Quick Vim Reference

Kurt Schmidt

April 2022

Abstract

Quick reference for Vim editor

Contents

Introduction 3

1 Modes 3

2 Administrative Commands (Normal Mode) 3
  2.1 Redrawing the Window ......................................... 3

3 Motion 4
  3.1 Searching ......................................................... 4

4 Insert Mode 5
  4.1 Inserting Literal Character i_ctrl-v .................................... 5
  4.2 Unicode Characters .............................................. 5

5 Edit Operations 5
  5.1 Undo / Redo .......................................................... 5
  5.2 Deleting Text ...................................................... 6
  5.3 Pasting Text (Put) .................................................. 6
  5.4 Copying Text (Yank) ................................................. 7
  5.5 Changing Text ........................................................ 7
  5.6 Registers ............................................................ 7

6 Formatting 7
  6.1 Indent/Exdent ..................................................... 7
  6.2 Joining 2 lines ................................................... 7
  6.3 Re-formatting ..................................................... 8

7 Vim Text Objects 8
  7.1 Source Code Objects .............................................. 8

8 Visual Mode 9

9 Splitting the Window 9
  9.1 Managing Windows .................................................. 9
  9.2 Moving Between Windows .......................................... 9
  9.3 Command History Window ......................................... 10
10 Buffers
  10.1 Buffers v tabs ................................................................. 10
  10.2 Using Multiple Buffers ....................................................... 10
  10.3 Working With Buffers ......................................................... 10

11 Marks .............................................................. 10

12 Folds .......................................................... 10

13 Miscellaneous ......................................................... 11
  13.1 Repeat ................................................................. 11
Introduction

This document is a quick reference, for things you've learned, and a quick survey of some features you might wish to learn.

You must go through the Vim tutorial

```
$ vimtutor
```
and keep practising. Get comfortable, and add new skills you find useful.

1 Modes

Vim typically starts in Normal Mode. You can navigate around your document, save it, go into insert mode, etc. Typically the ESC key will bring you back to Normal Mode.

Some of the other modes:

- **Insert Mode** Where you do all your typing

  Covered more in Insert Mode Section

- **Visual Mode** Use motions to visually select text

  Covered more in Visual Mode Section

- **Replace Mode** Really just an overstrike mode, as best I can tell.

  Use ESC to get back to Normal Mode, same as Insert

- **Command Mode** For things that start with a :, /, or ?. Shows up at the bottom of the window.

  See the Command History Window

2 Administrative Commands (Normal Mode)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:e fn</td>
<td>Edit fn in a new buffer</td>
</tr>
<tr>
<td>:q</td>
<td>Close current window (if no changes)</td>
</tr>
<tr>
<td>:q!</td>
<td>Close current window, dammit!</td>
</tr>
<tr>
<td>:w &lt;fn&gt;</td>
<td>Write buffer to fn, if provided; otherwise, use current buffer name</td>
</tr>
<tr>
<td>:w! &lt;fn&gt;</td>
<td>Write buffer, possibly overwriting target</td>
</tr>
<tr>
<td>:wq</td>
<td>Write and quit</td>
</tr>
<tr>
<td>ZZ</td>
<td>Write and quit</td>
</tr>
<tr>
<td>x</td>
<td>Write (only if dirty) and quit</td>
</tr>
</tbody>
</table>

2.1 Redrawing the Window

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ctrl-L</td>
<td>Redraw (refresh) the screen</td>
</tr>
<tr>
<td>z</td>
<td>Redraw, current line to top of window</td>
</tr>
<tr>
<td>z.</td>
<td>Redraw, current line to center of window</td>
</tr>
<tr>
<td>z-</td>
<td>Redraw, current line to bottom of window</td>
</tr>
</tbody>
</table>
3 Motion

You can move the cursor around with the following keystrokes while in *Command Mode*. Most (all?) of the following can be preceded by a count. Default is 1.

