Цикл for

Познакомимся с ещё одним видом циклов — **циклом for**.

Пример: у нас нет управляющего условия — есть только список доменных имён, которые нужно последовательно передать команде ping. Таким же образом, как если бы мы просто пять раз запустили команду с разными параметрами:

```
ping -c4 ya.ru
ping -c4 vk.com
ping -c4 google.com
ping -c4 facebook.com
ping -c4 twitter.com
```

Чтобы избежать потери времени и копировать одни и те же команды, воспользуемся циклом for. Он позволяет нам задать список объектов и выполнить какие-то действия для каждого из этих объектов.

Например, случай формирования списка рассылки:

```
for arg in [list]
do
 команда(ы)...
done
```

Его синтаксис похож на цикл while: у нас есть ключевое слово for, после которого обычно перечисляется список аргументов, для каждого из которых выполняется тело цикла, заключённое в do, done.

Давайте напишем наш пример с командой ping:
for i in ya.ru vk.com google.com facebook.com twitter.com

Мы тоже можем использовать переменную i — у нас небольшой цикл, переменная используется только в его рамках. Далее перечислим список доменов, для каждого из которых мы будем запускать ping:

```
do
ping -c4 -q $i
```

Здесь нам не нужен полный вывод команды, поэтому можно воспользоваться ключом -q.

```
done
```

Давайте сохраним скрипт и запустим его.

Сделаем наш скрипт чуть более удобным для пользователя:

```
ping -c4 -q $i &>/dev/null && echo "$i is available" || echo "$i is unreachable"
```

Посмотрим, какая тут логика. Во-первых, мы использовали уже знакомое нам обозначение для того, чтобы объединить потоки стандартного вывода и ошибок и перенаправить их в специальную «чёрную дыру». Затем, если команда ping завершается с кодом 0, мы выводим сообщение о том, что доменное имя доступно, по нему нам кто-то отвечает. А вот если одно из этих действий завершится с ненулевым кодом возврата (это будет ping, потому что у echo завершиться с ненулевым кодом шансов не очень много, он просто выводит строку), выводится сообщение о том, что по этому доменному имени нам не отвечают.

Использование циклов в однострочниках

Пример: напишем короткую версию скрипта, который мы писали выше:

```
for i in ya.ru vk.com google.com; do ping -c4 -q $i; done
```

Это правило верно и для цикла while/until:

```
while true; do echo "Hello World!"; done
```

Список для цикла не обязательно перечислять вручную. Например, мы можем использовать подстановку по шаблону:

```
for i in  *; do echo $i; done
```

Или можем взять результат выполнения команды — для этого вспомним конструкцию «доллар со скобками» (либо бэктики).

```
for i in $(ls /tmp/); do echo $i; done
```

Таким образом мы, например, можем получить все имена файлов и поддиректорий в текущей директории.

Использование цикла for для выполнения действий счётное количество раз

Давайте выведем фразу Hello World десять раз, используя цикл for.
Для этого нам потребуется написать:

```
for i in 1 2 3 4 5 6 7 8 9 10; do echo "Hello World!"; done
```

Но мы вручную ввели 10 цифр. Давайте улучшим это.

Способ 1. Цикл for поддерживает C-подобную форму записи:

```
for ((i=1;i<=10;i++))
do
echo "Hello World!"
done
```

Тут мы возвращаемся к использованию счётчика. То есть мы не указываем готовый список значений, а указываем первое значение счётчика (инициализируем его), условие для проверки значения (меньше либо равно 10). Это действие счётчика, исходя из логики использования счётчиков в while, тут должно быть что-то вроде i=i+1.
На самом деле i++ — это эквивалентная запись. Мы точно так же сохраняем в переменную её предыдущее значение, плюс единица, но в более короткой форме записи.

Способ 2. Программа, которая напечатает для нас последовательность чисел, называется seq (от sequence).
Мы можем вывести, например, числа от 1 до 3:

```
seq 1 3
```

Или от 3 до 1:

```
seq 3 1
```

Давайте перепишем наш цикл с использованием этого варианта:

```
for i in $(seq 1 10); do echo "Hello World!"; done
```

И не забываем указывать, что мы берём результат от выполнения команды.

Как и в случае с циклом while/until и условным оператором, мы можем вкладывать в цикл for другие циклы и операторы. А также использовать команды break и continue.