Vim-Plug-in

bash-support.vim

Version 5.0alpha

Hot keys

Key mappings for Vim/gVim/Neovim.

`https://vim.sourceforge.io` — **Wolfgang Mehner**, `wolfgang-mehner@web.de`

H*elp*

`\he`	English dictionary	
`\hb`	display the Bash manual	
`\hh`	help (Bash builtins)	(T)
`\hm`	show manual (cmd. line utilities)	
`\hp`	help (plug-in)	

B*ash*

`\bps`	**p**arameter **s**ubstitution (list)	(T)
`\bsp`	**s**pecial **p**arameters (list)	(T)
`\ben`	**en**vironment (list)	(T)
`\bbu`	**bu**iltins (list)	(T)
`\bse`	**se**t options (list)	(T)
`\bso`	**sho**pts (list)	(T)

C*omments*

[n]`\cl`	end-of-line comment	(v)
[n]`\cj`	adjust end-of-line comments	(v)
`\cs`	set end-of-line comment col.	
[n]`\cc`	code → comment	(v)
[n]`\co`	uncomment code	(v)
`\cfr`	frame comment	
`\cfu`	function description	
`\ch`	file header	
`\csh`	shebang	
`\cd`	date	
`\ct`	date & time	

C*omments (cont)*

`\css`	script sections	(T)
`\ckc`	keyword comments	(T)
`\cma`	plug-in macros	(T)
`\ce`	`echo "`*actual line*`"`	(v)
`\cr`	remove echo from actual line	(v)

S*tatements*

`\sc`	`case in ... esac`	
`\sei`	`elif then`	
`\sf`	`for in do done`	(s)
`\sfo`	`for ((...)) do done`	(s)
`\si`	`if then fi`	(s)
`\sie`	`if then else fi`	(s)
`\ss`	`select in do done`	(s)
`\su`	`until do done`	(s)
`\sw`	`while do done`	(s)
`\sfu`	`function`	(s)
`\se`	`echo -e ""`	(s)
`\sp`	`printf "%s"`	(s)
`\sae`	array element `${.[.]}`	(s)
`\saa`	arr. elem.s (all) `${.[@]}`	(s)
`\sas`	arr. elem.s (1 word) `${.[*]}`	(s)
`\ssa`	subarray `${.[@]::}`	(s)
`\san`	no. of arr. elem.s `${.[@]}`	(s)
`\sai`	list of indices `${.[*]}`	(s)

T*ests*

`\ta`	arithmetic tests	(T)
`\tfp`	file permissions	(T)
`\tft`	file types	(T)
`\tfc`	file characteristics	(T)
`\ts`	string comparisons	(T)
`\toe`	option is enabled	
`\tvs`	variables has been set	
`\tfd`	file descr. refers to a terminal	
`\tm`	string matches regexp	

IO*-Redirection*

`\ior`	IO-redirections (list)	(T)
`\ioh`	here-document	(s)

P*attern Matching*

`\pzo`	zero or one, `?( | )`	(s)
`\pzm`	zero or more, `*( | )`	(s)
`\pom`	one or more, `+( | )`	(s)
`\peo`	exactly one, `@( | )`	(s)
`\pae`	anything except, `!( | )`	(s)
`\ppc`	POSIX classes	(T)
`\pbr`	`${BASH_REMATCH[0...3]}`	(T)

S*n****ippets*

`\nr`	read code snippet	
`\nv`	view code snippet	
`\nw`	write code snippet	(v)
`\ne`	edit code snippet	
`\ntl`	edit local templates	
`\ntc`	edit custom templates	
`\ntp`	edit personal templates	
`\ntr`	reread the templates	
`\ntw`	template setup wizard	
`\nts`	choose style	(T)

all hotkeys work in normal and insert mode
visual mode: (v) use the range, (s) surround range
tab-completion: (T) specialized, (F) filenames

		***R**un*
`\rr`	run script	(v)
`\rsa`	set script cmd. line arguments	(F)
`\ria`	set interp. cmd. line arguments	(F)
`\rk`	check syntax	
`\rso`	syntax check options	
`\ro`	change output destination	(T)
`\rd`	set "direct run"	(T)
`\rse`	set shell executable	(T)
`\rx`	set xterm size	
`\re`	make script executable/not exec.	
`\rh`	hardcopy buffer	(v)
`\rs`	plug-in settings	
`\ra`	set script cmd. line arguments	(*d*)
`\rba`	set interp. cmd. line arguments	(*d*)
`\rc`	check syntax	(*d*)
`\rco`	syntax check options	(*d*)

		Buffer "Bash Output"
`\qf`	load errors into quickfix	
`\qj`	load into qf. and jump to first error	

(*d*) deprecated, will be removed in next version

Run

Run *bash* with optional arguments:

`:Bash [<args>]`

Memorize script arguments:

`:BashScriptArguments [<args>]`

The memorize arguments are used if `:Bash` is invoked without arguments.

Set interpreter arguments:

`:BashInterpArguments [<args>]`

Syntax

Run syntax checking:

`:BashCheck`

Set syntax checking options:

`:BashSyntaxCheckOptions <opts>`

Options

Set the output method:

`:BashOutputMethod [<method>]`

Set the shell executable:

`:BashExecutable [<exec>]`

Set "direct run":

`:BashDirectRun [ yes | no ]`

When set to `yes`, executable scripts will be run as the executable, instead of using `:BashExecutable`.

BashDB

Run the debugger `bashdb`:

`:BashDB`

. . .