Исполняемые файлы (bash)

Сценарии командной строки — это наборы тех же самых команд, которые можно вводить с клавиатуры, собранные в файлы и объединённые некоей общей целью. При этом результаты работы команд могут представлять либо самостоятельную ценность, либо служить входными данными для других команд.

Сценарии — это мощный способ автоматизации часто выполняемых действий.

Как устроены bash-скрипты

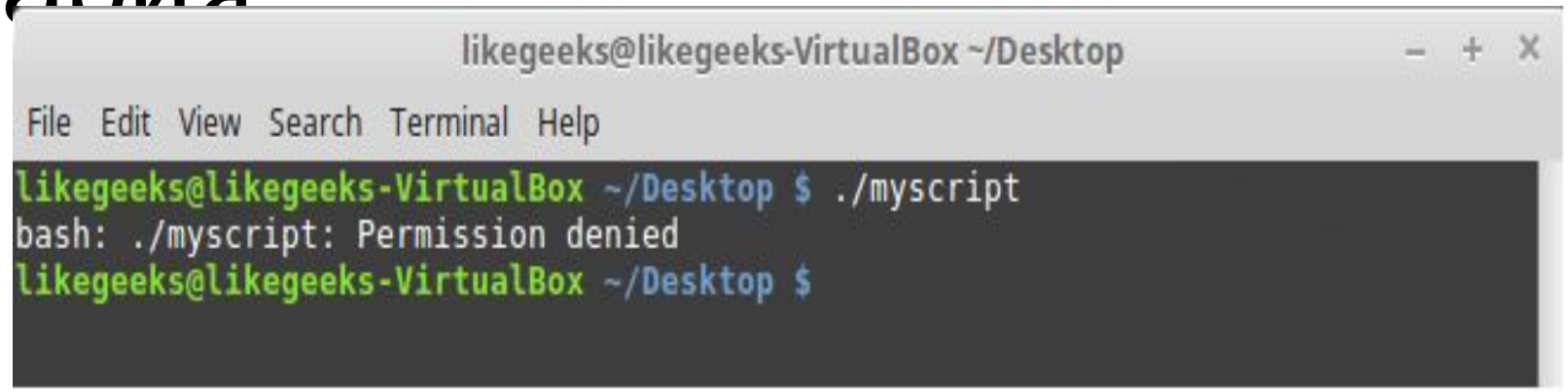
Создайте пустой файл с использованием команды touch.

#!/bin/bash

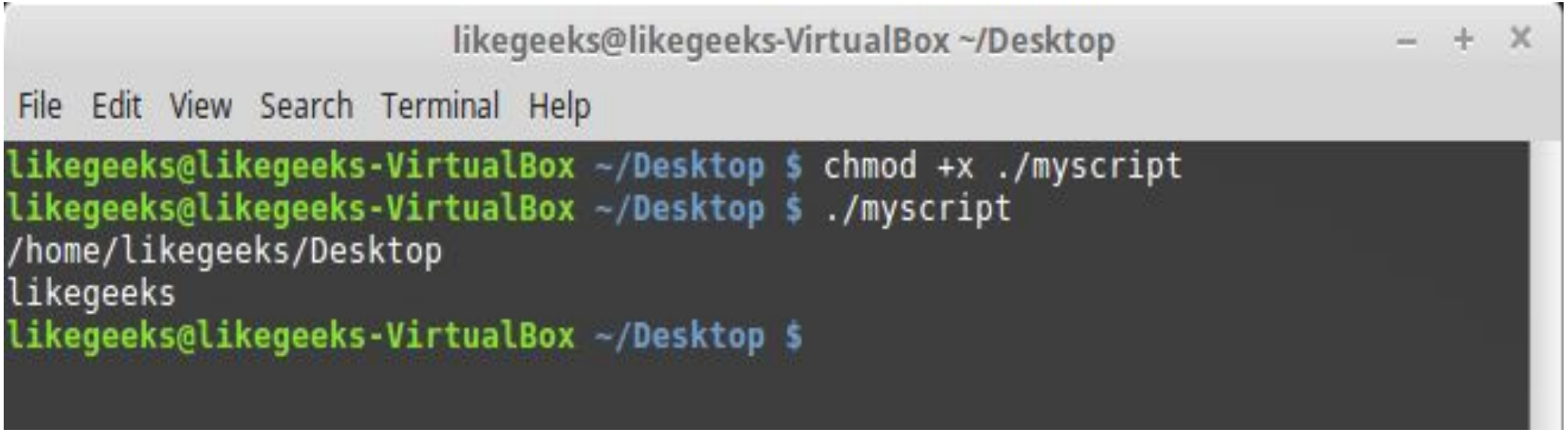
В других строках этого файла символ решётки используется для обозначения комментариев, которые оболочка не обрабатывает.

```
#!/bin/bash
# This is a comment
pwd
Whoami
```

