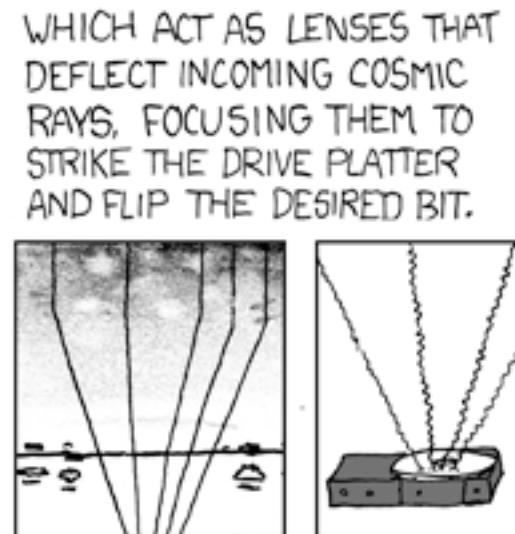
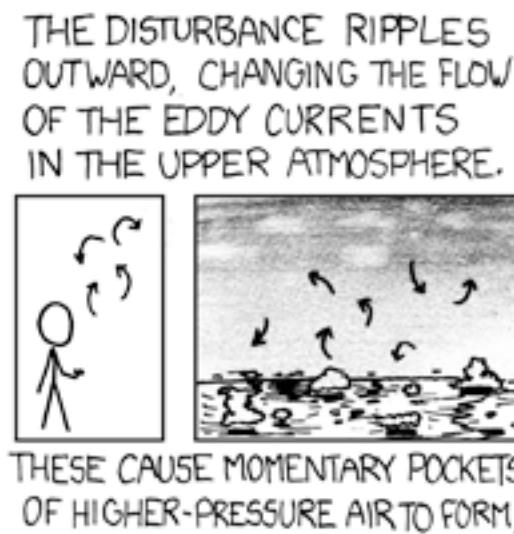
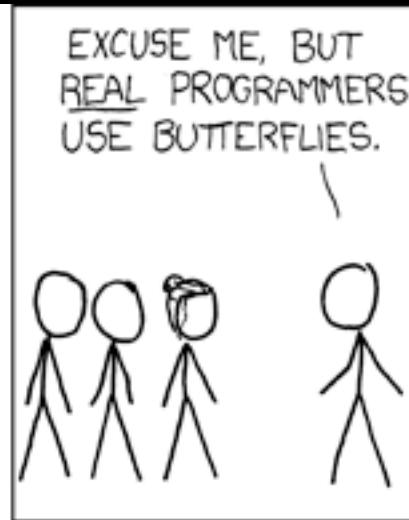
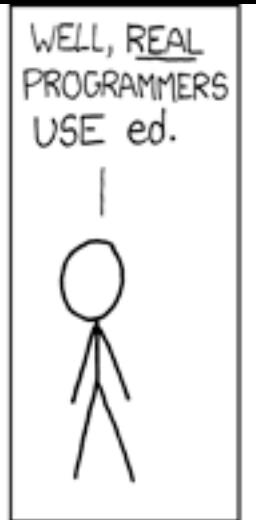
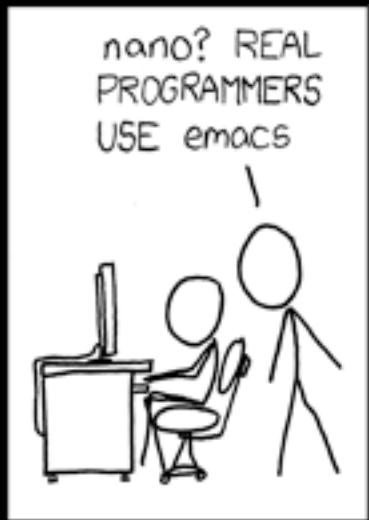
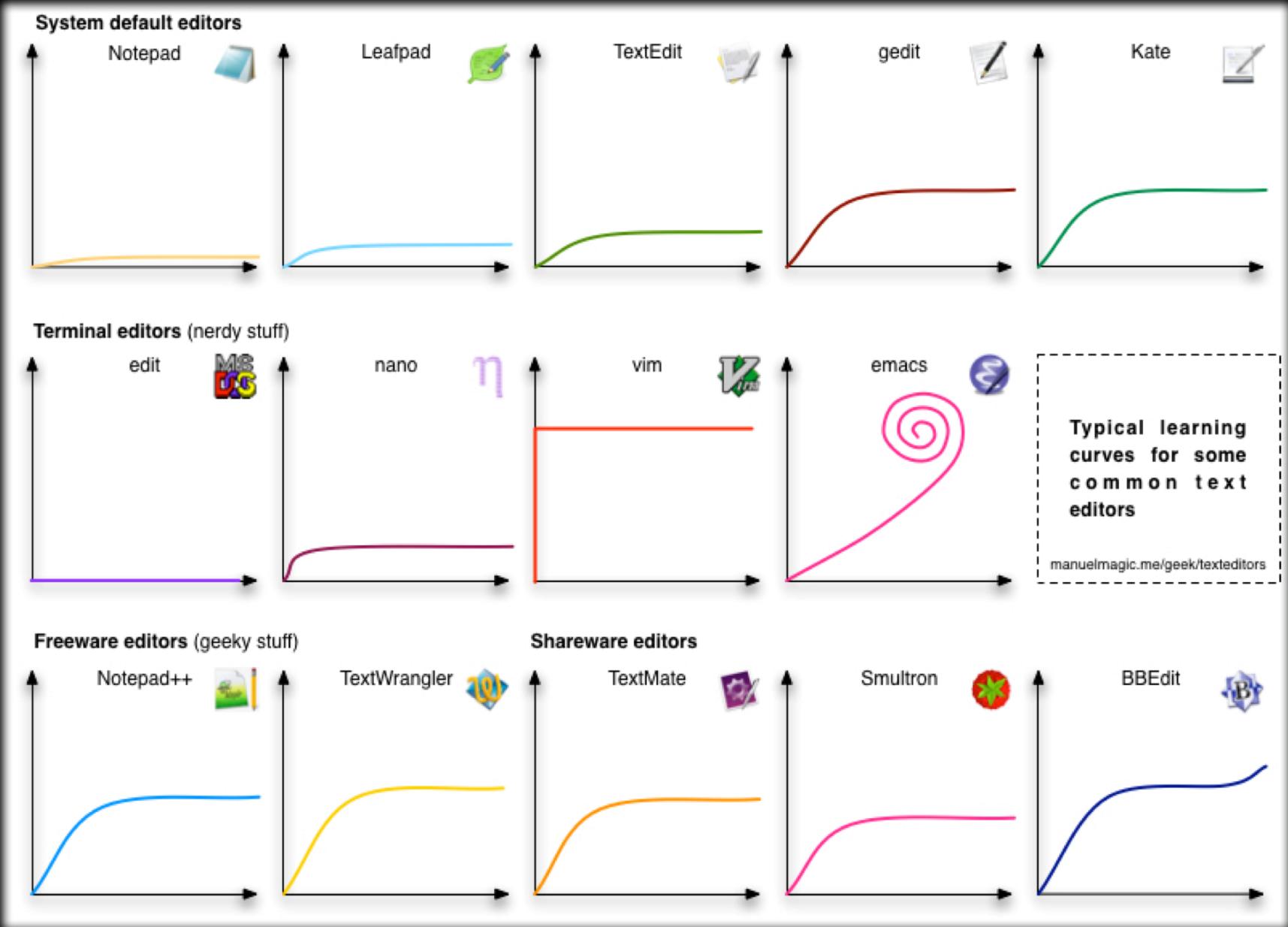