**Simple motions**

- `h` Left
- `j` Down
- `k` Up
- `l` Right
- `0` Beginning of line
- `$` End of line
- `H` Top of window
- `M` Middle line of window
- `L` Bottom of window
- `gg` 1st line of file
- `nG` nth (last) line of file

Please don’t use the arrow keys. They’re slower than the 4 basic motions, right on the home row. Get good at the basic motions, then add other motions. Don’t just dead-end at arrow keys. That’s Notepad.

**Motions over objects**

- `w` W Beginning of next (big) word
- `e` E End of next (big) word
- `b` B Back. Beginning of previous (big) word
- `( )` Beginning (end) of this or previous (next) sentence
- `{ }` Beginning (end) of this or previous (next) paragraph

3.1 Searching

- `/re` Forward incremental regex search
- `?re` Backward incremental regex search
- `n N` Go to next (previous) match, same direction
- `#` Match previous string under cursor
- `*` Match next string under cursor
- `f char` Advance cursor to next *char* on current line
- `t char` Advance cursor to 1 column before next *char* on current line
- `''` Return cursor to previous line

Note, `/` and `?` are *Command-line Mode* commands; the cursor will jump to the bottom of the window.
4 Insert Mode

There are various ways to get into Insert Mode from Command Mode:

- Insert at (prior to) cursor: i
- Insert at first (non-white) character in the line: I
- Append at next column after cursor: a
- Append at end of current line: A
- Open new line below cursor: o
- Open new line above cursor: O

Each of these can be preceded by a count. You type some text. When you hit ESC, the edit is repeated.

4.1 Inserting Literal Character i_ctrl-v

Use ctrl-v in Insert Mode (i_ctrl-v), or Command Mode, to quote a literal character, like, e.g., Backspace.

Handy if you want the replacement string in a substitute command to have a Newline in it:

\%s/, /\M/g

You can’t type `M as 2 characters; instead, you hit ctrl-v while typing, then the Enter key.

4.2 Unicode Characters

4.2.1 Enter a Unicode

From Insert Mode, hit ctrl-v + u + code. Use u2713 to insert a checkmark.

4.2.2 Digraphs

Vim maps a great many 2-character (and 3-character) sequences to special characters.

While typing in Insert Mode, use ctrl-k, then the 2-key combo.

To see a list of digraphs:

:help digraph

, then page down, or search.

5 Edit Operations

5.1 Undo / Redo

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>Undo</td>
</tr>
<tr>
<td>ctrl-r</td>
<td>Redo</td>
</tr>
</tbody>
</table>

Unlike in the original Vi, we have an undo (and a redo) stack, so, you can undo all the way back to the loaded buffer. (See variable undolevels.)
5.2 Deleting Text

These commands are available in Normal Mode, and can be preceded by a count:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>Delete character under the cursor</td>
</tr>
<tr>
<td>d motion</td>
<td>Delete text from cursor to point of motion</td>
</tr>
<tr>
<td>dd</td>
<td>Delete current (and following, if count &gt; 1) lines</td>
</tr>
<tr>
<td>D</td>
<td>Delete from cursor to end of line</td>
</tr>
</tbody>
</table>

5.3 Pasting Text (Put)

To paste deleted or yanked text into a document from Normal Mode

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>Paste (before cursor)</td>
</tr>
<tr>
<td>p</td>
<td>Paste (after cursor)</td>
</tr>
</tbody>
</table>

xp is a quick, handy way to transpose two characters

5.3.1 Pasting from Outside Vim

Note about pasting from an external clipboard: Things like autoindent, etc., may cause problems if you paste already formatted text or code into a document. Vim supplies a paste setting, which turns off much of the autoformatting:

```
:set paste
```

Remember to cancel this setting for regular editing when done:

```
:set nopaste
```

With this setting in my .vimrc I just hit a key to toggle on and off:

```
" Use <F2> to toggle between 'paste' and 'nopaste'
set pastetoggle=<F2>
```

5.3.2 Read a File, or Command Output, into Buffer

I often find it easier to read contents from a file <fn> right into the buffer. From Normal Mode:

```
:r <file>
```

, will insert contents starting at line beneath the cursor.

