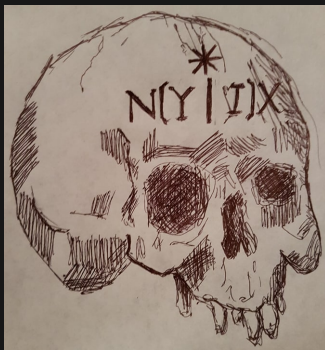


Learn Vim

Glorified Vimtutor

Ricardo Vela

martes 27 de febrero, 2018



- ITC de cuarto semestre
- Pseudo Perl Monger (gotta **grepresent**)
- Comegalletas
- Definitivamente no una autoridad para dar esta plática
- (y hago prácticas en NIC México)



- ¿Qué han escuchado de Vim?
- ¿Lo han usado?

Veremos..

- Instalación
- Cómo entrar
- Cómo salir
- Cómo salir
- Lidar con la frustración de no poder salir
- Cómo salir (reprise)

- ¿Qué distingue a Vim? ¿Por qué Vim?
- Instalación
- Los modos
 - Normal
 - Insert
 - Command/Ex
 - Visual
- Cómo entrar y salir (en serio)
- Normal Mode
 - Movimientos básicos
 - Text Objects y la filosofía de Vim
- Insert Mode
 - Las reglas de Insert
 - CTRL
- Plugins y vimrc
- Más información
- Cheat Sheet

Los propósitos de esta charla son

- Que *intimen* con sus teclados
- Que evalúen las herramientas que utilizan en su día a día.
- Si Vim fuese una espada, que pudieren cortar jamón con ella.

Esto es **independiente** al editor (o cualquier otra herramienta) que decidan utilizar.

¿Qué distingue a Vim? ¿Por qué Vim?

Hay dos cosas que distinguen a Vim entre editores

- Es un editor **modal**
 - Las teclas se comportan de manera diferente según el modo en el que uno esté.
- Sus comandos son **compuestos**
 - Tiene comandos que son **generales** y **contextuales**.
 - Editores como Emacs, Sublime, Atom, etc. tienen comandos que invocan funciones específicas.

www.vim.org

- La versión del repositorio de Debian es la más actual. Recomendando hacer **sudo apt install vim-gtk**.¹
- Si lo prefieren, pueden instalar gVim, la versión gráfica de Vim.

¹después de **sudo apt update**.

Las diapositivas, su código, y algunos ejemplos que trabajaremos están en un repositorio de Github.

```
git clone https://github.com/aricav96/learnvim.git
```

Hay muchos modos en Vim, pero nos interesan los 4 más básicos. Estos son:

Normal

Es el modo default. Aquí es donde ocurre buena parte de la magia (*la edición*). Se accede a él presionando la tecla ESC.

Insert

Es el modo en el que se escribe el texto. Hay muchas maneras de acceder a él: “A/a”, “I/i”, “c{movement}”, “S/s”, “O/o”...

Command/Ex

Es el modo para ejecutar comandos y activar o desactivar variables del editor. Se accede a él tecleando ":" en Normal Mode. En realidad, Command (o Command-line mode) es Ex Mode ejecutado una sola vez. A Ex mode se accede tecleando Q en Normal Mode.

Visual

Es un modo en el que se navega como en Normal Mode, pero hay un indicador de texto elegido. Se accede a él mediante "v", "V" o CTRL-V en Normal Mode.

De estos cuatro, le pondremos particular atención a **Normal** y a **Insert**.

Para entrar..

Basta correr **vim** <filename> o **vim** (sin archivo). En el segundo caso, se tiene que usar **:e** <path> para abrir el archivo deseado.

Para salir..

- En Normal Mode, hay que hacer **:q**
- ... o **:wq**
 - Esto guarda (write) y cierra (quit)
- Sus equivalentes son (también en Normal Mode)
 - **ZQ**
 - **ZZ**
 - **ZZ** no se queja si no se han hecho cambios...

NORMAL MODE

¿Por qué creen que se llama Normal?

- El movimiento en Vim se hace en Normal Mode.
- El movimiento más básico se hace con las teclas **h**, **j**, **k** y **l**
 - h = izquierda
 - j = abajo
 - k = arriba
 - l = derecha
- Los movimientos pueden ser prefijados con un número
 - E.g. 2j mueve el cursor dos líneas hacia abajo.

Text Objects y la filosofía de Vim

Vim reconoce ciertos “objetos”, como palabras, oraciones, párrafos, bloques de código delimitados (por paréntesis, llaves o corchetes) y tags. Estos también tienen sus teclas asignadas en Normal Mode.

- `w/W`, `b/B`, `e/E` para palabras (¿qué constituye una palabra?).
- `()` para oraciones.
- `{}` para párrafos.
- `%` para delimitadores de bloques.

Asimismo, Vim se puede mover a partes del archivo por líneas

- `0/$` para el comienzo/final de la línea.
- `^/_` para el primer caracter de la línea.
- `g_` para el último caracter de la línea.
- `gg/G` para ir a una línea específica (e.g. `2G` te deja en la línea 2).