# Text Editors



# Learning



Experience

vi

# Emacs



Do you need to bring lots of unnecessary items camping?

Maybe you will like a text editor that comes with a mail client, a web browser, and a clock

vi



Or do you appreciate  
beauty and simplicity?

vi seems  
challenging,  
but it's actually  
more natural  
and simple than  
emacs

# What's the real difference?

**Emacs:** One mode

- Text and commands must be different combinations of keys
  - Press modifier key every time
  - Commands can be complicated
  - Move your hands a lot
  - Easy to learn, hard to mess up, but annoying and inefficient to use

**vi:** Two modes

- Text and commands can use the same alphanumeric keys
  - Press escape or i once
  - Commands are simple and intuitive
  - Rarely move your hands
  - Until you get used to modality, you'll be a little frustrated, but the end result is a better experience

# Vi

My take on [EmacsVsVi](#) (for text editing, ignoring all the additional stuff Emacs can do) has always been that Emacs is easier to *learn*, but Vi is easier to *use*. A modeless tool is easier to get a handle on, because there's never any question as to what state it's in. But it means you need a lot more knobs and buttons to get everything done, making it necessarily bigger and klunkier. A tool with modes is harder to learn, but once you get the hang of it, can be quicker to use if well designed. [=](#)

[Stefan Vorkoetter](#) (who uses Vi after a brief stint with Emacs in the late 80's).

# Insert Mode

Press the i key to start  
INSERTING text

Move around with  
arrow keys

nomascus — ViCommandsSimple.txt (~) - VIM — vim — 79x43

```
1 you should use vi, because it's simpler
2 and lighter
3 and cooler
4
5 MODES
6
7 i           insert mode
8 escape      command mode
9
10 CUTTING AND PASTING
11
12 dd          delete (cut) line
13 cc          cut line
14 yy          yank line (copy)
15 yw          yank word
16 p           paste into next line
17 P           paste at cursor
18 u           undo
19 ctrl r     redo
20
21 MOVING AROUND
22
23 0           beginning of line
24 $           end of line
25 shift arrow key beginning/end of word
26 :number     move cursor to the (number) line
27 arrow key   move up, down, left, right
28 {
29 }
30
31 SAVING AND EXITING
32
33 :w          write file
34 :q          quit vi
35 :q!
36 :wq         quit without saving
37
38 OTHER
39
40 :set nu    add number lines
~
```

# Command Mode

Press the escape key to  
leave insert mode

Cut & paste  
Move  
Search  
Save & exit

Commands are simple  
and intuitive

You don't have to  
constantly press the  
control key

**Esc**

normal mode

~ toggle case	! external filter	@• play macro	# prev ident	\$ eol	% goto match	^ "soft" bol	& repeat :s	* 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 <sup>3</sup> 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 <sup>1,3</sup>	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 <sup>1,3</sup>	f find char	g extra <sup>6</sup> cmd	h ←	j ↓	k ↑	l →	; repeat t/T/f/F	'. goto mk. bol	\ not used!	
Z quit <sup>4</sup>	X back-space	C change to eol	V visual lines	B prev WORD	N prev (find)	M screen mid'l	< un- <sup>3</sup> indent	> indent <sup>3</sup>	? find (rev.)	/ find		
Z extra <sup>5</sup> cmds	X delete char	C change <sup>1,3</sup>	V visual mode	b prev word	n next (find)	m set mark	,	,				

**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), :new (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, d2w, 5i, 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)

There's also a video game to help you learn vi  
(you might be able to run this in emacs if you mash enough keys together...who knows...)

