

# Классический цикл for-loop

## Краткое описание

```
для <ИМЯ>; сделать  
<СПИСОК>  
Выполнено
```

```
для <ИМЯ> в <СЛОВА>; сделать  
<СПИСОК>  
Выполнено
```

альтернативный, исторический и недокументированный синтаксис <sup>1)</sup>

```
для <ИМЯ>; {  
  <СПИСОК>  
}  
  
для <ИМЯ> в <СЛОВА>; {  
  <СПИСОК>  
}
```

## Описание

Для каждого слова в <WORDS> выполняется одна итерация цикла, и переменной <NAME> присваивается значение текущего слова. Если нет " in <WORDS> ", чтобы задать собственный список слов, тогда используются позиционные параметры ( "\$@" ) (аргументы скрипта или функции). В этом случае (и только в этом случае) точка с запятой между именем переменной и do является необязательной.

Если вы используете переменную цикла внутри цикла for и она может содержать пробелы, вам нужно заключить ее в кавычки, поскольку применяются обычные процедуры разделения слов.

⚠ Как и все циклы (оба for цикла, while и until ), этот цикл может быть

- завершается (прерывается) break командой, необязательно break N для разрыва N уровней вложенных циклов
- вынужден немедленно выполнить следующую итерацию, используя continue команду, необязательно как continue N аналог break N

Bash знает альтернативный синтаксис для `for` цикла, заключая тело цикла в `{...}` вместо `do ... done`:

```
для x в 1 2 3
{
    echo $x
}
```

Этот синтаксис **не документирован** и не должен использоваться. Я нашел определения синтаксического анализатора для него в коде 1.x и в современном коде 4.x. Я предполагаю, что он существует по соображениям совместимости. Этот синтаксис не задан POSIX®.

## Возвращает статус

Статус возврата - это статус последней команды, выполненной в `<LIST>` or `0 ( TRUE )`, если список элементов `<WORDS>` ничего не оценивает (т. Е.: "пусто!").

## Примеры

### Перебор элементов массива

С помощью некоторого синтаксиса массива (см. Массивы) вы можете легко "передать" цикл `for` для перебора всех элементов в массиве (путем массового расширения всех элементов):

```
для элемента в "${myarray[@]}"; выполнить
    эхо "Элемент: $element"
готово
```

Другой способ - массовое расширение всех используемых индексов и доступ к массиву по индексу:

```
для индекса в "${!myarray[@]}"; выполнить
    эхо "Элемент [$index]: ${myarray[$index]}"
готово
```

### Список позиционных параметров

Вы можете использовать эту функцию, чтобы проверить, как аргументы команды будут интерпретироваться, анализироваться и, наконец, использоваться:

```
argtest() {  
  n= 1  
  для аргумента; выполнить  
    echo "Аргумент $((n ++)): \"$arg \""  
  готово  
}
```

## Цикл по каталогу

Поскольку расширение имени пути приведет к расширению всех имен файлов до отдельных слов, независимо от пробелов, вы можете использовать цикл for для перебора имен файлов в каталоге:

```
для fn в *; выполните  
  if [ -h "$fn" ]; затем  
    echo -n "Символическая ссылка: "  
  elif [ -d "$fn" ]; затем  
    echo -n "Dir: "  
  elif [ -f "$fn" ]; затем  
    echo -n "File: "  
  else  
    echo -n "Неизвестно: "  
  fi  
  echo "$fn"  
готово
```

Глупый пример, я знаю 😊

## Цикл по строкам вывода

Для полноты: вы можете изменить разделитель внутренних полей (IFS) на новую строку и, таким образом, создать цикл for, повторяющийся по строкам вместо слов:

```
IFS=$' \n'  
' for f в $(ls); выполнить  
  echo $f
```

Это всего лишь пример. В *общем*

- это не очень хорошая идея для анализа `ls(1)` выходных данных
- цикл while (с использованием `read` команды) - лучший способ перебора строк

## Вложенные циклы for

Конечно, можно использовать другой цикл for как `<LIST>`. Здесь, считая от 0 до 99 странным образом:

```
для x в 0 1 2 3 4 5 6 7 8 9; делать
  для y в 0 1 2 3 4 5 6 7 8 9; do
    echo $ x $ y
  сделано
сделано
```

## Цикл по диапазону чисел

Начиная с Bash 4, вы также можете использовать синтаксическую форму расширения фигурных скобок в виде выражения последовательности при циклическом переборе чисел, и эта форма не создает начальные нули, если вы не попросите их:

```
для
  чисел # 100 нет начальных нулей x в {0..99}; выполнить
  echo $x
```

```
для
  # любого другого числа ширина 3 x в {000..99..2}; выполнить
  echo $ x
```

**ПРЕДУПРЕЖДЕНИЕ:** весь список создается до начала цикла. Если ваш список огромен, это может быть проблемой, но не более, чем для глобуса, который расширяется до огромного списка.

## Соображения о переносимости



## Смотрите также

- Цикл for в стиле C

1)

[http://pubs.opengroup.org/onlinepubs/9699919799/xrat/V4\\_xcu\\_chap02.html#tag\\_23\\_02\\_09\\_12](http://pubs.opengroup.org/onlinepubs/9699919799/xrat/V4_xcu_chap02.html#tag_23_02_09_12)  
([http://pubs.opengroup.org/onlinepubs/9699919799/xrat/V4\\_xcu\\_chap02.html#tag\\_23\\_02\\_09\\_12](http://pubs.opengroup.org/onlinepubs/9699919799/xrat/V4_xcu_chap02.html#tag_23_02_09_12))

## Обсуждение

 [syntax/ccmd/classic\\_for.txt](#)  Последнее редактирование: 01.01.19 2017 21:02 автор: 4dummies

---

Этот сайт поддерживается Performing Databases - вашими  
экспертами по администрированию баз данных

---

Bash Hackers Wiki

---



Если не указано иное, содержимое этой вики лицензируется по следующей лицензии:  
Лицензия GNU Free Documentation 1.3