

# Команда declare builtin

## Краткое описание

```
объявить [-aAfFgIlNrTux] [-p] [ИМЯ[=ЗНАЧЕНИЕ] ...]
```

# устаревший набор синонимов

```
набор текста [-aAfFgIlNrTux] [-p] [ИМЯ[=ЗНАЧЕНИЕ] ...]
```

## Описание

`declare` используется для отображения или задания переменных вместе с атрибутами переменных. При использовании для отображения переменных / функций и их значения выходные данные могут быть повторно использованы в качестве входных данных для оболочки.

Если `NAME` задано значение `po`, отображаются значения всех переменных или функций, когда они ограничены `-f` параметром.

Если `NAME` за ней следует `=VALUE`, `declare` также задает значение для переменной.

При использовании в функции `declare` создает `NAMEs` локальные переменные, если они не используются с `-g` параметром.

Не используйте ее синоним `typeset` при написании кода для Bash, поскольку она помечена как устаревшая.

## Опции

Ниже `[-+]` указан атрибут, используемый `-x` для установки атрибута, `+x` чтобы удалить его.

Опция	Описание
<code>[-+]</code> a	создание имен индексированных массивов (удаление с <code>+a</code> помощью допустимого синтаксиса, но приводит к сообщению об ошибке)
<code>[-+]</code> A	сделайте имена ассоциативными массивами

Опция	Описание
-------	----------

[ - + ]c	<b>Недокументированное</b> преобразование имен в "cарcase" при назначении (делает первую букву заглавной, а остальные строчными). Требуется, чтобы Bash был построен с -DCASEMOD_CAPCASE
-f	ограничить действие или отображение именами и определениями функций (удаление с +f помощью допустимого синтаксиса, но приводит к сообщению об ошибке)
-F	ограничить отображение только именами функций (плюс номер строки и исходный файл при отладке)
-g	создавать глобальные переменные при использовании в функции оболочки; в противном случае игнорируется (по умолчанию declare объявляются локальные переменные области видимости при использовании в функциях оболочки)
[ - + ]i	сделайте так, чтобы имена имели атрибут "integer"
[ - + ]l	преобразуйте имена в нижний регистр при назначении (убедитесь, что переменная содержит только строчные буквы)
[ - + ]n	сделайте NAME ссылкой на переменную, названную по ее значению. Введена в Bash 4.3-alpha. \${!NAME} показывает имя ссылочной переменной, ЗНАЧЕНИЕ. Используется unset -n NAME для отмены установки переменной. ( unset -v NAME сбрасывает ЗНАЧЕНИЕ переменной.) Используется [[ -R NAME ]] для проверки, установлено ли значение NAME в ЗНАЧЕНИЕ, имя другой переменной.
-p	отображение атрибутов и значения каждого ИМЕНИ
[ - + ]r	сделать имена доступными только для чтения (удаление с +r помощью допустимого синтаксиса, но невозможно)
[ - + ]t	создайте имена с атрибутом "трассировка" (действует только для функций)
[ - + ]u	преобразуйте имена в верхний регистр при назначении (убедитесь, что переменная содержит только буквы верхнего регистра)
[ - + ]x	сделать имена экспортируемыми

## Возвращает статус

Статус	Причина
--------	---------

0	ошибки нет
!= 0	недопустимый параметр

## Статус Причина

!= 0	задано недопустимое имя переменной
!= 0	попытка <b>определить</b> функцию с помощью <code>-f</code>
!= 0	присвоение переменной, доступной только для чтения
!= 0	удаление атрибута только для чтения из переменной только для чтения
!= 0	присваивание переменной массива без синтаксиса составного присваивания ( <code>array=(...)</code> )
!= 0	попытка использовать <code>+a</code> для "уничтожения" массива
!= 0	попытка отобразить несуществующую функцию с помощью <code>-f</code>

## Примечания

Оболочки Unix предлагают очень мало типов данных. Bash и некоторые другие оболочки расширяют это, позволяя устанавливать "атрибуты" для имен переменных. Единственными атрибутами, указанными в POSIX, являются `export` and `readonly`, которые устанавливаются их собственными встроенными. Типы данных в bash имеют несколько других интересных возможностей, таких как возможность изменять данные при назначении.

## Примеры

### Отображение определенных функций

`declare -f` может использоваться для отображения всех определенных функций...

```
$ declare -f
foo ()
{
  echo "FOO - это BAR"
}
world ()
{
  echo "Привет, мир!"
}
```

... или просто определенная определенная функция.

```
$ declare -f foo
foo ()
{
  echo "FOO - это BAR"
}
```

## Nameref

Bash 4.3 добавляет новый способ косвенной ссылки на переменные. `typeset -n` или `declare -n` может использоваться, чтобы заставить переменную косвенно ссылаться на другую. В Bash значение присваивания, присвоенное `typeset -n` or `declare -n`, будет ссылаться на переменную, имя которой расширяется в RHS.

`typeset -n` используется в примере ниже. Смотрите примечания ниже.

```
# Суммируйте набор массивов и присваивайте результат косвенно, также
печатайте каждый промежуточный результат (без обходных путей переносимо
сти)
# sum имя arrname [ arrname ... ]
функция sum {
    набор текста -n _result=$1 _arr
    набор текста IFS=+
    _result=0
    для _arr в "${@:2}"; # Продемонстрировать специальное свойство "for"
    в ссылке на имя.
    (( _result += ${_arr[*]} ))
    printf '%s = %d \n' "${!_result}" "$_result" # Продемонстрировать с
    пециальное свойство $ {!ссылка} на ссылку на имя.
    Выполнено
}

a =(1 2 3) b =(6 5 4) c = (2 4 6)
итоговая сумма a b c
printf 'Конечное значение "итого" равно: %d \n' "$ total"
```

`typeset -n` в настоящее время реализована в ksh93, mksh и Bash 4.3. Реализации Bash и mksh очень похожи, но сильно отличаются от ksh93. Подробности см. в разделе Соображения о переносимости. ссылки на имена ksh93 намного мощнее, чем у Bash.



## Соображения о переносимости

- `declare` не указана в POSIX®
- `declare` уникальна для Bash и полностью непереносима, за исключением, возможно, Zsh в режиме совместимости с Bash. Bash помечает синоним `typeset` как устаревший, который в Bash ведет себя идентично `declare`. Все другие оболочки, подобные Korn, используют `typeset`, так что, вероятно, это не исчезнет в ближайшее время. К сожалению, будучи нестандартной встроенной, `typeset` она значительно отличается в разных оболочках. ksh93 также учитывает `typeset` специальную встроенную команду, в то время как Bash этого не делает - даже в режиме POSIX. Если вы используете `typeset`, вы должны пытаться использовать ее только переносимыми способами.
- **Переносимость todo** `nameref`...

## Смотрите также

- Массивы
- Встроенная команда только для чтения
- Команда unset builtin
- команды объявления (<http://austingroupbugs.net/view.php?id=351>) изменят поведение определенных встроенных функций, например, `export` в следующей версии POSIX.

## Обсуждение

 [commands/builtin/declare.txt](#)  Последнее редактирование: 2022/11/09 01:48 автор fgrouse

---

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

---

Bash Hackers Wiki



Except where otherwise noted, content on this wiki is licensed under the following license:  
GNU Free Documentation License 1.3