Программирование в bash

Использование переменных и циклов в командной строке. Запуск программ по расписанию (cron).

**Оглавление**

[Циклы](#_tyjcwt)

[Цикл for in](#_1x6wdhiavb2b)

[Цикл while](#_vf0hunoxtswp)

[Цикл until](#_e83lnuitdxgg)

[Настройка планировщика Cron в Linux](#_nh8nm6noxqij)

[Практическое задание](#_111kx3o)

[Дополнительные материалы](#_odecvkkwb3hz)

[Используемая литература](#_206ipza)

# 

# Циклы

Циклы используются при написании большинства программ или скриптов. При помощи циклов мы можем выполнять один и тот же код несколько раз. В командной оболочке **bash** существует несколько типов циклов. Рассмотрим циклы **for in**, **for**, **while**, **until**. Циклы **for in** и **for** считаются разными правилами одного оператора, они отличаются больше, чем **while** от **until**.

Иногда необходимо много раз выполнить одну команду из командной оболочки в Linux.

В **bash** есть простая конструкция, с помощью которой можно написать команду один раз, но задать условие, чтобы она была выполнена N раз.

Ниже покажем, что один из лучших методов, чтобы выполнить одну команду N раз подряд, — это использование в **bash** цикла **FOR**.

## Цикл for in

Это одна из основных разновидностей циклов, и она значительно отличается от аналога в языке C.

Ниже приведем примеры выполнения циклов **FOR**.

|  |
| --- |
| ~ $ for char in {1..10}; do <Список команд>; done ~ $ for n in {1..4}; do echo "Linux является одной из самых безопасных операционных систем"; done Linux является одной из самых безопасных операционных систем Linux является одной из самых безопасных операционных систем Linux является одной из самых безопасных операционных систем Linux является одной из самых безопасных операционных систем |

Если вам необходимо часто использовать такую конструкцию, тогда не помешает создать соответствующую bash-функцию (добавьте в **~/.bashrc**):

|  |
| --- |
| function run() {  number=$1  shift  for n in $(seq $number); do  $@  done } |

Использование:

|  |
| --- |
| $ run 5 <КОМАНДА> |

Пример:

|  |
| --- |
| $ run 5 echo "Я умею использовать операционную систему Windows Server Standart 2012 R2" Я умею использовать операционную систему Windows Server Standart 2012 R2 Я умею использовать операционную систему Windows Server Standart 2012 R2 Я умею использовать операционную систему Windows Server Standart 2012 R2 Я умею использовать операционную систему Windows Server Standart 2012 R2 Я умею использовать операционную систему Windows Server Standart 2012 R2 |

## Цикл while

Это достаточно простая конструкция, которая проверяет условие, стоящее за оператором **while**. В случае его истинности выполняется блок команд, находящийся между словами **do** и **done**, и затем программа опять переходит к проверке условия. Если проверка вернет ложь, то цикл заканчивается и начинают выполняться команды, следующие за **done**. Нужно обязательно следить за тем, чтобы проверка условия зависела от кода, выполняющегося в цикле. Иначе, если результат проверки не изменяется, вы получите бесконечный цикл.

Стандартное устройство ввода для цикла **while** можно перенаправить на файл с помощью команды перенаправления **<** в конце цикла.

Синтаксис цикла **while**:

|  |
| --- |
| while команды1 do команды2 done |

Цикл **while** будет в работе, пока условие, заданное в списке команды 1, будет верным. Поэтому цикл **while** также называют циклом с верным условием. Приведем ниже пример:

|  |
| --- |
| y=1 while [$y -It 10] do  echo $y  Y = $(( $Y + 1 )) done |

В командной оболочке будет выведено:

|  |
| --- |
| 1 2 3 4 5 6 7 8 9 |

В момент, когда переменная **Y** примет значение 10, цикл **while** закончит свою работу, так как программа выведет значение **FALSE** (**Y** уже не меньше, а равен 10).

## Цикл until

Цикл **until** имеет аналогичную структуру, но выполняется несколько иначе:

|  |
| --- |
| until команда1 do  команда2 done |

Оператор **while** может иметь несколько условий одновременно. Последнее указанное условие определяет возможность продолжения цикла. В этом случае синтаксис оператора цикла будет отличаться от обычного.

Пример цикла **until**:

|  |
| --- |
| #!/bin/bash POINTER=20 until [ $ POINTER -lt 10 ]; do echo POINTER $ POINTER let POINTER -=1 done |

# Настройка планировщика Cron в Linux

В инструкции указано использование планировщика **Cron** с помощью утилиты **Crontab** в Linux.

**Cron** — системный демон, используемый для выполнения задач (в фоновом режиме) в указанное время.

**Crontab** — команда, которая используется для управления планировщиком **Cron**. Команда **crontab** создает файл **crontab**, содержащий команды и инструкции для запуска демона **cron**.

Каждая запись в **crontab** состоит из шести полей, указываемых в следующем порядке:

|  |
| --- |
| minute(s) hour(s) day(s) month(s) weekday(s) command(s) |

Поля разделяются пробелами или знаками табуляции. Первые пять представляют собой числовые значения, а шестое — это команда, которая будет выполняться.

Рассмотрим примеры.

Выполнять задание в 19 часов 5 минут 15 мая, если это пятница:

|  |
| --- |
| 5 19 15 5 5 /var/www\_mysite/myssqlss.pl |

Выполнять задание раз в два часа в 10 минут текущего часа (то есть в 00:10, 02:10, 04:10 и т.д.):

|  |
| --- |
| 10 \*/2 \* \* \* /var/www\_mysite /mysql\_script.pl |

Выполнять задание каждые десять часов в 10 минут текущего часа (то есть в 00:10, 10:10 и т.д.):

|  |
| --- |
| 10 \*/10 \* \* /var/www\_mysite /exescripts.pl |

Выполнять задание по воскресеньям в 12 часов 26 минут:

|  |
| --- |
| 26 12 \* \* 0 /var/www\_mysite /myscript.sh |

Выполнять в 10 утра каждый день:

|  |
| --- |
| 0 10 \* \* \* /var/www\_mysite /myscript.sh |

# Практическое задание

1. Вывести на экран 3 раза имя пользователя, от которого запускается команда.
2. Вывести с помощью цикла **while** все четные числа от 0 до 100 включительно.
3. Создать с помощью **nano** файл **test.txt**. Настроить автоматический бэкап этого файла раз в 10 минут в файл с названием **test.txt.bak** с использованием **cron**.

# Дополнительные материалы

1. [Цикл for в Bash](http://shonty.wmsite.ru/browse_articles/bash_for_loops).
2. [Популярный Linux. Циклы](https://poplinux.ru/adv-bash/chapter-10.1.html).
3. [Делаем «жизнь» в Linux проще, или Автоматизация запуска процессов с помощью cron](https://habr.com/ru/post/217655/).

# Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. [Циклы Bash](https://losst.ru/tsikly-bash).
2. [Использование планировщика cron в Linux](http://rus-linux.net/MyLDP/admin/cron.html).