Условный оператор if

Вы умеете передавать стандартный вывод одной команды на стандартный ввод другой при помощи пайпа, умеете выполнять команды последовательно, указывая их через точку с запятой. Теперь нам требуется, чтобы вторая команда выполнилась, если успешно выполнится первая.

Этого мы можем добиться при помощи вот таких конструкций:

```
test -d /home/victoria/testdir && echo "Success! Directory exists."
```

```
test -d /home/victoria/testdir || echo "Error! Directory does not
exist."
```

&& — конструкция, которая говорит о том, что вторая команда выполнится, только если первая выполнилась успешно (то есть её код возврата 0).

|| — конструкция, которая выполнит вторую команду, только если первая вернула не нулевой код возврата (то есть какую-то ошибку). Если директории нет — об этом будет выведено сообщение.

&& и || называются логическим И и ИЛИ.

Такие конструкции — отличный помощник, и они широко используются в скриптах. Они всё-таки работают как однострочник.

В языках программирования это называется условным оператором.

Условный оператор — это некая конструкция языка программирования (в нашем случае, напоминаю, это bash), которая позволяет проверить некоторое условие, затем, если условие выполняется, выполнить одни действия, а в противном случае — другие.

Синтаксис

Синтаксис можно прочитать как текст.

Если сегодня 31 декабря, то время покупать ёлку, иначе ёлку покупать рано.

```
if [ дата = 31.12 ]
then
    echo "Время покупать ёлку!"
else
    echo "Ёлку пока покупать ещё рано."
fi
```

Квадратные скобки — это псевдоним команды test. То есть записи:

```
test -d /tmp
[ -d /tmp ]
```

совершенно равнозначны.

```
[ -d /tmp ] && echo "it exists!"
```

Между скобками и первым символом внутри скобок обязательно нужно оставлять пробел.

Баш-скрипт с условным оператором

Сначала, конечно, шебанг (she-bang) и путь до bash:

```
#!/bin/bash
```

Создадим переменную и запишем в неё сегодняшнюю дату, число и месяц, используя команду *date*-. Назовём нашу переменную:

```
today=$(date +'%d-%m')
```

и теперь пишем цикл.

Если в переменной today значение совпадает с 31 декабря:

```
if [ $today =31.12 ]
then
     echo "Happy New Year!"
else
в противном случае давайте напомним сегодняшний день
     echo "Today is $today"
fi
```

Сохраним файл. Выставим право на исполнение и запустим наш скрипт.
Ошибка.

Продолжим улучшать наш скрипт.

Сначала мы добавим циклы, которые будут проверять существование наших директорий, и если хотя бы одна не существует или является файлом, выведем ошибку.

Используем -d (проверка того, что объект существует и директория) и добавим отрицание !:

```
if [ ! -d $1 ]
then
   echo "Error! Directory $1 does not exist or not a directory"
fi
```

И второй такой же:

```
if [ ! -d $1 ]
then
   echo "Error! Directory $2 does not exist or not a directory"
fi
```

Давайте сделаем так, чтобы директория удалялась только после успешного копирования.

```
cp -r $1 $2 && rm -rf $1
```

И после этого давайте выведем сообщение:

```
echo "Directory $1 was successfully moved to $2"
```

Отлично, теперь давайте поправим комментарии.

Добавим комментарий о том, что мы проверяем существование обеих директорий. Добавим комментарий о том, что удаление будет происходить только после успешного копирования.

Хорошим тоном считается возможность команды выводить короткую справку о самой себе.

```
if [ $# -ne 2 ]
   then echo "2 arguments are needed, directory being moved and target
directory"
   exit 1
fi
```

Теперь, если мы наберём имя скрипта, он расскажет нам, сколько и какие аргументы ему необходимы.

Выглядит так, что мы избавились от недокументированной возможности (файл не скопируется, а выдаст ошибку), сделали сообщение об успешном завершении переноса директории, добавили сообщения об ошибках, стали проверять, существуют ли те директории, которые передаются скрипту.

Давайте попробуем его выполнить:

```
./mvdir
```

С одной стороны, мы действительно получили информацию о том, что требуется два аргумента. С другой, очевидно, что скрипт продолжил пытаться выполниться (хотя и выводил ошибки). И, что гораздо хуже, вывел сообщение о том, что директория скопирована успешно.

Для остановки выполнения скрипта существует команда exit. У неё очень простой синтаксис:

```
exit N
```

Вместо N подставляется тот код возврата, который мы хотим отдать.
Если код возврата не указать, он берётся из последней выполненной команды.
exit 0 означает успешное выполнение.
exit 1 и так далее — ошибку.
Если вы запустите exit не в скрипте, а просто в консоли, он консоль закроет.

Добавим после выводов сообщения об ошибке exit 1. Здесь нам не очень важно подбирать правильное значение кода возврата, нам нужно только остановить выполнение скрипта.

Напишем ещё один условный оператор:

```
if  cp -r $1 $2  &&  rm -rf $1
```

```
then
  echo "directory $1 was successfully moved to $2"
```

После этого можно завершить скрипт с нулевым кодом:

```
  exit 0
else
  echo "Error!"
  exit 1
fi
```

Давайте попробуем вызвать скрипт без аргументов:

```
./mvdir
```

И протестируем его в случае с несуществующей директорией:

```
./mvdir aaaaa forest
```

И в случае, когда все директории существуют:

```
./mvdir tree2 forest
```

Выглядит всё довольно неплохо. Но возникает вот какой вопрос. Аргументы скрипта разделяются пробелами. А что случится, если в названии папки будет пробел? Давайте проверим:

```
mkdir "Untitled Folder"
./mvdir.sh
```

Как оказалось, ничего хорошего. Никакого предохранителя на такой случай нет.

Для того чтобы этого избежать, давайте возьмём в кавычки позиционные переменные и проставим кавычки в нашем скрипте.