

Up-Down motions

k	<i>count</i> lines upward
j	<i>count</i> lines downward
gk	<i>count</i> display lines upward
gj	<i>count</i> display lines downward
-	<i>count</i> lines upward on the first non-blank char
+	<i>count</i> lines downward on the first non-blank char
_	<i>count</i> - 1 lines downward on the first non-blank char
G	goto line <i>count</i> , default last line
gg	goto line <i>count</i> , default first line
: <i>[range]</i>	set cursor on last line number in <i>range</i>
{ <i>count</i> }%	go to <i>count</i> percentage in file

Left-right motions

h	<i>count</i> chars to the left
l	<i>count</i> chars to the right
0	to the first char of the line
^	to the first non-blank char of the line
\$	to the end of the line
g_	to the last non-blank char of the line and <i>count</i> - 1 lines downward
g0	wrap: to the first char of the screen line
g^	rwap: to the first non-blank char of the screen line
gm	like g0, but half a screenwidth to the right
g\$	wrap: to the last char of the screen line
	to screen column <i>count</i> in current line
f <i>char</i>	to <i>count</i> occurrence of <i>char</i> to the right
F <i>char</i>	to <i>count</i> occurrence of <i>char</i> to the left

Left-right motions (cont)

t <i>char</i>	till before <i>count</i> occurrence of <i>char</i> to the right
T <i>char</i>	till after <i>count</i> occurrence of <i>char</i> to the left
;	repeat latest f, t, F, T <i>count</i> times
,	repeat latest f, t, F, T <i>count</i> times in opposite direction

Scrolling

CTRL-U	scroll down half a screen of text
CTRL-D	scroll up half a screen of text
CTRL-E	scroll one line up
CTRL-Y	scroll one line down
CTRL-F	scroll forward a whole screen
CTRL-B	scroll backward a whole screen
zz	scroll to see the context of the line
zt	put the cursor line at the top
zb	put the cursor line at the bottom

Word motions

w	move forward <i>count</i> words
W	move forward <i>count</i> WORDS
b	move backward <i>count</i> words
B	move backward <i>count</i> WORDS
e	move to the next end of a word
E	move to next end of a WORD
ge	move to the previous end of a word
gE	move to the previous end of a WORD

a WORD is white-space separated word.

Operators and motions

dd	delete whole line
x or dl	delete char under the cursor
X or dh	delete char left of the cursor
D or d\$	delete to the end of the line
C or c\$	change to the end of the line
s or cl	change one char
S or cc	change whole line excluding indentation
r	command to replace char under cursor
.	repeat last change

Windows and Panes

CTRL-W +	increase current window height by N
CTRL-W -	decrease current window height by N
CTRL-W =	evensize all windows
CTRL-W <	decrease current window width by N
CTRL-W >	increase current window width by N

Jumps

CTRL-O	Go to [count] Older cursor position in jump list
CTRL-I	go to [count] newer cursor position in jump list
:ju[mps]	print the jump list
:cle[arjumps]	clear the jump list of the current window
g;	go to <i>count</i> older position in change list
g,	go to <i>count</i> newer position in change list
:changes	print the change list

Jumps (cont)

%	find the next item in this line and jump to its match
((go to <i>count</i> previous unmatched (
))	go to <i>count</i> next unmatched)
{{	go to count previous unmatched {
}}	go to count next unmatched }
]m	go to count next start of a method
]M	go to count next end of a Method
[m	go to count previous start of a method
[M	go to count previous end of a method
H	move to first visible line
M	move to middle visible line
L	move to last visible line

The following commands are "jump" commands: "", "", "G", "/", "?", "n", "N", "-", "%", "(", ")", "[", "]", "{", "}", ":s", ":tag", "L", "-", "M", "H" and the commands that start editing a new file.

Using marks

`` or "	to the position of the latest jump
CTRL-O	jump to older position
CTRL-I	jump back to newer position
:jumps	Give list of positions of jumps
mchar	mark the place under cursor with mark <i>char</i>
`{a-z} or '{a-z}	move to the mark <i>char</i> in current buffer
`{A-Z0-9} or '{A-Z0-9}	move to the mark <i>char</i> in other file
`char	move to the mark <i>char</i>
'char	move to first non-blank char of line with mark <i>char</i>
:marks	get a list of marks

Using marks (cont)

"	cursor position when last editing the file
[start of the last change
]	end of the last change
:delm[-arks]	Delete the specified <i>marks</i> { <i>marks</i> }
:delm!	Delete all marks for current buffer
"" or ``	to the position when last exiting the current buffer
^ or ^	to the position where the cursor was the last time when Insert mode was stopped
`. or !	to the position of the last change
'(or `(to the start of the current sentence
') or `)	to the end of the current sentence
'{ or `{	to the start of the current paragraph

When making jumps to positions further than within the same line, vim remembers the position before the jump and sets a mark.

Search mode

/	activate forward search mode
?	activate backward search mode
n	continue search forward
N	continue search backward
<UP>	Search through history upwards
<DOWN>	Search through history downwards
*	search <i>count</i> words under cursor forward
#	search <i>count</i> word under cursor backward
\>	match the end of a word
\<	match the beginning of a word

Visual mode

v	start visual mode
V	visual mode on whole lines
CTRL-V	visual block mode
o	move to the other end of selection
O	in block mode: move to the other corner of the same line

Operators

c	change
d	delete
y	yank into register
~	swap case under cursor
g~[<i>motion</i>]	swap case within <i>motion</i>
gu[<i>motion</i>]	make lowercase within <i>motion</i>
gU[<i>motion</i>]	make uppercase within <i>motion</i>
!	filter through external program
g?	ROT-13 encoding
>	shift right
<	shift left
zf	define a fold
g@	call function set with the 'operatorfunc' option

Moving text

p	paste back last deleted text
P	past back last deleted text before the cursor
<i>number</i> p	like p but <i>number</i> times
<i>number</i> P	like P but <i>number</i> times
xp	swap the next char with the char under cursor
y	yank char under cursor
yw	yank word
Y or yy	yank whole line
*yy	yank whole line to clipboard

Moving text (cont)

*p put text from the clipboard back

Text object selection

aw a word

iw inner word

aW a WORD

as a sentence includes white space after the sentence

ap a paragraph

a] or a[a [] block

a) or a(or ab a () block

a< or a> a <> block

at a tag block

a} or a{ or aB a {} block

a" or a' or a` a quoted string

a blocks may be inner blocks with **i** instead of **a**.

These commands can only be used while in **Visual mode** or after an **operator**

Text object motions

(*count* sentences backward

) *count* sentences forward

{ *count* paragraphs backward

} *count* paragraphs forward

]] *count* sections forward or to the next { in the first column

[[*count* sections backward or to the previous { 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

