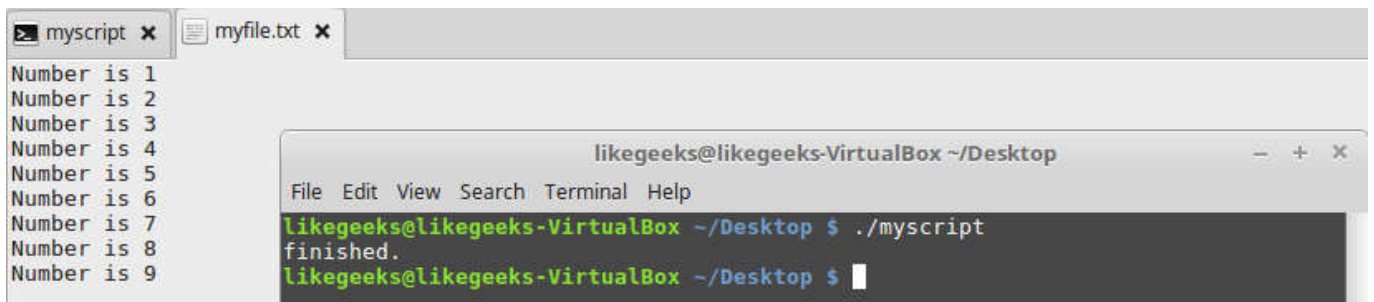
## Обработка вывода, выполняемого в цикле

Данные, выводимые в цикле, можно обработать, либо перенаправив вывод, либо передав их в конвейер. Делается это с помощью добавления команд обработки вывода после инструкции done. Например, вместо того, чтобы показывать на экране то, что выводится в цикле, можно записать всё это в файл или передать ещё куда-нибудь:

```
#!/bin/bash
for (( a = 1; a < 10; a++ ))
do
echo "Number is $a"
done > myfile.txt
echo "finished."
```

Оболочка создаст файл myfile.txt и перенаправит в этот файл вывод конструкции for. Откроем файл и удостоверимся в том, что он содержит именно то, что ожидается.



The screenshot shows a terminal window with two tabs: 'myscript' and 'myfile.txt'. The 'myscript' tab is active and displays the output of a script: 'Number is 1' through 'Number is 9'. The 'myfile.txt' tab is also visible but empty. Below the script output, there is a separate terminal window titled 'likegeeks@likegeeks-VirtualBox ~/Desktop'. This window shows the command './myscript' being executed, followed by the output 'finished.' and a prompt for the next command.