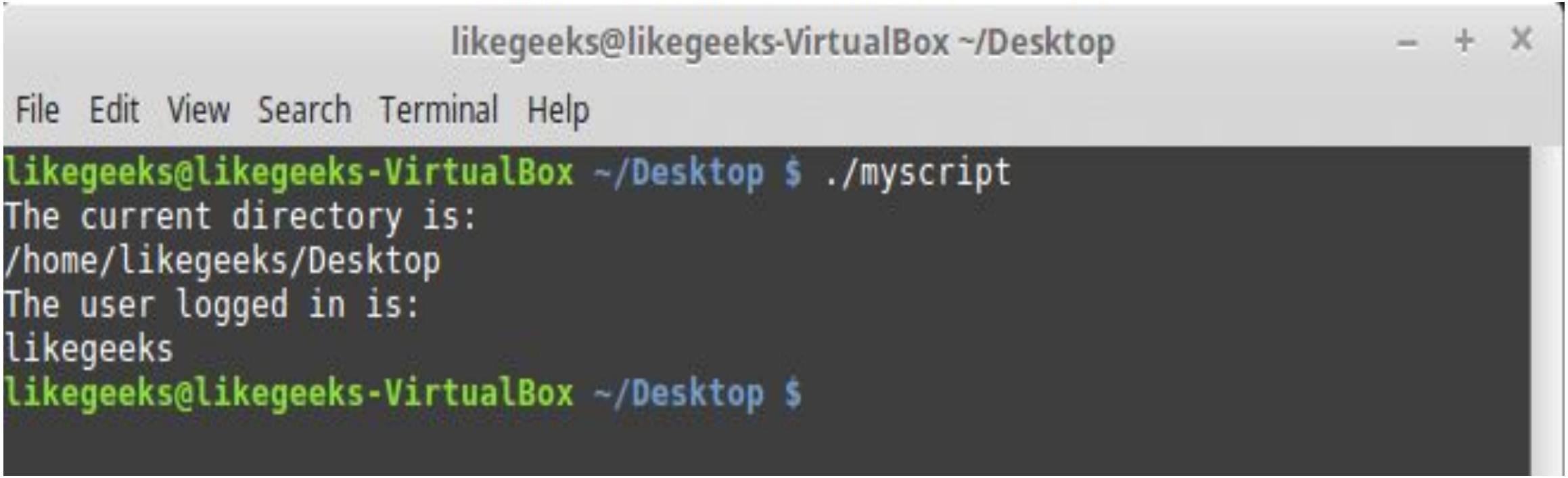
Установка разрешений для файла сценария



Попытка запуска файла сценария с неправильно настроенными разрешениями



```
#!/bin/bash
# our comment is here
echo "The current directory is:"
pwd
echo "The user logged in is:"
whoami
```



