You are here / 🏠 / internals / List of shell options

internals:shell_options

List of shell options

This information was taken from a Bash version " `4.1` ", every now and then new options are added, so likely, this list isn't complete.

The shell-options can be set with the shopt builtin command.

Shell options

autocd

Option:	autocd	Since:	4.0-alpha
Shell mode:	interactive only	Default:	off

If set, a command name that is the name of a directory is executed as if it were the argument to the cd command.

cdable_vars

Option:	cdable_vars	Since:	unknown
Shell mode:	all	Default:	off

Treat every **non-directory argument** to the `cd` -command as variable name containing a directory to `cd` into.

cdspell

Option:	cdspell	Since:	unknown
Shell mode:	interactive only	Default:	off

If set, minor errors in the spelling of a directory component in a cd command will be corrected. The errors checked for are transposed characters, a missing character, and one character too many. If a correction is found, the corrected file name is printed, and the command proceeds.

checkhash

Option:	checkhash	Since:	unknown
Shell mode:	all	Default:	off

If set, Bash checks that a command found in the hash table exists before trying to execute it. If a hashed command no longer exists, a normal path search is performed.

checkjobs

Option:	checkjobs	Since:	4.0-alpha
Shell mode:	interactive only	Default:	off

If set, Bash lists the status of any stopped and running jobs before exiting an interactive shell. If any jobs are running, this causes the exit to be deferred until a second exit is attempted without an intervening command. The shell always postpones exiting if any jobs are stopped.

checkwinsize

Option:	checkwinsize	Since:	unknown
Shell mode:	all	Default:	off

If set, Bash checks the window size after each command and, if necessary, updates the values of the variables LINES and COLUMNS.

cmdhist

Option:	cmdhist	Since:	unknown
Shell mode:	all	Default:	off

If set, Bash attempts to save all lines of a multiple-line command in the same history entry. This allows easy re-editing of multi-line commands.

compat31

Option:	compat31	Since:	3.2
Shell mode:	all	Default:	off

Compatiblity mode for Bash 3.1

compat32

Option:	compat32	Since:	4.0
Shell mode:	all	Default:	off

Compatiblity mode for Bash 3.2

compat40

Option:	compat40	Since:	4.1-beta
Shell mode:	all	Default:	off

Compatiblity mode for Bash 4.0

compat41

Option:	compat41	Since:	4.2-alpha
Shell mode:	all	Default:	off

Compatiblity mode for Bash 4.1

direxpand

Option:	direxpand	Since:	4.3-alpha
Shell mode:	all	Default:	off (unless changed on compile-time with `--enable-direxpand-default`)

If set, bash replaces directory names with the results of word expansion when performing filename completion. This changes the contents of the readline editing buffer. If not set, bash attempts to preserve what the user typed.

dirspell

Option:	dirspell	Since:	4.0-alpha
Shell mode:	all	Default:	off

If set, Bash will perform spelling corrections on directory names to match a glob.

dotglob

Option:	dotglob	Since:	unknown

Shell mode:	all	**Default:**	off

If set, Bash includes filenames beginning with a `.` (dot) in the results of pathname expansion.

execfail

Option:	execfail	**Since:**	unknown
Shell mode:	non-interactive	**Default:**	off

If set, a non-interactive shell will not exit if it cannot execute the file specified as an argument to the `exec`-builtin command. An interactive shell does not exit if `exec` fails.

expand_aliases

Option:	expand_aliases	**Since:**	unknown
Shell mode:	all	**Default:**	on (interactive), off (non-interactive)

If set, aliases are expanded. This option is enabled by default for interactive shells.

extdebug

Option:	extdebug	**Since:**	3.0-alpha
Shell mode:	all	**Default:**	off

If set, behavior intended for use by debuggers is enabled.

extglob

Option:	extglob	**Since:**	2.02-alpha1
Shell mode:	all	**Default:**	off

If set, the extended pattern matching features are enabled. See the important note below under Parser configurations.

