

# Vim Commands Cheat Sheet

## How to Exit

:q[uit]	Quit Vim. This fails when changes have been made.
:q[uit]!	Quit without writing.
:cq[uit]	Quit always, without writing.
:wq	Write the current file and exit.
:wq!	Write the current file and exit always.
:wq {file}	Write to {file}. Exit if not editing the last
:wq! {file}	Write to {file} and exit always.
:[range]wq[!]	[file] Same as above, but only write the lines in [range].
ZZ	Write current file, if modified, and exit.
ZQ	Quit current file and exit (same as ":q!").

## Editing a File

:e[dit]	Edit the current file. This is useful to re-edit the current file, when it has been changed outside of Vim.
:e[dit]!	Edit the current file always. Discard any changes to the current buffer. This is useful if you want to start all over again.
:e[dit] {file}	Edit {file}.
:e[dit]! {file}	Edit {file} always. Discard any changes to the current buffer.
gf	Edit the file whose name is under or after the cursor. Mnemonic: "goto file".

## Inserting Text

a	Append text after the cursor [count] times.
A	Append text at the end of the line [count] times.
i	Insert text before the cursor [count] times.
I	Insert text before the first non-blank in the line [count] times.
gl	Insert text in column 1 [count] times.
o	Begin a new line below the cursor and insert text, repeat [count] times.
O	Begin a new line above the cursor and insert text, repeat [count] times.

## Inserting a file

:r[ead] [name]	Insert the file [name] below the cursor.
:r[ead] !{cmd}	Execute {cmd} and insert its standard output below the cursor.

## Deleting Text

<Del> or x	Delete [count] characters under and after the cursor
X	Delete [count] characters before the cursor
d{motion}	Delete text that {motion} moves over
dd	Delete [count] lines
D	Delete the characters under the cursor until the end of the line
{Visual}x or {Visual}d	Delete the highlighted text (for {Visual} see <a href="#">Selecting Text</a> ).
{Visual}CTRL-H or {Visual}	When in Select mode: Delete the highlighted text
{Visual}X or {Visual}D	Delete the highlighted lines
:[range]d[elete]	Delete [range] lines (default: current line)
:[range]d[elete] {count}	Delete {count} lines, starting with [range]

## Changing (or Replacing) Text

r{char}	replace the character under the cursor with {char}.
R	Enter Insert mode, replacing characters rather than inserting
~	Switch case of the character under the cursor and move the cursor to the right. If a [count] is given, do that many characters.
~{motion}	switch case of {motion} text.
{Visual}~	Switch case of highlighted text

## Substituting

:[range]s[substitute]/{pattern}/{string}/[c][e][g][p][r][i][l] [count]	For each line in [range] replace a match of {pattern} with {string}.
---	--

```
:[range]s[substitute] [c][e][g][r][i][I] [count]
:[range]&[c][e][g][r][i][I] [count]
```

Repeat last :substitute with same search pattern and substitute string, but without the same flags. You may add extra flags

The arguments that you can use for the substitute commands:

- [c] Confirm each substitution. Vim positions the cursor on the matching string. You can type:
  - 'y' to substitute this match
  - 'n' to skip this match
  - to skip this match
  - 'a' to substitute this and all remaining matches {not in Vi}
  - 'q' to quit substituting {not in Vi}
  - CTRL-E to scroll the screen up {not in Vi}
  - CTRL-Y to scroll the screen down {not in Vi}.
- [e] When the search pattern fails, do not issue an error message and, in particular, continue in maps as if no error occurred.
- [g] Replace all occurrences in the line. Without this argument, replacement occurs only for the first occurrence in each line.
- [i] Ignore case for the pattern.
- [I] Don't ignore case for the pattern.
- [p] Print the line containing the last substitute.