Использование переменных

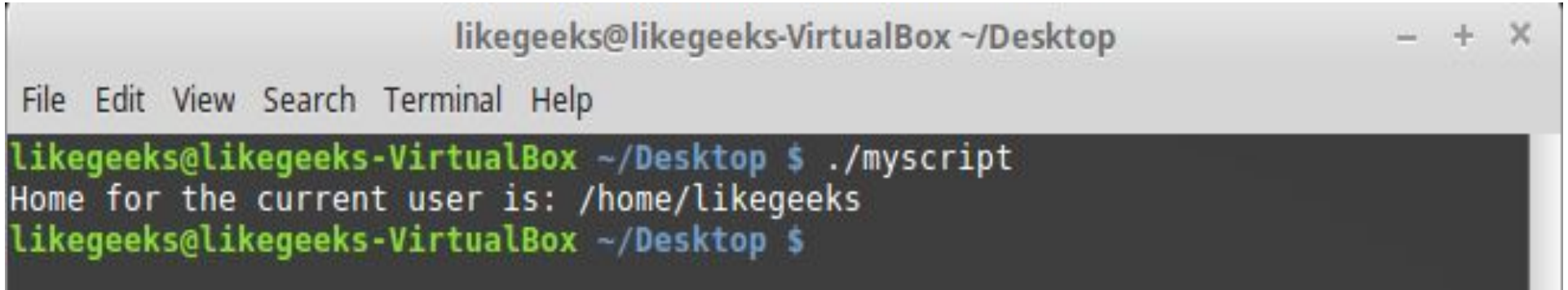
Переменные позволяют хранить в файле сценария информацию, например — результаты работы команд для использования их другими командами.

Существуют два типа переменных, которые можно использовать в bash-скриптах:

— Переменные среды

— Пользовательские переменные

```
#!/bin/bash
# display user home
echo "Home for the current user is: $HOME"
```



Зарезервированные переменные:

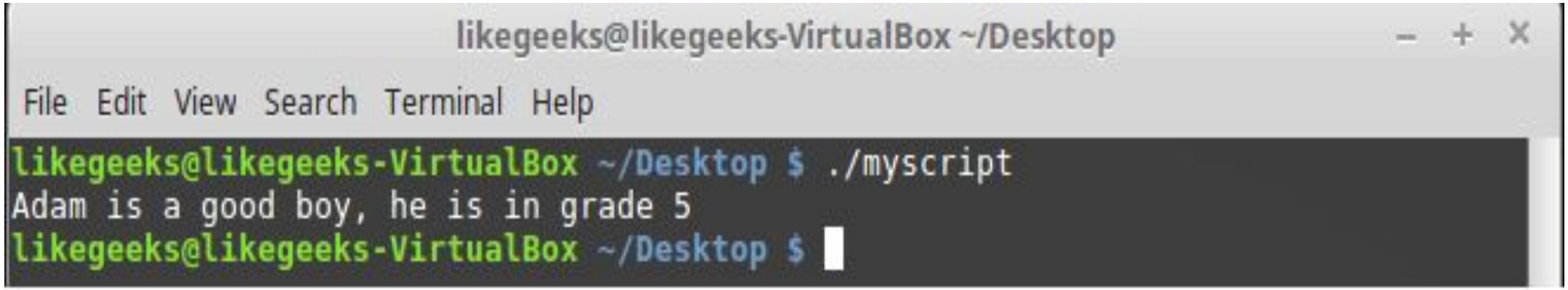
$EDITOR - текстовый редактор по умолчанию
$EUID - Эффективный UID. Если вы использовали программу su для выполнения команд от другого пользователя, то эта переменная содержит UID этого пользователя, в то время как...
$UID - ...содержит реальный идентификатор, который устанавливается только при логине.
$GROUPS - массив групп к которым принадлежит текущий пользователь
$HOME - домашний каталог пользователя
$HOSTNAME - ваш hostname
$HOSTTYPE - архитектура машины.
$LC_CTYPE - внутренняя переменная, котороя определяет кодировку символов
$OLDPWD - прежний рабочий каталог
$OSTYPE - тип ОС
$PATH - путь поиска программ
$SECONDS - время работы скрипта(в сек.)
$# - общее количество параметров переданных скрипту
$* - все аргументы переданыне скрипту(выводятся в строку)
$@ - тоже самое, что и предыдущий, но параметры выводятся в столбик
$! - PID последнего запущенного в фоне процесса
$$ - PID самого скрипта

Пользовательские переменные

Bash-скрипты позволяют задавать и использовать в сценарии собственные переменные.

Подобные переменные хранят значение до тех пор, пока не завершится выполнение сценария.

