

Встроенная команда shift

Синописис

```
сдвиг [n]
```

Описание

`shift` Встроенная команда используется для "сдвига" позиционных параметров на заданное число `n` или на 1, если число не задано.

Это означает, что количество и положение позиционных параметров изменяются. Самый первый позиционный параметр отбрасывается, второй становится первым и т.д.

Представьте себе следующий набор позиционных параметров (`$1` to `$4`):

1	Это
2	есть
3	а
4	тест

Когда вы будете использовать `shift 1` , они будут изменены на:

1	есть
2	а
3	тест

Специальный параметр `$#` будет отражать конечное количество позиционных параметров.

Если заданное число равно 0, позиционные параметры не изменяются.

Опции

Вариантов нет.

Статус возврата

Статус	Причина
0	ошибки нет
1	нечисловой аргумент
1	заданное число (или значение по умолчанию 1) больше, чем количество фактически присутствующих позиционных параметров
1	данное число является отрицательным

Примеры

Соображения по переносимости

- `shift` Встроенная команда определяется POSIX®.
- Многие оболочки выдают фатальную ошибку при попытке `shift` ввести больше, чем количество позиционных параметров. **POSIX не требует такого поведения.** Bash (даже в режиме POSIX) и Zsh возвращают 1, когда нет аргументов, и вывод ошибок не производится, если не включена опция `shift_verbose` shopt . Ksh93, pdksh, posh, mksh и dash, все выдают бесполезные фатальные ошибки оболочки.

```
$ dash -c 'f() { если shift; то echo "$ 1"; иначе echo "без аргу
ментов"; fi;}; f'
dash: 1: shift: не могу сдвинуть так много
```

В большинстве оболочек вы можете обойти эту проблему, используя встроенную команду для подавления неустраняемых ошибок, вызванных *специальными встроенными* командами.

```
$ dash -c 'f() { если команда shift 2>/dev/null; затем echo
"$1"; иначе echo "нет аргументов"; fi; }; f'
нет аргументов
```

Хотя POSIX требует такого поведения, оно не очень очевидно, и некоторые оболочки делают это неправильно. Чтобы обойти это, вы можете использовать что-то вроде:

```
$ mksh -c 'f() { if! ${1+false} && shift; затем echo "$1"; else echo
"без аргументов"; fi; }; f'
без аргументов
```

~~Исправлено shift, что сопровождающий mksh отказывается изменять command~~
встроенные или ..

(<https://github.com/MirBSD/mksh/commit/996e05548ab82f7ef2dea61f109cc7b6d13837fa>)
(Спасибо!)



- Возможно, почти так же плохо, как и выше, `busybox sh shift` всегда возвращает успех, даже при попытке перейти за пределы последнего аргумента.

```
$ bb -c 'f() {если shift; то echo "$1"; иначе echo "без аргумен  
тов"; fi;}; f'  
(нет вывода)
```

Описанный выше обходной путь `mksh` будет работать и в этом случае.

См . также

Обсуждение

 `commands/builtin/shift.txt`  Последнее изменение: 2015/05/10 03:57 автор : ormaaj

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

Bash Хакеры Вики



Если не указано иное, содержимое этой вики-страницы лицензируется по следующей лицензии:
Лицензия на бесплатную документацию GNU 1.3