## Copying and Moving Text

"{a-zA-Z0-9.%#:-}"	Use register {a-zA-Z0-9.%#:-} for next delete, yank or put (use uppercase character to append with delete and yank) ({.%#:-} only work with put).
:reg[isters]	Display the contents of all numbered and named registers.
:reg[isters] {arg}	Display the contents of the numbered and named registers that are mentioned in {arg}.
:di[splay] [arg]	Same as :registers.
["x]y{motion}	Yank {motion} text [into register x].
["x]yy	Yank [count] lines [into register x]
["x]Y	yank [count] lines [into register x] (synonym for yy).
{Visual}["x]y	Yank the highlighted text [into register x] (for {Visual} see <a href="#">Selecting Text</a> ).
{Visual}["x]Y	Yank the highlighted lines [into register x]
:[range]y[ank] [x]	Yank [range] lines [into register x].
:[range]y[ank] [x] {count}	Yank {count} lines, starting with last line number in [range] (default: current line), [into register x].
["x]p	Put the text [from register x] after the cursor [count] times.
["x]P	Put the text [from register x] before the cursor [count] times.
["x]gp	Just like "p", but leave the cursor just after the new text.
["x]gP	Just like "P", but leave the cursor just after the new text.
:[line]pu[t] [x]	Put the text [from register x] after [line] (default current line).

`:[line]pu[t]! [x]` Put the text [from register x] before [line] (default current line).

## Undo/Redo/Repeat

<code>u</code>	Undo [count] changes.
<code>:u[ndo]</code>	Undo one change.
<code>CTRL-R</code>	Redo [count] changes which were undone.
<code>:red[o]</code>	Redo one change which was undone.
<code>U</code>	Undo all latest changes on one line. {Vi: while not moved off of it}
<code>.</code>	Repeat last change, with count replaced with [count].

## Moving Around

Basic motion commands:

```

    k
h   l
    j
    
```

<code>h</code> or	[count] characters to the left (exclusive).
<code>l</code> or <code>or</code>	[count] characters to the right (exclusive).
<code>k</code> or <code>or</code> <code>CTRL-P</code>	[count] lines upward
<code>j</code> or <code>or</code> <code>CTRL-J</code> or <code>or</code> <code>CTRL-N</code>	[count] lines downward (linewise).
<code>0</code>	To the first character of the line (exclusive).
<code>&lt;Home&gt;</code>	To the first character of the line (exclusive).
<code>^</code>	To the first non-blank character of the line
<code>\$</code> or <code>&lt;End&gt;</code>	To the end of the line and [count - 1] lines downward
<code>g0</code> or <code>g&lt;Home&gt;</code>	When lines wrap ('wrap on): To the first character of the screen line (exclusive). Differs from "0" when a line is wider than the screen. When lines don't wrap ('wrap' off): To the leftmost character of the current line that is on the screen. Differs from "0" when the first character of the line is not on the screen.
<code>g^</code>	When lines wrap ('wrap' on): To the first non-blank character of the screen line (exclusive). Differs from "^" when a line is wider than the screen. When lines don't wrap ('wrap' off): To the leftmost non-blank character of the current line

	that is on the screen. Differs from "^" when the first non-blank character of the line is not on the screen.
g\$ or g<End&gr;	When lines wrap ('wrap' on): To the last character of the screen line and [count - 1] screen lines downward (inclusive). Differs from "\$" when a line is wider than the screen. When lines don't wrap ('wrap' off): To the rightmost character of the current line that is visible on the screen. Differs from "\$" when the last character of the line is not on the screen or when a count is used.
f{char}	To [count]'th occurrence of {char} to the right. The cursor is placed on {char} (inclusive).
F{char}	To the [count]'th occurrence of {char} to the left. The cursor is placed on {char} (inclusive).
t{char}	Till before [count]'th occurrence of {char} to the right. The cursor is placed on the character left of {char} (inclusive).
T{char}	Till after [count]'th occurrence of {char} to the left. The cursor is placed on the character right of {char} (inclusive).
;	Repeat latest f, t, F or T [count] times.
,	Repeat latest f, t, F or T in opposite direction [count] times.
- <minus>	[count] lines upward, on the first non-blank character (linewise).
+ or CTRL-M or <CR>	[count] lines downward, on the first non-blank character (linewise).
_ <underscore>	[count] - 1 lines downward, on the first non-blank character (linewise).
<C-End> or G	Goto line [count], default last line, on the first non-blank character.
<C-Home> or gg	Goto line [count], default first line, on the first non-blank character.
<S-Right> or w	[count] words forward
<C-Right> or W	[count] WORDS forward
e	Forward to the end of word [count]
E	Forward to the end of WORD [count]
<S-Left> or b	[count] words backward
<C-Left> or B	[count] WORDS backward
ge	Backward to the end of word [count]
gE	Backward to the end of WORD [count]

These commands move over words or WORDS.

