Chapter 33. Options

Options are settings that change shell and/or script behavior.

The set command enables options within a script. At the point in the script where you want the options to take effect, use **set -o option-name** or, in short form, **set -option-abbrev**. These two forms are equivalent.

```
#!/bin/bash

set -o verbose
# Echoes all commands before executing.
```

```
#!/bin/bash

set -v
# Exact same effect as above.
```

To *disable* an option within a script, use **set +o option-name** or **set +option-abbrev**.

```
#!/bin/bash

set -o verbose
# Command echoing on.
command
...
```

```
command

set +o verbose
# Command echoing off.
command
# Not echoed.


set -v
# Command echoing on.
command
...
command

set +v
# Command echoing off.
command

exit 0
```

An alternate method of enabling options in a script is to specify them immediately following the `#!` script header.

```
#!/bin/bash -x
#
# Body of script follows.
```

It is also possible to enable script options from the command line. Some options that will not work with **set** are available this way. Among these are `-i`, force script to run interactive.

`bash -v script-name`

`bash -o verbose script-name`

The following is a listing of some useful options. They may be specified in either abbreviated form (preceded by a single dash) or by complete name (preceded by a *double* dash or by `-o`).

Table 33-1. Bash options

Abbreviation	Name	Effect
`-B`	brace expansion	*Enable* brace expansion (default setting = *on*)
`+B`	brace expansion	*Disable* brace expansion
`-C`	noclobber	Prevent overwriting of files by redirection (may be overridden by >|)
`-D`	(none)	List double-quoted strings prefixed by $, but do not execute commands in script
`-a`	allexport	Export all defined variables
`-b`	notify	Notify when jobs running in background terminate (not of much use in a script)
`-c ...`	(none)	Read commands from **...**
`checkjobs`		Informs user of any open jobs upon shell exit. Introduced in version 4 of Bash, and still "experimental." *Usage:* shopt -s checkjobs (*Caution:* may hang!)
`-e`	errexit	Abort script at first error, when a command exits with non-zero status (except in until or while loops, if-tests, list constructs)
`-f`	noglob	Filename expansion (globbing) disabled
`globstar`	*globbing* star-match	Enables the ** globbing operator (version 4+ of Bash). *Usage:* shopt -s globstar
`-i`	interactive	Script runs in *interactive* mode
`-n`	noexec	Read commands in script, but do not execute them (syntax check)
`-o Option-Name`	(none)	Invoke the *Option-Name* option
`-o posix`	POSIX	Change the behavior of Bash, or invoked script, to conform to POSIX standard.
`-o pipefail`	pipe failure	Causes a pipeline to return the exit status of the last command in the pipe that returned a non-zero return value.
`-p`	privileged	Script runs as "suid" (caution!)

Abbreviation	Name	Effect
`-r`	restricted	Script runs in *restricted* mode (see Chapter 22).
`-s`	stdin	Read commands from `stdin`
`-t`	(none)	Exit after first command
`-u`	nounset	Attempt to use undefined variable outputs error message, and forces an exit
`-v`	verbose	Print each command to `stdout` before executing it
`-x`	xtrace	Similar to `-v`, but expands commands
`-`	(none)	End of options flag. All other arguments are positional parameters.
`--`	(none)	Unset positional parameters. If arguments given (`-- arg1 arg2`), positional parameters set to arguments.

Prev Home Next
Debugging Up Gotchas