# Массивы bash

- *nix

*Предлагаю вашему вниманию перевод статьи Митча Фрейзера (Mitch Frazier) "Bash Arrays" с сайта linuxjournal.com.*

Если вы используете «стандартную» оболочку *NIX-системы, возможно, вы не знакомы с такой полезной особенностью bash как массивы. Хотя массивы в bash не так круты, как в P-языках (Perl, Python и PHP) и других языках программирования, они часто бывают полезны.
Bash-массивы имеют только численные индексы, но они не обязательны к использованию, то есть вы не должны определять значения всех индексов в явном виде. Массив целиком может быть определен путем заключения записей в круглые скобки:

```
arr=(Hello World)
```

Отдельные записи могут быть определены с помощью знакомого всем синтаксиса (от Бейсика (да простит меня Дейкстра — прим. переводчика) до Фортрана):

```
arr[0]=Hello
```

```
arr[1]=World
```

Правда, обратное выглядит сравнительно более уродливо. Если нужно обратиться к определенной записи, тогда:

```
echo ${arr[0]} ${arr[1]}
```

Из страницы man:

"*Фигурные скобки нужны для предотвращения конфликтов при разворачивании полных путей к файлам.*"

Кроме того, доступны следующие странноватые конструкции:

```
${arr[*]} # Все записи в массиве
```

```
${!arr[*]}# Все индексы в массиве
```

```
${#arr[*]}# Количество записей в массиве
```

```
${#arr[0]}# Длина первой записи (нумерация с нуля)
```

${!arr[*]} — сравнительно новое дополнение в bash и не является частью оригинальной реализации. Следующая конструкция демонстрирует пример простого использования массива. Обратите внимание на "[index]=value", это позволяет назначить конкретное значение конкретному номеру записи.

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

```
array=(one two three four [5]=five)
```

```
echo "Array size: ${#array[*]}"  # Выводим размер массива
```

```
echo "Array items:" # Выводим записи массива
```

```
for item in ${array[*]}
```

```
do
```

```
    printf "   %s\n" $item
```

```
done
```

```
echo "Array indexes:" # Выводим индексы массива
```

```
for index in ${!array[*]}
```

```
do
```

```
    printf "   %d\n" $index
done

echo "Array items and indexes:" # Выводим записи массива с их индекса
ми
for index in ${!array[*]}
do
    printf "%4d: %s\n" $index ${array[$index]}
done
```

Запуск скрипта породит следующий вывод:

Array size: 5

Array items:

one
two
three
four
five

Array indexes:

0
1
2
3
5

Array items and indexes:

0: one
1: two
2: three
3: four
5: five

Обратите внимание, что символ "@" может быть использован вместо "*" в конструкциях типа {arr[*]}, результат будет одинаковым за исключением разворачивания записи в кавычках. "$*" и "$@" выведут записи в кавычках, "${arr[*]}" вернет каждую запись как одно слово, "${arr[@]}" вернет каждую запись в виде отдельных слов.

Следующий пример покажет, как кавычки и конструкции без кавычек возвращают строки (особенно важно, когда в этих строках есть пробелы):

```
#!/bin/bash

array=("first item" "second item" "third" "item")

echo "Number of items in original array: ${#array[*]}"
for ix in ${!array[*]}
do
   printf "   %s\n" "${array[$ix]}"
done
echo

arr=(${array[*]})
echo "After unquoted expansion: ${#arr[*]}"
for ix in ${!arr[*]}
do
   printf "   %s\n" "${arr[$ix]}"
done
```

```
echo

arr=("${array[*]}")
echo "After * quoted expansion: ${#arr[*]}"
for ix in ${!arr[*]}
do
    printf "   %s\n" "${arr[$ix]}"
done
echo

arr=("${array[@]}")
echo "After @ quoted expansion: ${#arr[*]}"
for ix in ${!arr[*]}
do
    printf "   %s\n" "${arr[$ix]}"
done
```

Вывод при запуске:

```
Number of items in original array: 4

first item
second item
third
item

After unquoted expansion: 6
```

first
item
second
item
third
item

After * quoted expansion: 1

first item second item third item

After @ quoted expansion: 4

first item
second item
third
item

**Теги:**

bash scripting