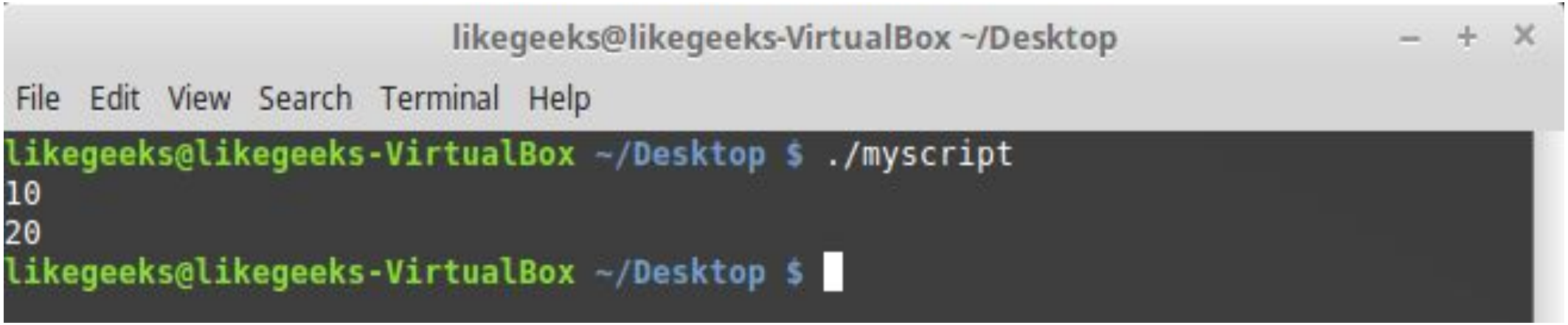
```
#!/bin/bash
# testing variables
grade=5
person="Adam"
echo "$person is a good boy, he is in grade $grade"
```



Математические операции

Для выполнения математических операций в файле скрипта можно использовать конструкцию вида $((a+b)):

```
#!/bin/bash
var1=$(( 5 + 5 ))
echo $var1
var2=$(( $var1 * 2 ))
echo $var2
```



Управляющая конструкция if-then

Управляющая конструкция if-then. В наиболее простом виде она выглядит так:

```
if команда
then
команды
fi
```

А вот рабочий пример:

```
#!/bin/bash
    if pwd
    then
    echo "It works"
    fi
```

Управляющая конструкция if-then-else

Для того, чтобы программа смогла сообщить и о результатах успешного поиска, и о неудаче, воспользуемся конструкцией if-then-else. Вот как она устроена:

```
if команда
then
команды
else
команды
fi
```

Сравнение чисел

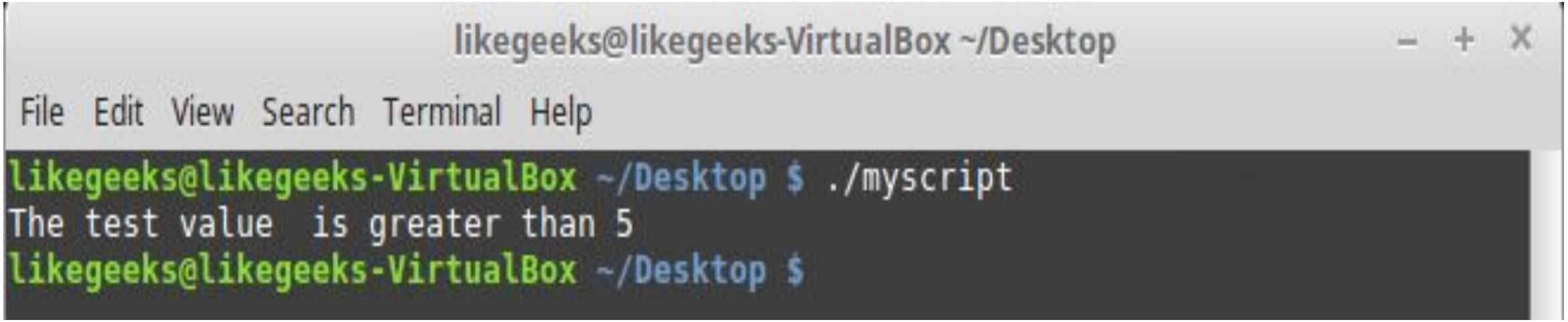
Логические операторы:

```
-z          #строка пуста
-n          #строка не пуста
!=          #строки неравны
-eq         #равно
-ne         #неравно
-lt,(< )    #меньше
-le,(<=)    #меньше или равно
-gt,(>)     #больше
-ge,(>=)    #больше или равно
!           #отрицание логического выражения
-a,(&&)     #логическое «И»
-o,(||)     #логическое «ИЛИ»
```

Задание 1:

Ввести числовую переменную.