Algunos movimientos útiles son

- CTRL-B CTRL-U CTRL-Y scroll up
- CTRL-F CTRL-D CTRL-E scroll down
- CTRL-O CTRL-I saltar frenéticamente por el jumplist

Hay otros movimientos que son útiles cuando queremos preservar la posición del cursor

- zz posiciona la línea en el centro de la pantalla
- zt posiciona la línea hasta arriba
- zb posiciona la línea hasta abajo

En casos más heterodoxos, podemos emplear búsquedas con:

- f/F busca un caracter hacia adelante/hacia atrás
- "/" nos permite escribir un regex para buscar

En casos aún más heterodoxos, podemos emplear marcas

- `m{a-zA-Z}` crea una marca en el lugar donde está el cursor
- `'{a-zA-Z}` navega a dicha marca (principio de línea)
- ``{a-zA-Z}` navega a dicha (exactamente a donde está la marca).

Así como podemos prefijar movimientos, podemos prefijar comandos con movimientos.

- cw = change word
- c} = change paragraph
- dw = delete word
- d) = delete sentence

Otros comandos ya incluyen una mezcla de movimiento e inserción

- s = substitute
- r = replace
- a = append
- A = append EOL
- S = substitute line ...

Me tomaré la libertad de poner un **gran recordatorio** aquí para parar y demostrar las demás porque la verdad son bastantes jejejeje.

INSERT MODE

En **Insert Mode** las teclas actúan como siempre. Es el modo donde se puede escribir como lo sabemos (y amamos) hacer.

Esto tiene una implicación importante respecto a teclas modificadoras: Las combinaciones de CTRL también son únicas dependiendo del modo en el que nos encontremos.

- **CTRL-a** inserta texto recientemente añadido. Es el dot command de Insert Mode.
- **CTRL-r + registro** nos permite poner texto en el **registro** referido.
- **CTRL-n y CTRL-p** nos permiten autocompletar según lo que hay en el documento.
- **CTRL-x** nos pone en un modo especial para autocompletar

Vim tiene definido en dónde va a buscar (set complete?). Para indicarle que busque en archivos de un mismo proyecto, es preciso utilizar **ctags**.

- Vanilla(???) Vim es bonito, pero hay algunos plugins que valen la pena.
- Tim Pope es nuestro pastor, nada nos faltará.



Algunos de los plugins más célebres de Tpope son

- vim-surround
- vim-commentary
- vim-sensible

¿Dónde van?

Sin un administrador, hay que descargarlos, ponerlos en `.vim/plugin`. Es importante que tengan permiso de ejecución.

Hay muchas fuentes para consultar sobre Vim:

Documentación

- vimtutor
- :help
- man pages

Libros

- “Practical Vim” de Drew Neil
- “The Vim Tutorial and Reference” Steve Oualline

¿Dudas?

(si no, party tricks maybe?)

version 1.1
April 1st, 06

vi / vim graphical cheat sheet

Esc
normal
mode

~ toggle case	! external filter	@ play macro	# prev ident	\$ eol	% goto match	^ "soft" bol	& repeat is	* next ident	(begin sentence) end sentence	"soft" bol down	+ next line
. goto mark	1	2	3	4	5	6	7	8	9	0 "hard" bol	- prev line	= auto-format
Q ex mode	W next word	E end word	R replace mode	T back 'till	Y yank line	U undo line	I insert at bol	O open above	P paste before	{ begin parag.	}	end parag.
q record macro	w next word	e end word	r replace char	t 'till	y yank	u undo	i insert mode	o open below	p paste after	. misc	.	misc
A append at eol	S subst line	D delete to eol	F "back" find ch	G eof/ goto ln	H screen top	J join lines	K help	L screen bottom	.	ex cmd line	" reg. spec	/ bol/ goto col
a append	s subst char	d delete	f find char	g extra cmds	h	j	k	l	.	repeat t/T/f/F	' goto mk. bol	\ not used!
Z quit	X back-space	C change to eol	V visual lines	B prev WORD	N prev (find)	M screen mid	< un-indent	> indent	?	find (rev.)		
Z extra cmds	X delete char	C change	v visual mode	b prev word	n next (find)	m set mark	.	reverse t/T/f/F	.	repeat cmd	/	find

motion	moves the cursor, or defines the range for an operator
command	direct action command, if red , it enters insert mode
operator	requires a motion afterwards, operates between cursor & destination
extra	special functions, requires extra input
q	commands with a dot need a char argument afterwards

bol = beginning of line, eol = end of line, mk = mark, yank = copy

words: quux(foo, bar, baz)
WORDS: Quux(Foo, Bar, Baz)Main command line commands ('ex'):
:w (save), :q (quit), :q! (quit w/o saving)
:e f (open file f),
:%s/x/y/g (replace 'x' by 'y' filewide),
:h (help in vim), :mew (new file in vim),

Other important commands:

CTRL-R: redo (vim),
CTRL-F/-B: page up/down,
CTRL-E/-Y: scroll line up/down,
CTRL-V: block-visual mode (vim only)

Visual mode:

Move around and type operator to act on selected region (vim only)

Notes:

- (1) use "x before a yank/paste/del command to use that register ('clipboard') (x=a..z,*) (e.g.: "ay\$ to copy rest of line to reg 'a')
- (2) type in a number before any action to repeat it that number of times (e.g.: 2p, dzw, 5l, d4j)
- (3) duplicate operator to act on current line (dd = delete line, >> = indent line)
- (4) ZZ to save & quit, ZQ to quit w/o saving
- (5) zt: scroll cursor to top, zb: bottom, zz: center
- (6) gg: top of file (vim only), gf: open file under cursor (vim only)

For a graphical vi/vim tutorial & more tips, go to www.viemu.com - home of ViEmu, vi/vim emulation for Microsoft Visual Studio