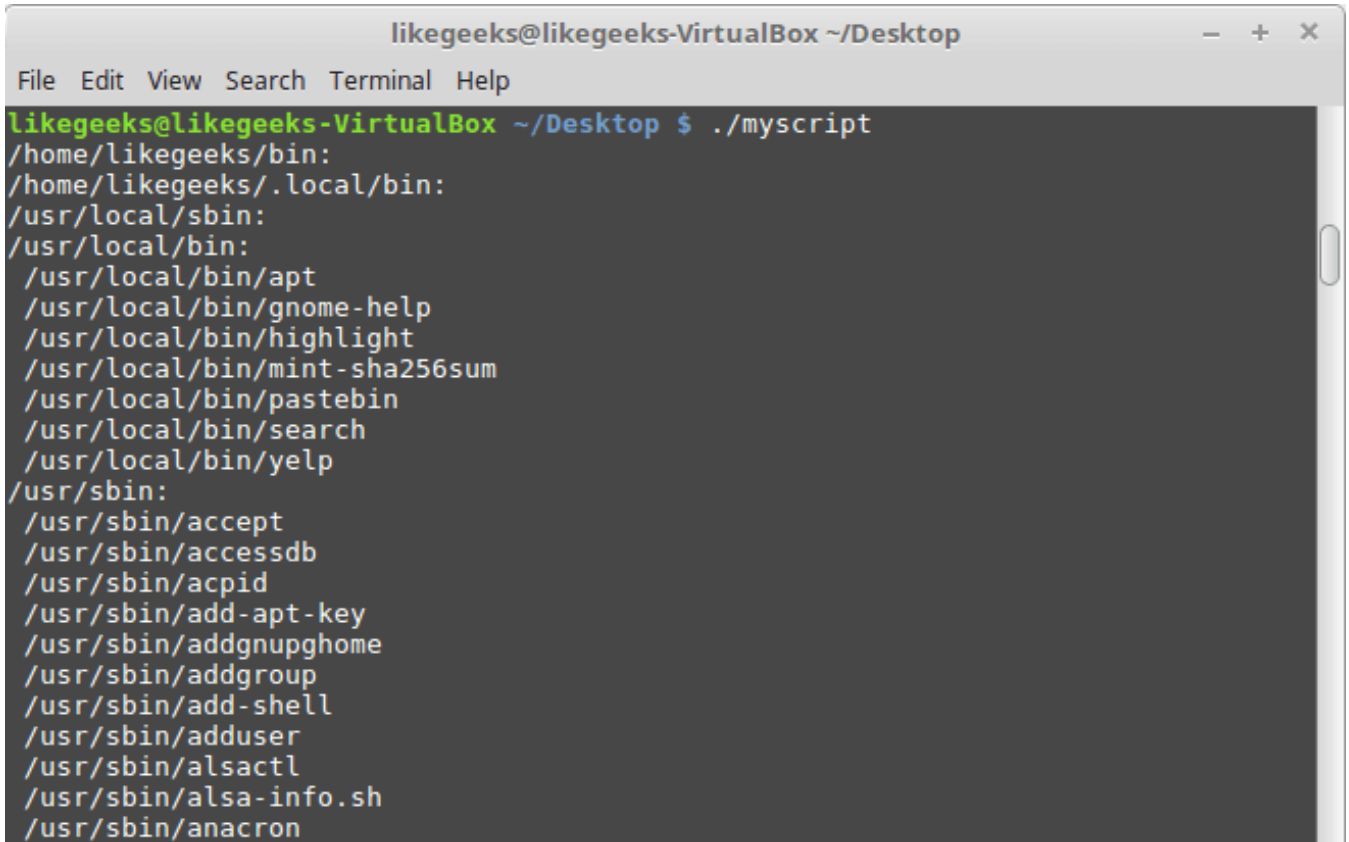
*Перенаправление вывода цикла в файл*

## Пример: поиск исполняемых файлов

Давайте воспользуемся тем, что мы уже разобрали, и напомним что-нибудь полезное. Например, если надо выяснить, какие именно исполняемые файлы доступны в системе, можно просканировать все папки, записанные в переменную окружения PATH. Весь арсенал средств, который для этого нужен, у нас уже есть, надо лишь собрать всё это воедино:

```
#!/bin/bash
IFS=:
for folder in $PATH
do
echo "$folder:"
for file in $folder/*
do
if [ -x $file ]
then
echo " $file"
fi
done
done
```

Такой вот скрипт, небольшой и несложный, позволил получить список исполняемых файлов, хранящихся в папках из PATH.

A screenshot of a terminal window titled 'likegeeks@likegeeks-VirtualBox ~/Desktop'. The terminal shows the command './myscript' being executed, which outputs a list of directories and executables from the PATH environment variable. The output is as follows:

```
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
/home/likegeeks/bin:
/home/likegeeks/.local/bin:
/usr/local/sbin:
/usr/local/bin:
/usr/local/bin/apt
/usr/local/bin/gnome-help
/usr/local/bin/highlight
/usr/local/bin/mint-sha256sum
/usr/local/bin/pastebin
/usr/local/bin/search
/usr/local/bin/yelp
/usr/sbin:
/usr/sbin/accept
/usr/sbin/accessdb
/usr/sbin/acpid
/usr/sbin/add-apt-key
/usr/sbin/addgnupghome
/usr/sbin/addgroup
/usr/sbin/add-shell
/usr/sbin/adduser
/usr/sbin/alsactl
/usr/sbin/alsa-info.sh
/usr/sbin/anacron
```

*Поиск исполняемых файлов в папках из переменной PATH*