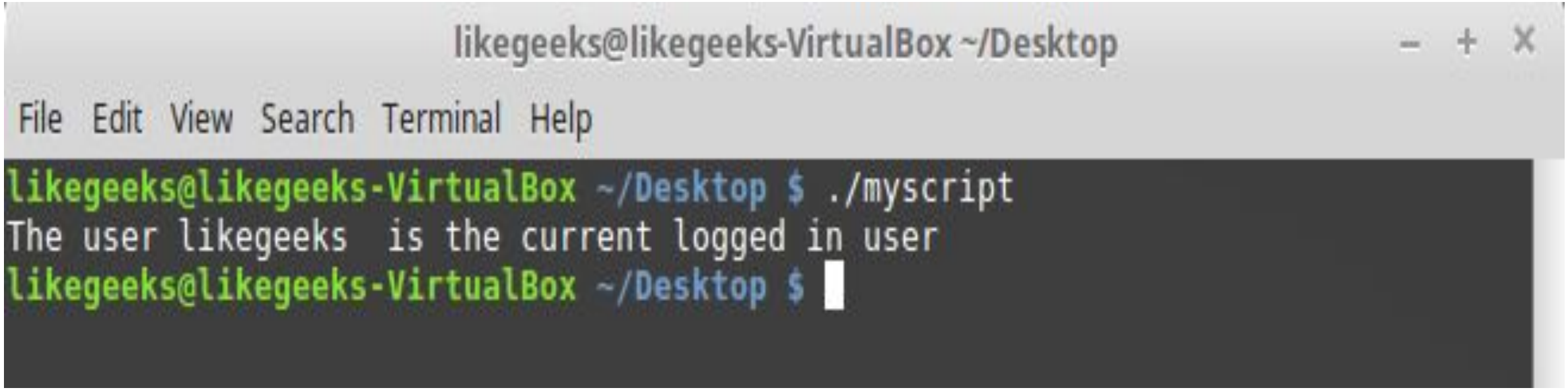
Если переменная больше 5 то вывести на экран «The test value <переменная> is greater than 5» иначе «The test value <переменная> is not greater than 5»



Сравнение строк

- str1 = str2 Проверяет строки на равенство, возвращает истину, если строки идентичны;
- str1 != str2Возвращает истину, если строки не идентичны;
- str1 < str2Возвращает истину, если str1меньше, чем str2;
- str1 > str2 Возвращает истину, если str1больше, чем str2;
- -n str1 Возвращает истину, если длина str1больше нуля;
- -z str1Возвращает истину, если длина str1равна нулю.

```
#!/bin/bash
user ="likegeeks"
if [$user = $USER]
then
echo "The user $user  is the current logged in user"
fi
```



Циклы for

Базовая структура таких циклов:

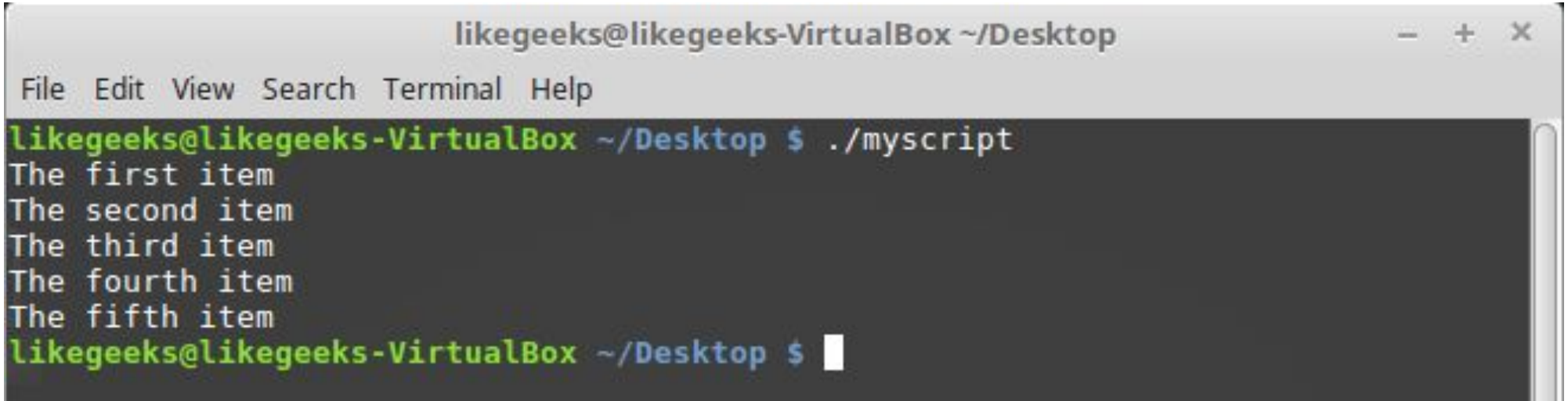
```
for var in list
    do
        команды
    Done //дан
```