extquote

Option:	extquote	**Since:**	3.0-alpha (?)
Shell mode:	all	**Default:**	on

If set, `$'string'` and `$"string"` quoting is performed within parameter expansions enclosed in double quotes. See the important note below under Parser configurations.

failglob

Option:	failglob	Since:	3.0-alpha
Shell mode:	all	Default:	off

If set, patterns which fail to match filenames during pathname expansion result in an error message.

force_fignore

Option:	force_fignore	Since:	3.0-alpha
Shell mode:	interactive (?)	Default:	on

If set, the suffixes specified by the FIGNORE shell variable cause words to be ignored when performing word completion even if the ignored words are the only possible completions. This option is enabled by default.

globasciiranges

Option:	globasciiranges	Since:	4.3-alpha
Shell mode:	all	Default:	off

If set, range expressions used in pattern matching behave as if in the traditional C locale when performing comparisons. That is, the current locale's collating sequence is not taken into account, so b will not collate between A and B, and upper-case and lower-case ASCII (American Standard Code for Information Interchange) characters will collate together.

globstar

Option:	globstar	Since:	4.0-alpha
Shell mode:	all	Default:	off

If set, recursive globbing with `**` is enabled.

gnu_errfmt

Option:	gnu_errfmt	Since:	3.0-alpha
Shell mode:	all	Default:	off

If set, shell error messages are written in the "standard GNU error message format".

histappend

Option:	histappend	Since:	unknown
Shell mode:	interactive (?)	Default:	off

If set, the history list is appended to the file named by the value of the HISTFILE variable when the shell exits, rather than overwriting the file.

histreedit

Option:	histreedit	Since:	unknown
Shell mode:	interactive (?)	Default:	off

If set, and readline is being used, a user is given the opportunity to re-edit a failed history substitution.

histverify

Option:	histverify	Since:	unknown
Shell mode:	interactive (?)	Default:	off

Allow to review a history substitution result by loading the resulting line into the editing buffer, rather than directly executing it.

hostcomplete

Option:	hostcomplete	Since:	2.0-alpha3
Shell mode:	interactive (?)	Default:	on

If set, Bash completion also completes hostnames. On by default.

huponexit

Option:	huponexit	Since:	2.02-alpha1
Shell mode:	interactive login	Default:	off

If set, Bash will send the `SIGHUP` signal to all jobs when an interactive login shell exits.

interactive_comments

Option:	interactive_comments	Since:	unknown
Shell mode:	interactive	Default:	on

Allow commenting in interactive shells, on by default.

lastpipe

Option:	lastpipe	Since:	4.2-alpha
Shell mode:	all	Default:	off

If set, **and job control is not active**, the shell runs the last command of a pipeline not executed in the background in the current shell environment.

lithist

Option:	lithist	Since:	unknown
Shell mode:	interactive	Default:	off

If set, and the cmdhist option is enabled, multi-line commands are saved to the history with embedded newlines rather than using semicolon separators where possible.

login_shell

Option:	login_shell	Since:	2.05a-alpha1
Shell mode:	all	Default:	n/a

The option is set when Bash is a login shell. This is a readonly option.

mailwarn

Option:	mailwarn	Since:	unknown
Shell mode:	interactive (?)	Default:	off

If set, and a file that Bash is checking for mail has been accessed since the last time it was checked, the message "The mail in mailfile has been read" is displayed.

no_empty_cmd_completion

Option:	mailwarn	Since:	unknown
Shell mode:	interactive (?)	Default:	off

If set, and readline is being used, Bash will not attempt to search the PATH for possible completions when completion is attempted on an empty line.

nocaseglob

Option:	nocaseglob	**Since:**	2.02-alpha1
Shell mode:	interactive (?)	**Default:**	off

If set, Bash matches filenames in a case-insensitive fashion when performing pathname expansion.

nocasematch

Option:	nocasematch	**Since:**	3.1-alpha1
Shell mode:	interactive (?)	**Default:**	off

If set, Bash matches patterns in a case-insensitive fashion when performing matching while executing `case` or `[[` conditional commands.