To place output from a command into buffer, e.g.:

```
:r! grep Waldo *.tex
:r! date -R
```

5.3.3 Open Source in New Buffer

Or, open the file in a new buffer, use Vim’s clipboard to copy the text over.
5.4 Copying Text (Yank)

<table>
<thead>
<tr>
<th>motion</th>
<th>Yank (copy) word, sentence, paragraph, etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>cntyy</td>
<td>Yank (copy) cnt line(s)</td>
</tr>
<tr>
<td>Y</td>
<td>Yank (copy) cnt lines</td>
</tr>
<tr>
<td>y$</td>
<td>Yank (copy) to end of line</td>
</tr>
</tbody>
</table>

5.5 Changing Text

- Replace single character under the cursor with another. Leaves you in *Normal Mode*.
- Puts you in *Replace Mode*.
- Used much like d, above, but leaves you in input mode.
- Delete char under cursor, leaves you in *Insert Mode*.
- Changes case of character under cursor. Can be used w/a count.

5.6 Registers

Any of the lower-case letters can be used to store deleted or yanked (copied) text, when preceded by a ".". Simply precede any delete, yank, or put operation with a register specifier:

- "f2yy # copy current line and line below into register f
- # move elsewhere
- "fp # paste the 2 lines after the cursor
- "fP # paste the 2 lines before the cursor

You can use *Visual Mode* when yanking or deleting text.

The numbered registers contain a history of your yanks. Vim maintains other registers, too.

`:reg` Will show you all the registers and their contents.

Vim uses the numbered registers, 0-9, and a few others, to automatically store yanked/deleted text and other things.

- "%p # paste filename of current buffer into the buffer

6 Formatting

6.1 Indent/Exdent

- `>>` indent line
- `<<` exdent line

- As with many commands, can be preceded by a count
- Or, use *Visual Mode*, then a single < or > to move whole section
- Or, < followed by a *motion* or a text object

6.2 Joining 2 lines

Use J to join the current line with the line below.
6.3 Re-formatting

You can ask Vim to reformat, e.g., a paragraph, or a function

- Use *Visual Mode*, then `gp`
- Or, `gp<motion>`
  - `gp}` – reformat to end of paragraph
- Inner-objects
  - `gpi` – reformats entire paragraph cursor is on
  - (See Text Objects, below)

7 Vim Text Objects

These *Normal Mode* keystrokes are sorta like motions, but, they don’t generally use the location of the cursor as a mark. E.g., if the cursor is anywhere in a word, we can delete the word, or the sentence, or reformat the paragraph.

Each object can be selected with (a) or without (i) surrounding whitespace (sorta).

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>aw</code></td>
<td>A word</td>
</tr>
<tr>
<td><code>iw</code></td>
<td>An inner word</td>
</tr>
<tr>
<td><code>aW</code></td>
<td>A (inner) big word</td>
</tr>
<tr>
<td><code>as</code></td>
<td>A (inner) sentence</td>
</tr>
<tr>
<td><code>ap</code></td>
<td>A (inner) paragraph</td>
</tr>
</tbody>
</table>

7.1 Source Code Objects

I don’t know that the behavior is dependent on filetype, I’ve not played, but, these objects make more sense in the context of code, some of them language-dependent.

7.1.1 Strings

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a&quot;</code></td>
<td>A (inner) double-quoted string</td>
</tr>
<tr>
<td><code>a'</code></td>
<td>A (inner) single-quoted string</td>
</tr>
<tr>
<td><code>a</code></td>
<td>A (inner) back-quoted string</td>
</tr>
</tbody>
</table>

7.1.2 Other Bracketed Expressions

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>`a)</td>
<td>A (inner) parenthesized block</td>
</tr>
<tr>
<td>`a[</td>
<td>A (inner) bracketed block</td>
</tr>
<tr>
<td>`a{</td>
<td>A (inner) brace block</td>
</tr>
<tr>
<td>`at</td>
<td>A (inner) XML-like markup tag block</td>
</tr>
<tr>
<td><code>ai</code></td>
<td>A (inner) indented block</td>
</tr>
</tbody>
</table>
8 Visual Mode

Visual Mode can be used to select text for many actions, including deleting/yanking, reformatting, substitute (:s) and filter (:!) commands.

\[
\begin{align*}
v & \text{ visual} \\
V & \text{ visual, line mode} \\
\text{ctrl-v} & \text{ visual, block mode}
\end{align*}
\]

ESC will cancel the current Visual Mode, return you to Normal Mode.