В каждой итерации цикла в переменную var будет записываться следующее значение из списка list

Перебор простых значений

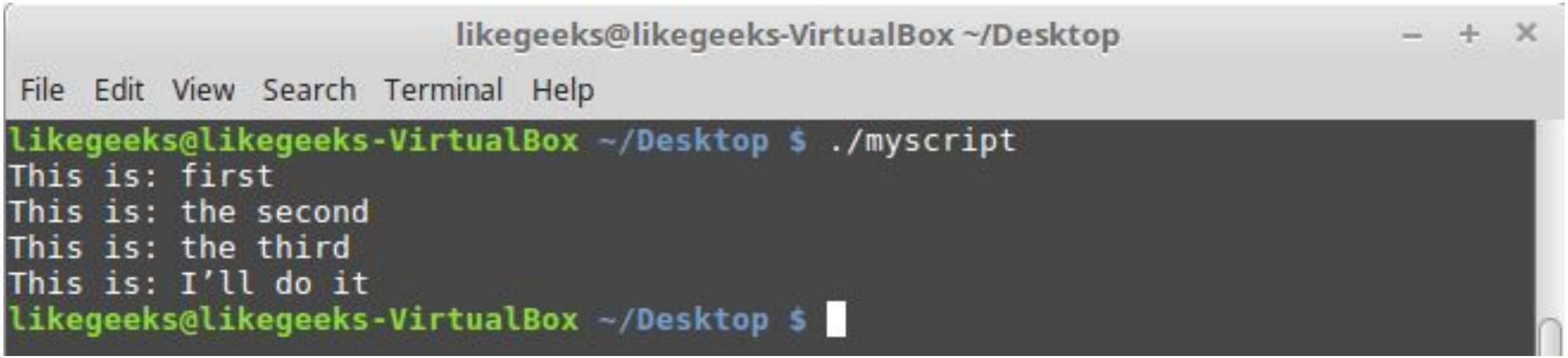
Перебор списка простых значений

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



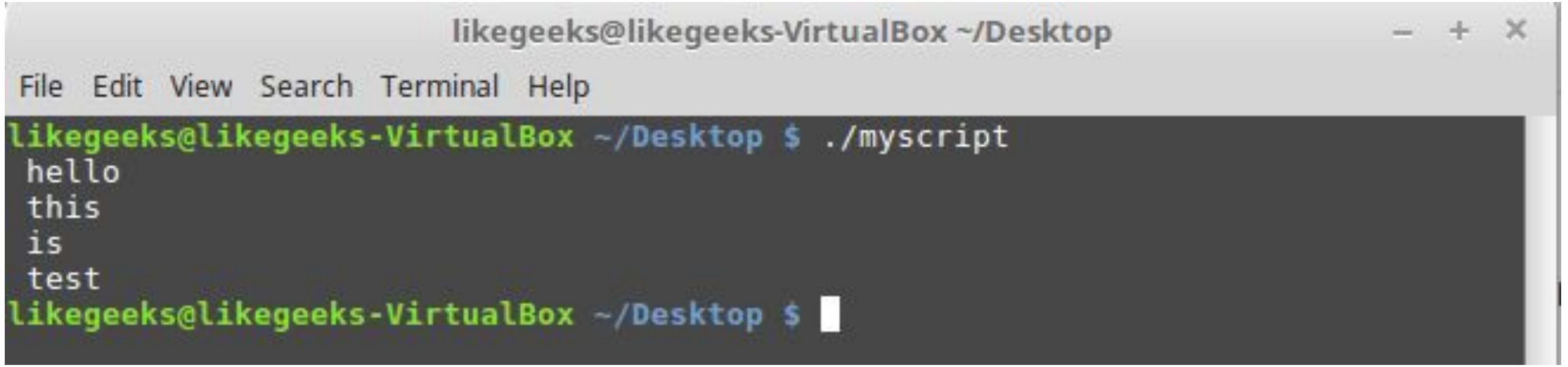
Перебор сложных

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



Инициализация цикла списком, полученным из результатов работы команды

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



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

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

