Vim Notes

Saurabh Barjatiya

26 June, 2007

Contents

1 General

1.1 Introduction

- If one creates a *'.vimrc'* file than vim does not enables vi-compatibility mode by default during startup. We can also do this by using command *'set :nocompatible'*.

- Create a *'.vimrc'* file with following contents to automatically indent c, c++ or java file: (case-sensitive)

```
set shiftwidth=4
autocmd FileType c,cpp,java set cindent
```

- Vim is a modal editor. It is always in one of three modes:
  - Normal
  - Visual
  - Insert

- Most commands can be precedded by a number to repeat it many times. Like to delete 5 charaters one can use '`xxxxx`' or '`5x`'.

2 Shortcut

2.1 Movement

Shortcut	Description
h	To move left one character in normal mode.
j	To move down one line in normal mode.
k	To move up one line in normal mode.
l	To move right one character in normal mode.
<Left>	To move one character left.
<Down>	To move one character down.
<Up>	To move one character up.
<Right>	To move one character right.
w	To move one word right.
b	To move one word left.
$	To move to end of line.
<End>	To move to end of line.
<kEnd>	To move to end of line. ('k' here is for keypad.)
^	To move to first non-blank character on this line.
0	To move to first character on this line.
<Home>	To move to first character on this line.
<kHome>	To move to first character on this line.
<number>G	Move to given line number.
G	Move to last line of file.

2.2 Search

Shortcut	Description
f<character>	Forward search given character from current position. Search ends with cursor pointing to character.
F<character>	Backward search given character from current position. Search ends with cursor pointing to character.
t<character>	Forward search till given character. Search ends with cursor pointing to character before searched character.
T<character>	Backward search till given character. Search ends with cursor pointing to character right to searched character.

2.3 Editing

Shortcut	Description
`x`	To delete one character under the cursor in normal mode.
`u`	Undo last action.
`U`	Undo last action on last line that was edited. Using it twice will redo the changes done to last line or in order words undo the last *'U'*.
`Ctrl+R`	Redo last change or cancel last *'u'*.
`dd`	To delete current line and copy it into buffer.
`o`	To insert a new line below current line and start insert mode.
`O`	To insert a new line above current line and start insert mode.

2.4 Window Management

Shortcut	Description
Ctrl+w, {Arrow}	To switch between open windows. Here arrow in the direction in which you want to move.
Ctrl+w, s	To split current window into two windows horizontally
Ctrl+w, v	To split current window into two windows horizontally
Ctrl+w, Ctrl+v	Same
Ctrl+w, n	To split window and start editing in new buffer
Ctrl+w, Ctrl+n	Same
Ctrl+w, q	To quit current window

2.5 Jumping

Shortcut	Description
Crtl+j	To jump to given hyperlink
Ctrl+t	To jump back from where we jumped. Like all jumps are stored on stack. This is like pop operation.
Ctrl+o	Jump to old location in last file. Can be reused to go as back as possible.
Ctrl+i	To undo coming back by Ctrl+o and move forward instead.
Ctrl+^	To move to alternate file. Like alt on remote.
‘.	To move to place where we made last change in file.
‘¨	To jump to line we where last on while browsing the file.

2.6 Other

Shortcut	Description
i	To start insert mode from normal mode. Insertion starts from current position.
<Esc>	To go back to normal mode from insert mode.
ZZ	Save and exit vim.
a	To start insert mode from normal mode. Insertion starts after current position.
Ctrl-g	Show status information on position and editing.
v	To go into visual mode. In visual mode one can use arrow keys to select. Then use 'd' to delete or 'y' to yank and so on.
Ctrl+v	To enable block selection in visual mode.

3 Command

3.1 Window Management

Command	Description
:sp	To split current window into two window horizontally
:split	Same
:vs	To split current window into two window vertically
:vsplit	Same
:new	To split current window horizontally and start editing in new window
:vne	To split current window vertically and start editing in new window
:q	To quit current window
:b N	To jump to buffer N
:b {name}	To junp to buffer in which file of given name is open
:sp file	To split window and start editing given file in new window
:sv file	To split window and start editing given file in new window. Also set readonly option for this buffer

3.2 File Management

Command	Description
:files	To see list of open buffers
:buffers	To see list of open buffers
:edit! file	To discard changes done in current buffer and start editing given file
:hide edit file	To hide current buffer and start editing given file. Changes made in current buffer are neither saved nor abandoned.
:next	To go to next file
:n	Same
:wnext	To go to next file and save changes in current file.
:wn	Same
:next!	To abandon changes done in current buffer and move to next file
:args	To list files opened using command line arguments
:previous	To go to previous file
:wprevioud	To save changes and go to previous file
:first	To go to first file
:last	To go to last file

Note: :Nprevious and :Nnext can be used to move N files back and forward.

3.3 Option

Command	Description
:set number	Display line numbers before each line.
:set nonumber	Do not display line numbers.
:set statusline=line	To set custom status line.
:set statusline=	To remove custom status line.
:set fileformat=x	To set file format as unix or dos.
:set fileformats	To see list of available formats.
:set mouse=a	To enable mouse

3.4 Window related Option

Command	Description
:set winwidth=N	To set minimum width of current window.
:set winheight=N	To set minimum height of current window.
:set readonly	To make current window readonly.
:set ro	Same
:set noreadonly	To remove readonly flag from current window
:set noro	Same
:set modifiable	To make window modifiable.
:set ma	Same
:set nomodifiable	To disable editing of current window
:set noma	Same

Note: Readonly buffers can be modified. Only while editing it generates a warning. Also while saving it will generate warning and cannot be saved with ':w'. One must use ':w!' to save a modified readonly buffer.

