# Фоновые процессы и ещё несколько полезных встроенных команд шелла

## Команда *alias*

Команда *alias* позволяет назначить короткое имя для вызова какой-нибудь команды с аргументами. По умолчанию в баше уже есть несколько алиасов.

Давайте посмотрим, какие алиасы уже заданы:

```
alias
```

Мы можем увидеть, что для команд *grep* и *ls* созданы алиасы, которые сразу включают подсветку разными цветами вывода команд.

Давайте создадим несколько алиасов. Указываем ключи -la:

```
alias lsa="ls -la"
ps
```

```
alias pss="ps -auxf"
```

Теперь нам не нужно будет набирать четыре ключа для того, чтобы получить нужный вид дерева процессов.

Чтобы эти изменения не исчезли после того, как мы закроем терминал, давайте перенесём их в файл .bashrc.

## Команда *declare*

Сама по себе команда *declare* просто назначает значение переменной, то есть мы получаем тот же результат, как если бы просто написали в скрипте:

```
letter=a
```

Давайте попробуем:

```
declare letter=a && echo $letter
```

Для того чтобы уничтожить переменную, используется другая команда — unset:

```
unset letter && echo $letter
```

Пока не очень полезно. Но у команды declare есть замечательный ключ -r. Он позволяет создать константу, то есть переменную, значение которой нельзя изменить.

```
declare -r letter=a && echo letter
declare letter=b
echo $letter
```

Как видим, ничего не изменилось.

Такое поведение полезно как для объявления каких-то математических констант, так и для более привычных вещей, например порога по времени, старше которого файлы удаляются.

## Фоновые процессы

Давайте научимся уводить в фон процессы самостоятельно.
Это очень полезная возможность, ведь, например, если в нашем скрипте мы будем копировать много разных файлов или очень большую директорию, консоль окажется заблокирована и нам придётся ждать, пока не выполнятся все действия.

Представим, что мы хотим запустить команду *ping* с отправкой не четырёх запросов, а 1000.

Нам потребуется команда ping с ключом -c 1000, далее перенаправим вывод в файл. И наконец, для того чтобы увести процесс в фон, добавим в конец команды символ амперсанд.

```
ping -c 1000 ya.ru > ping.txt &
```

Как мы видим, на консоли теперь не отображаются бегущие запросы, а есть только цифра в квадратных скобках и число. Цифра в скобках — это порядковый номер запущенной задачи. Число — PID, числовой идентификатор процесса.

Давайте в этом убедимся:

```
pss | grep PID
```

Отлично, видим запущенный процесс (и grep, который по-прежнему находит в списке свой собственный процесс с этим идентификатором в качестве аргумента).

Давайте посмотрим содержимое файла при помощи команды *less*. При помощи команды *jobs* мы можем посмотреть все фоновые задачи, которые мы запустили в этой консоли.

Когда команда завершится, задача из статуса running перейдёт в статус done.

Но что делать, если вы уже запустили команду? Для этого нужно нажать сочетание клавиш CTRL+Z, далее:

```
jobs
```

Мы видим сообщение о том, что процесс остановлен. Теперь мы можем перевести его в фон при помощи команды:

```
bg номер процесса
jobs
```

Видим, что процесс из stopped перешёл в running.

Если нам зачем-то хочется снова вывести процесс из фона и отдать ему консоль, нужно просто набрать:

```
fg номер процесса
```

и мы видим, что он снова с нами и будет держать консоль пока не завершится.

## Что из этого мы можем использовать в bash-скрипте?

Мы точно так же можем запустить из скрипта процесс в фоне при помощи значка амперсанда. После этого можно получить код возврата процесса при помощи echo $? И ещё кое-что.

Как мы с вами уже говорили, у каждого процесса существуют потоки стандартного ввода, стандартного вывода и ошибок.
Например, когда мы говорим:

```
host ya.ru
```

Информация, которую мы получаем, — это стандартный вывод.

Что, если мы не хотим получать эту информацию, а нам требуется только код ответа? Мы уже умеем перенаправлять поток стандартного вывода в файл — для этого мы просто пишем > /tmp/file. На самом деле, в этой конструкции подразумевается 1 (номер потока вывода) >/tmp/file, просто 1 в этом случае можно не писать.

Но что, если эта информация нам просто не нужна? В этом случае мы можем перенаправить её в специальное место /dev/null. Сюда можно успешно записать что угодно, но прочитать отсюда ничего нельзя.

Чтобы перенаправить в /dev/null поток ошибок, мы можем добавить 2>/dev/null. Существует более короткая запись для того, чтобы объединить потоки ошибок и вывода и перенаправить &> /dev/null. Таким образом мы можем сделать так, чтобы команда host не выводила никакой лишней информации, а только получила код возврата.

Если домен записан в файле /etc/hosts, команда host отработает успешно, вне зависимости, есть у вашего компьютера доступ к DNS-серверу или нет.