Циклы for в стиле C

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

Схема цикла выглядит так:

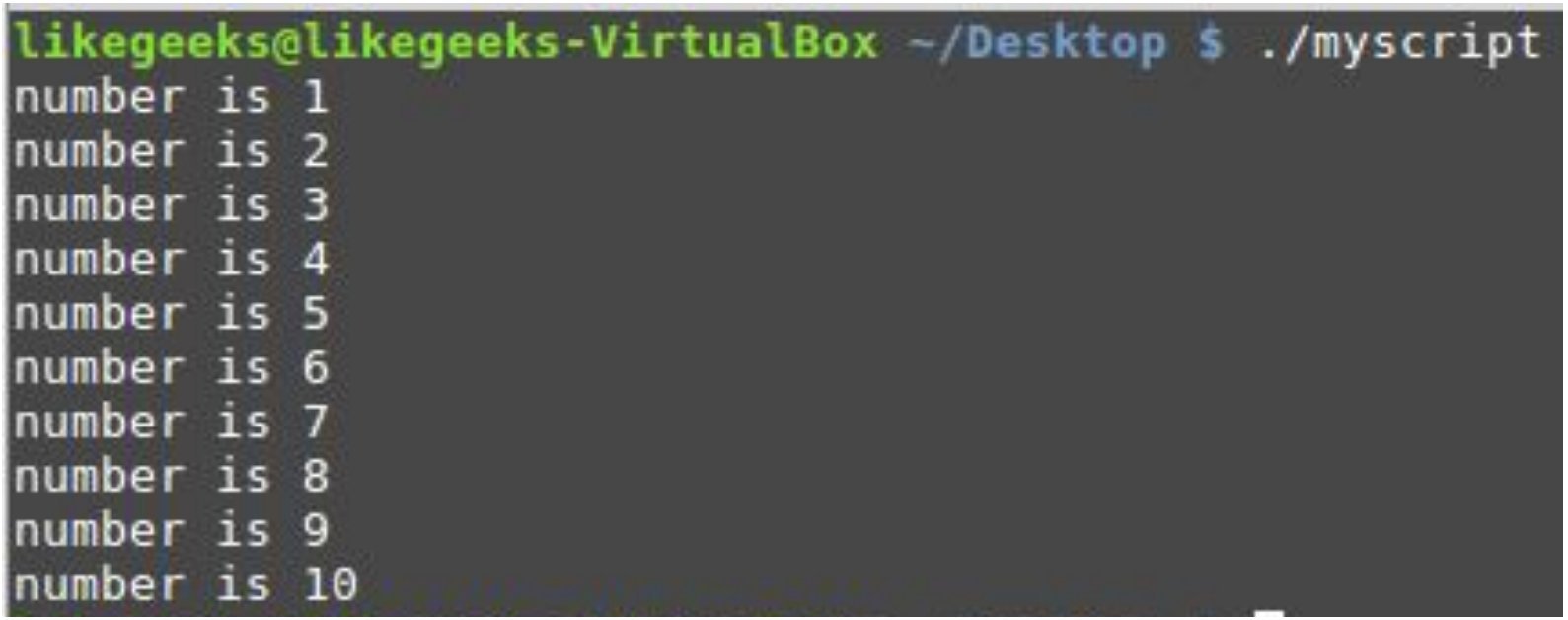
for ((начальное значение переменной ; условие окончания цикла; изменение переменной))

На bash это можно записать так:

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

Пример:

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



Цикл while

Вот схема организации циклов while

while команда проверки условия

do

другие команды

done

ПРИМЕР:

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