

vim workshop - part I

efficient text editing inside the terminal

Branko (@branquito on Twitter) Dušan (dusandimitric.com)

Why vim?



- Extremely efficient text editing mouse/touchpad
 - Designed to deal with text
 - No need to use the mouse/touchpad
 - Macroing, RegExp, Marks, Registers, Tab Sessions. . .
- Highly composable
- Lightweight (therefore fast)
- Commands are (mostly) easy to remember
- You get to learn something new every day
- GPL-compatible license (free & open-source)
- Installed everywhere by default

Getting your feet wet

```
int main(int argc, char** argv) { printf("arg 1 %d", char[0]);  
printf("Macroing; RegExp; Marks. Registers; Tab Sessions...");  
printf("Highly composable"); printf("Lightweight (therefore fast)");  
printf("Commands are (mostly) easy to remember"); printf("You  
get to learn something new every day"); printf("GPL-compatible  
license (free & open-source)"); printf("Installed everywhere by  
default"); exit 0; }
```

Opening files:

```
$ sadf-jdf vim file
```

i	- Insert text before the cursor
<Esc> <C-[>	- Return to NORMAL mode
:w[rite]	- Write buffer to file
:q[uit]	- Terminate current window
:x ZZ	- Save & Close current buffer
:qa! :quitall!	- Close all buffers and exit without saving
:wqa :xa	- Save & Close all buffers and exit

Horizontal motions (1)

- w - next word
- W - next word (skips non-word characters)
- b - previous word
- B - previous word (skips non-word characters)
- e - move to end of word
- E - move to end of word (skips non-word characters)
- ge - backwards to end of word
- gE - backwards to end of word (skips non-word characters)

Horizontal jumps:

- ^ - move to the first non-blank character of the line
- \$ - move to the end of the line
- O - move to the first character of the line
- g_ - move to the last non-blank character of the line
- [n]| - move to column [n] (not a pipe!)

Horizontal motions (2)

Jumping to certain characters:

f{char} - find next occurrence of {char} to the right
t{char} - move 'till next occurrence of {char} to the right
F{char} - find next occurrence of {char} to the left
T{char} - move 'till next occurrence of {char} to the left
; - next
, - previous

Vertical motions (1)

[count] modifier can prefix most vim commands!

[count]j - move [count] lines down ↓
[count]k - move [count] lines up ↑
:[line] - goto [line]
[line]gg - goto [line] (goes to first line by default)
[line]G - goto [line] (goes to last line by default)
{count}% - goto {count} percentage in the file
} - next paragraph
{ - previous paragraph
% - jump to matching ({[<

Vertical motions (2)

[count]H - home line of window + [count]

M - middle line of window

[count]L - last line of window - [count]

Vertical motions (3)

Scrolling

CTRL-E - N lines down (default: 1)

CTRL-Y - N lines up (default: 1)

CTRL-D - half-page down

CTRL-U - half-page up

CTRL-F - 1 page down

CTRL-B - 1 page up

zt - put current line to the top of the window

zz - put current line to the middle of the window

zb - put current line to the bottom of the window

Modes

vim is a **modal** editor and it has six BASIC modes:

- **Normal (command)** mode
- **Insert** mode
- **Visual** mode
- **Comman-line (cmdline)** mode
- **Select** mode *
- **Ex** mode *

* - not widely used

Input commands

i	- insert before cursor
I, ^i	- insert before first non-blank in the line
gI	- insert at column 1
a	- append after cursor
A, \$a	- append at the end of the line
o	- open line below
O	- open line above
:r file	- insert file after cursor line
:r ![cmd]	- insert output of a command [cmd]

Insert commands put **vim** in **INSERT** mode.

Change commands

<code>cw</code>	- change until the end of word
<code>caw</code>	- change a whole word
<code>ciw</code>	- change a whole word (whithout whitespace)
<code>C, c\$</code>	- change to end of line
<code>rc</code>	- replace character under cursor with 'c'
<code>R</code>	- replace by overwriting
<code>s</code>	- substitute 1 character with string
<code>S</code>	- substitute the whole line with text
<code>.</code>	- repeat last change
<code>u</code>	- undo
<code>CTRL-R</code>	- redo

Change commands put changed text to **unnamed register** `""`.

Delete commands

x - delete character
d[n]w - delete [n] words
db - delete previous word
d) - delete to end of sentence
D - delete to end of line
[n]dd - delete [n] lines
dj, dk - delete current line and the line below / above
:10,20d [register] - delete lines 10-20 and put them
 in [register]
:%d - empty the whole buffer

Delete commands put deleted text to **unnamed register ""**.

Copying and pasting text

y{motion}	- yank {motion} text (yiw, yg_, yj, yk)
Y, yy	- yank the entire line
yip, yap	- yank paragraph
:%y	- yank the entire buffer
p	- put text from unnamed buffer after cursor
P	- put text from unnamed buffer before cursor
gp	- same as p, but leaves cursor after new text
gP	- same as P, but leaves cursor after new text

Copy text from one place to another - get text into a **register** using **yank**, **delete** or **change**, then insert the **register** contents with a **put** command.

Registers

The **unnamed register ""** is being used by default.

Named registers "a to "z or "A to "Z registers are only filled by the user.

`:reg[isters]` - list registers and their content

`"{a-zA-Z0-9+"}` - use register (:h copy-move)

`:h copy-move` - help page for copying into registers

`"_{command}` - black hole register (_dd for example)

`:h registers` - more information on registers

VISUAL mode

`v` - per-character VISUAL mode
`V` - linewise VISUAL mode

Most NORMAL mode operators can be used in VISUAL mode.
Some useful operators:

`:` - start Ex command for highlighted lines
`r` - replace with character
`u`, `U`, `~` - lowercase, uppercase, toggle case
`gv` - previous visual selection

Blockwise VISUAL mode:

`CTRL-V` - start blockwise VISUAL mode
`v_b_I` - insert
`v_b_A` - append

Miscellaneous

CTRL+L - redraw

J - join lines

>> - shift right

<< - shift left

CTRL+A - increment number under cursor or next on the line

CTRL+X - decrement number under cursor or next on the line

:h[elp]

Important takeaways



- use the least number of keystrokes possible
- if you're holding a key, you're probably doing something wrong
- use cheat-sheets when learning, there are many online
- use the built-in help, it's very good
- use vimtutor, vim adventure and other resources to learn the basics in an easy and fun way