

# The shift builtin command

## Synopsis

```
shift [n]
```

## Description

The `shift` builtin command is used to "shift" the positional parameters by the given number `n` or by 1, if no number is given.

This means, the number and the position of the positional parameters are changed. The very first positional parameter is discarded, the second becomes the first one, etc.

Imagine the following set of positional parameters ( `$1` to `$4` ):

1	This
2	is
3	a
4	test

When you use `shift 1`, they will be changed to:

1	is
2	a
3	test

The special parameter `$#` will reflect the final number of positional parameters.

If the number given is 0, no changes are made to the positional parameters.

## Options

There are no options.

## Return status

Status	Reason
0	no error
1	non-numeric argument
1	given number (or the default 1) is bigger than the number of actually present positional parameters
1	given number is negative

## Examples

## Portability considerations

- The `shift` builtin command is specified by POSIX®.
- Many shells will throw a fatal error when attempting to `shift` more than the number of positional parameters. **POSIX does not require that behavior.** Bash (even in POSIX mode) and Zsh return 1 when there are no args, and no error output is produced unless the `shift_verbose` shopt option is enabled. Ksh93, pdksh, posh, mksh, and dash, all throw useless fatal shell errors.

```
$ dash -c 'f() { if shift; then echo "$1"; else echo "no args";
fi; }; f'
dash: 1: shift: can't shift that many
```

In most shells, you can work around this problem using the command builtin to suppress fatal errors caused by *special builtins*.

```
$ dash -c 'f() { if command shift 2>/dev/null; then echo "$1";
else echo "no args"; fi; }; f'
no args
```

While, POSIX requires this behavior, it isn't very obvious and some shells don't do it correctly. To work around this, you can use something like:

```
$ mksh -c 'f() { if ! ${1+false} && shift; then echo "$1"; else echo
"no args"; fi; }; f'
no args
```

~~The mksh maintainer refuses to change either the `shift` or `command` builtins.~~ Fixed (<https://github.com/MirBSD/mksh/commit/996e05548ab82f7ef2dea61f109cc7b6d13837fa>). (Thanks!)

- Perhaps almost as bad as the above, busybox sh's `shift` always returns success, even when attempting to shift beyond the final argument.

```
$ bb -c 'f() { if shift; then echo "$1"; else echo "no args"; f
i; }; f'
(no output)
```



The above mksh workaround will work in this case too.

## See also

---

## Discussion

---

 [commands/builtin/shift.txt](#)  Last modified: 2015/05/10 03:57 by ormaaj

---

This site is supported by Performing Databases - your experts for database administration

---

Bash Hackers Wiki

---



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