A word consists of a sequence of letters, digits and underscores, or a sequence of other non-blank characters, separated with white space (spaces, tabs, ). This can be changed with the 'iskeyword' option.

A WORD consists of a sequence of non-blank characters, separated with white space. An empty line is also considered to be a word and a WORD.

(	[count] sentences backward
)	[count] sentences forward
{	[count] paragraphs backward
}	[count] paragraphs forward
[[	[count] sections forward or to the next '{' in the first column. When used after an operator, then the '}' in the first column.
]]	[count] sections forward or to the next '}' in the first column
[[	[count] sections backward or to the previous '{' in the first column
]]	[count] sections backward or to the previous '}' in the first column

## Marks

m{a-zA-Z}	Set mark {a-zA-Z} at cursor position (does not move the cursor, this is not a motion command).
m' or m`	Set the previous context mark. This can be jumped to with the "" or "" command (does not move the cursor, this is not a motion command).
:[range]ma[rk] {a-zA-Z}	Set mark {a-zA-Z} at last line number in [range], column 0. Default is cursor line.
:[range]k{a-zA-Z}	Same as :mark, but the space before the mark name can be omitted.
'{a-z}	To the first non-blank character on the line with mark {a-z} (linewise).
'{A-Z0-9}	To the first non-blank character on the line with mark {A-Z0-9} in the correct file
`{a-z}	To the mark {a-z}
`{A-Z0-9}	To the mark {A-Z0-9} in the correct file
:marks	List all the current marks (not a motion command).
:marks {arg}	List the marks that are mentioned in {arg} (not a motion command). For example:

## Searching

/[pattern]/[/]	Search forward for the [count]'th occurrence of {pattern}
/[pattern]/{offset}	Search forward for the [count]'th occurrence of {pattern} and go {offset} lines up or down.
/ <cr&gt;< td=""> <td>Search forward for the [count]'th latest used pattern</td> </cr&gt;<>	Search forward for the [count]'th latest used pattern
//{offset}\CR	Search forward for the [count]'th latest used pattern with new. If {offset} is

/[offset]pattern	empty no offset is used.
?{pattern}[?] ]<CR>	Search backward for the [count]'th previous occurrence of {pattern}
?{pattern}? {offset}<CR>	Search backward for the [count]'th previous occurrence of {pattern} and go {offset} lines up or down
?<CR>	Search backward for the [count]'th latest used pattern
??{offset}<CR>	Search backward for the [count]'th latest used pattern with new {offset}. If {offset} is empty no offset is used.
n	Repeat the latest "/" or "?" [count] times.
N	Repeat the latest "/" or "?" [count] times in opposite direction.

## Selecting Text (Visual Mode)

To select text, enter visual mode with one of the commands below, and use [motion commands](#) to highlight the text you are interested in. Then, use some command on the text.

The operators that can be used are:

```

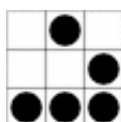
~  switch case
d  delete
c  change
y  yank
>  shift right
<  shift left
!  filter through external command
=  filter through 'equalprg' option command
gq  format lines to 'textwidth' length

```

v	start Visual mode per character.
V	start Visual mode linewise.
<Esc>	exit Visual mode without making any changes

## How to Suspend

CTRL-Z	Suspend Vim, like ":stop". Works in Normal and in Visual mode. In Insert and Command-line mode, the CTRL-Z is inserted as a normal character.
:sus[pend][!] or :st[op][!]	Suspend Vim. If the '!' is not given and 'autowrite' is set, every buffer with changes and a file name is written out. If the '!' is given or 'autowrite' is not set, changed buffers are not written, don't forget to bring Vim back to the foreground later!



*Daniel Gryniewicz* / [dang@fprintf.net](mailto:dang@fprintf.net)