9 Splitting the Window

[I don’t quite have the jargon correct, so, don’t get too formal.]

When you call for :help, e.g., the window splits into 2 panes.

Other ways to split the window / open a new one (from Normal Mode):

\[
\begin{align*}
:sp1 & \text{ Splits window horizontally} \\
:sp1 <\text{fn}> & \text{ Opens new window horizontally, opens file} \\
:vspl & \text{ Splits window vertically} \\
:vspl <\text{fn}> & \text{ Opens new window vertically, opens file} \\
\text{ctrl-w n} & \text{ Opens new window horizontally} \\
\text{ctrl-w v} & \text{ Splits window vertically} \\
:term & \text{ Opens new window above, starts a shell instance} \\
q: & \text{ Opens the Command History Window}
\end{align*}
\]

9.1 Managing Windows

Again, many of these can be preceded by a count, which often determines window’s size (height or width)

\[
\begin{align*}
:q[!] & \text{ Quit current window [dammit!]} \\
\text{ctrl-w c} & \text{ Close current window} \\
\text{ctrl-w o} & \text{ Make current window only (close others)} \\
\text{ctrl-w +/}- & \text{ Increase/Decrease height of window} \\
\text{ctrl-w </>} & \text{ Increase/Decrease width of window} \\
\text{ctrl-w H/L} & \text{ Move window left/right} \\
\text{ctrl-w J/K} & \text{ Move window down/up}
\end{align*}
\]

9.2 Moving Between Windows

\[
\begin{align*}
\text{ctrl-w h} & \text{ Go n windows left} \\
\text{ctrl-w l} & \text{ Go n windows right} \\
\text{ctrl-w j} & \text{ Go n windows down} \\
\text{ctrl-w k} & \text{ Go n windows up}
\end{align*}
\]
9.3 Command History Window

This lets you search back through your command history, perhaps edit, and re-execute commands. q: will open a small (7-line) window, which you can navigate in the normal way.

Hit [ENTER] to execute the command the cursor is on.

:q or ctrl-C [ENTER] to cancel.

10 Buffers

10.1 Buffers v tabs

In short, Vim “tabs” should be called “layouts”. Using buffers is the intention. You can certainly skip this discussion.

10.2 Using Multiple Buffers

While editing a file you can use :e <fn> to open another file into a new buffer.

10.3 Working With Buffers

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:ls</td>
<td>List (and index) buffers</td>
</tr>
<tr>
<td>:b &lt;n&gt;</td>
<td>Switch to buffer -</td>
</tr>
<tr>
<td>:bd</td>
<td>Delete (close) current buffer (if no changes)</td>
</tr>
<tr>
<td>ctrl-6</td>
<td>Switch between 2 most recent buffers</td>
</tr>
</tbody>
</table>

I have this handy binding in my .vimrc, so I hit gb, the buffers are listed, then I just type the new buffer number:

nnoremap gb :ls<CR>:b<Space>

Careful, :q will try to quit all the buffers.

11 Marks

- To set a mark: mc, where c is any lower-case letter
- To go to that line: ’c, where c is the mark

12 Folds

The easiest way to create a fold is to use visual mode. Highlight section, then:

:fo

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>zo</td>
<td>Open fold under cursor</td>
</tr>
<tr>
<td>zc</td>
<td>Close fold under cursor</td>
</tr>
<tr>
<td>za</td>
<td>Alternate (open closed fold, close open fold)</td>
</tr>
</tbody>
</table>
13 Miscellaneous

13.1 Repeat

- Use the . to repeat the last edit. Way handy.