nullglob

Option:	nullglob	**Since:**	unknown
Shell mode:	interactive (?)	**Default:**	off

If set, Bash allows patterns which match no files to expand to a null string, rather than themselves.

progcomp

Option:	progcomp	**Since:**	2.04-alpha1
Shell mode:	interactive (?)	**Default:**	on

If set, the programmable completion facilities are enabled. This option is enabled by default.

promptvars

Option:	promptvars	**Since:**	unknown
Shell mode:	interactive (?)	**Default:**	on

If set, prompt strings undergo parameter expansion, command substitution, arithmetic expansion, and quote removal after being expanded using the prompt special sequences. This option is enabled by default.

restricted_shell

Option:	restricted_shell	**Since:**	2.03-alpha
Shell mode:	interactive (?)	**Default:**	off

The option is set when Bash is a restricted shell. This is a readonly option.

shift_verbose

Option:	shift_verbose	Since:	unknown
Shell mode:	interactive (?)	Default:	off

If set, the shift builtin prints an error message when the shift count exceeds the number of positional parameters.

sourcepath

Option:	sourcepath	Since:	unknown
Shell mode:	interactive (?)	Default:	on

If set, the source builtin command uses the value of PATH to find the directory containing the file supplied as an argument. This option is enabled by default.

xpg_echo

Option:	xpg_echo	Since:	2.04-beta1
Shell mode:	interactive (?)	Default:	off

If set, the `echo` -builtin command expands backslash-escape sequences by default (POSIX, SUS, XPG).

Parser configurations

Parser configurations change the way the Bash parser recognizes the syntax when parsing a line. This, of course, is impossible for a line that already was parsed.

There are two options that influence the parsing this way:

- `extglob`
- `extquote`

Consequence: You **can't** use the new syntax (e.g. the extended globbing syntax) and the command to enable it **in the same line**.

```
$ shopt -s extglob; echo !(*.txt) # this is the WRONG way!
-bash: syntax error near unexpected token `('
```

You have to configure the parser **before** a line with new syntax is parsed:

```
$ shopt -s extglob # standalone - CORRECT way!
$ echo !(*.txt)
...
```

See also

- Internal: shopt builtin command
- Internal: set builtin command

💬 Discussion

Juanma, 2015/04/17 14:21 (2015-04-17T14:21:33Z)

The `compat*` variables are mentioned but not documented. My guess is that turning one of them on brings Bash to the behavior expected in that version. Am I right? I'm wondering because I can't make the regexp operator (`=~`) work as I expect, which means **not** doing this:

```
# re='[[:blank:]]*'; if [[ 'z' =~ $re ]]; then echo "match!"; fi
match!
```

or even this:

```
# re='[ ]*'; if [[ 'z' =~ $re ]]; then echo "match!"; fi
match!
```

Thanks to anyone willing to help.

Levi, 2015/08/21 12:07 (2015-08-21T12:07:48Z)

You're matching the null string in front of and behind the 'z'. To get it to work properly, try:

```
$ re='^[[:blank:]]*$'; if [[ 'z' =~ $re ]]; then echo "match!"; fi
$ re='^[ ]*$'; if [[ 'z' =~ $re ]]; then echo "match!"; fi</pre>
```

Now the anchors will match the nothingness and the rest will have to match whatever's in between.

There's a very similar case in the camel book, which comes with a footnote saying:
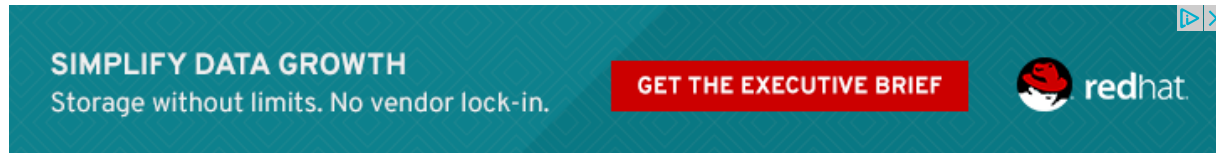
Don't feel bad. Even the authors get caught by this from time to time.

You could leave a comment if you were logged in.

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