3.5 Files related Option

Command	Description
:set autowrite	To enable autosave while movin between multiple files. This will stop warning messages while moving using ':n', ':previous', etc.
:args files	To set argument list to files.
:set backup	To make vim create backup of old file during save.
:set backupext=ext	To change backup extension from ' ' to ext.
:set backupdir=dir	To keep backups in specified directory rather than in files original directory.
:set patchmode=.orig	To keep first backup of original file with extension .orig. This backup will not be overwritten in subsequent saves.
:write >> name	To append text in current buffer to end of specified file.
:saveas name	To save contents of buffer with given named file.
:file name	To set name of current file. The contents of buffer will be saved with the give name file with next save command. No changes are done to given name file until next save.

Using ':q' while some files are left unedited, vim will produce warning N

more files to edit. Use ':q' just next to it to ignore warning and quit. One can always use ':q!' to ignore all warnings and quit.

3.6 Spell check commands

Command	Description
:setlocal spell spelllang=en_us	To enable spell checker for us english for current file
zg	Add word as good word to global spell file. Changes are saved.
zw	Mark word as wrong word in global spell file. Changes are saved.
zG	Mark word as good word in local buffer
zW	Mark word as bad word in local buffer
z=	Suggest corrections
:spellr[epall]	Repeat the replacement done by z= for all matches with the replaced word in the current window
]s	Move to next misspelled word
[s	Move to previous misspelled word
]S	Like ']s' but only stops at bad words and not on word of another region or rare words
[S	Like ']S' backwards.
zug	Undo zg
zuw	Undo zw
zuG	Undo zG
zuW	Undo zW
:spe[llgood] word	To add word as good word in spell file.
:spe[llgood]! word	To add word as good word in internal word list which is temporary.
:spellw[rong] word	To add word as bad word to spell file.
:spellw[rong]! word	To add word as bad word to internal file.

Important points:

- In insert mode use 'Ctrl+X' at end of word for suggestions on current word. It is like autocomplete. Use 'Ctrl+N' for next suggestion and 'Ctrl+P' for previous suggestion. 'Ctrl+F' for autocompletion of file names.
- 'spellsuggest' can be used to control how these suggestions work.
- 'spellcapcheck' option is used to check for capitalization errors.

3.7 Generic Command

Command	Description
:N	Jump to given line number
:q	Quit
:q!	Quit and discard changes.
:wq	Save and exit.

4 Command line arguments

Switch	Description
-o	To open list of files supplied on command line in horizontal windows
-oN	To open list of files supplied on command line in horizontal windows, maxmimum N windows to create
-O	To open list of files supplied on command line in vertical windows
-ON	To open list of files supplied on command line in vertical windows, maxmimum N windows to create
-R	To open all files in readonly mode for files opened using command line arguments.
-M	To set nomodifiable on all files opened using command line arguments.

5 Marks and Registers.

Marks are very powerful feature of vim which can be used to jump to specific location from anywhere. Registers are like named clipboards where we can copy data and paste anywhere we want. In all options below one can use any small alphabet or digit in place of '@' in command. Use of '@' is not allowed, it is just placeholder.

- We can see list of marks in current file using ':marks'.
- We can set a local mark in file using ':m@'.
- To jump to any mark we can use '`@'.
- If we use captital letters instead of small letter to mark, the marks are global and can be used to jump across files.
- To see where set of marks are set use ':marks set'.

- We can use '"@y' to copy selected text into register named @.
- We can use '"@p' to paste text from register named @ at current location.
- We can use '"@d' to delete selected text into register named @.
- We can also use combination like '"@3y' to yank three lines in register named @.

6 Help

Help system in vim works as if you are editing a normal file. Some important points related to help are:

- To just start help, use '`:help`'.
- You can use all commands that one uses while editing file while browsing help as well.
- Help has hyperlinks like '`|:help|`'. To jump to hyperlink or tag in vim terminology use '`Ctrl+]`'.
- To go back to last place from where we jumped we can use '`Ctrl+t`'. It is like popping tag from tag stack.
- To get help on particular subject, use '`:help <subject>`'.
- To get help on particular command or shortcut, like x, use '`:help x`'. This gives help on normal mode commands or shortcuts by default.
- To get complete index of what is available, use '`:help index`'.
- To get other mode help on command, we can prefix the command with mode prefix. Some of the prefixes are:

Type	**Prefix**	**Examples**
Normal mode commands	(nothing)	`:help x`
Control mode commands	`CTRL-`	`:help CTRL-A`
Insert mode commands	`i_`	`:help i_<Esc>`
Visual mode commands	`v_`	`:help v_u`
ex-mode commands	`:`	`:help :quit`
command line editing	`c_`	`:help c_<Del>`
Vim command arguments	`-`	`:help -t`
Options	`'` (both sides)	`:help 'shiftwidth'`

- Special keys are enclosed in brackets. For example, to find help on up arrow key one can use '`:help <Up>`'.