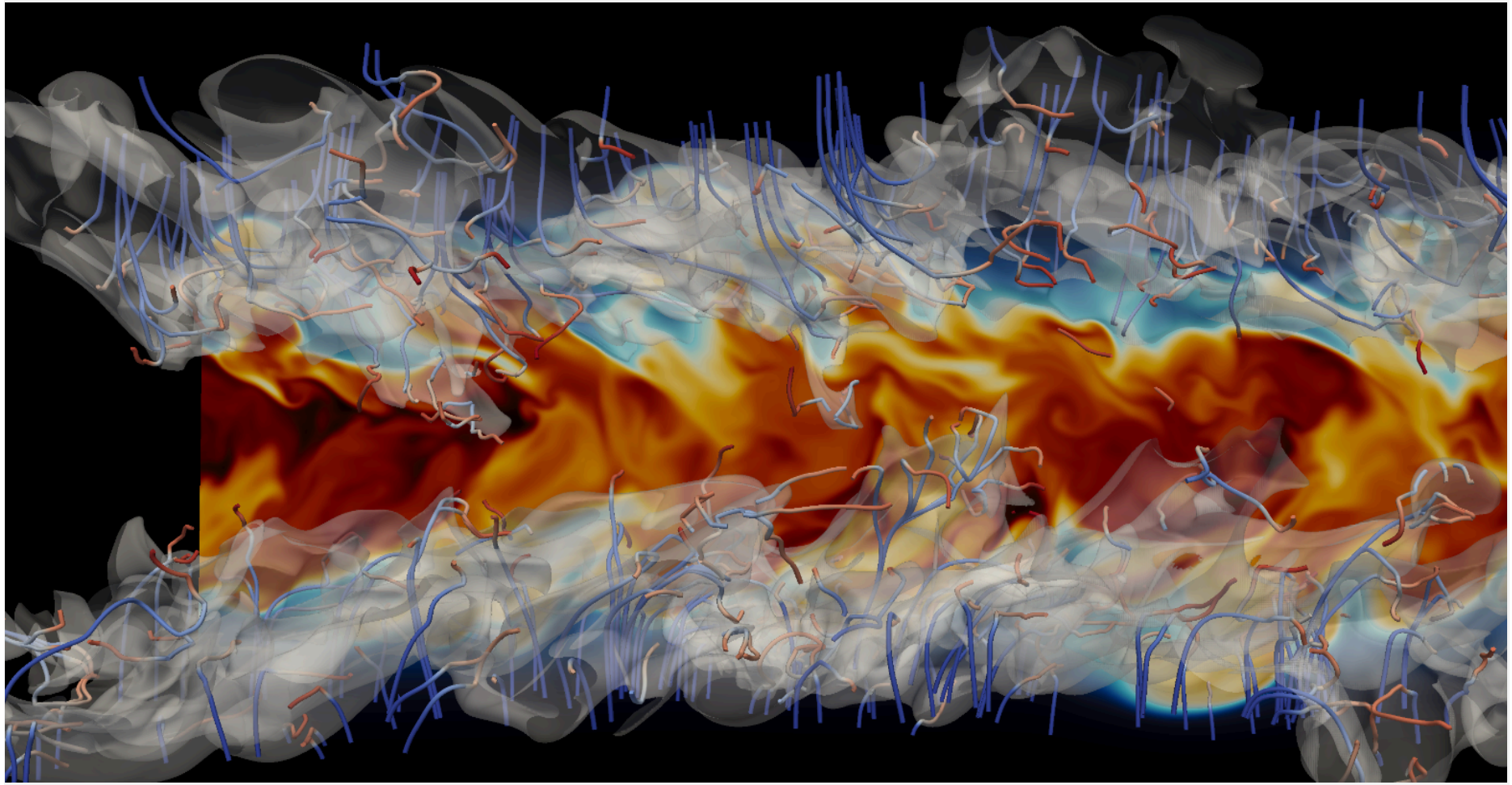
Software Tools for UNIX/Linux Systems

Part 4: vim

C. Hasse

TECHNISCHE UNIVERSITÄT DARMSTADT

STFS



Agenda

1 History
2 Motivation
3 Fundamentals
4 Conditionals
5 Loops
6 Command-Line Options
7 Functions
8 Command Subtitution
9 Real-World Examples

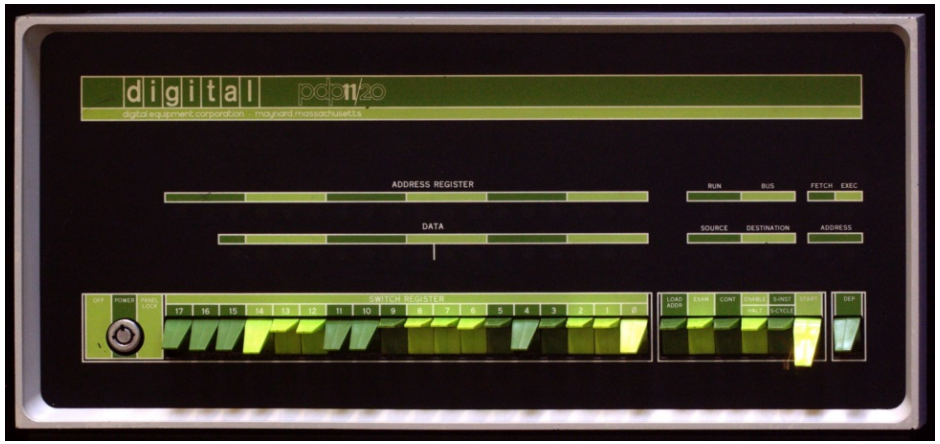
History – PDP 11

so you got your new computer in the early 70s ... lets start programming

← is that a keyboard?

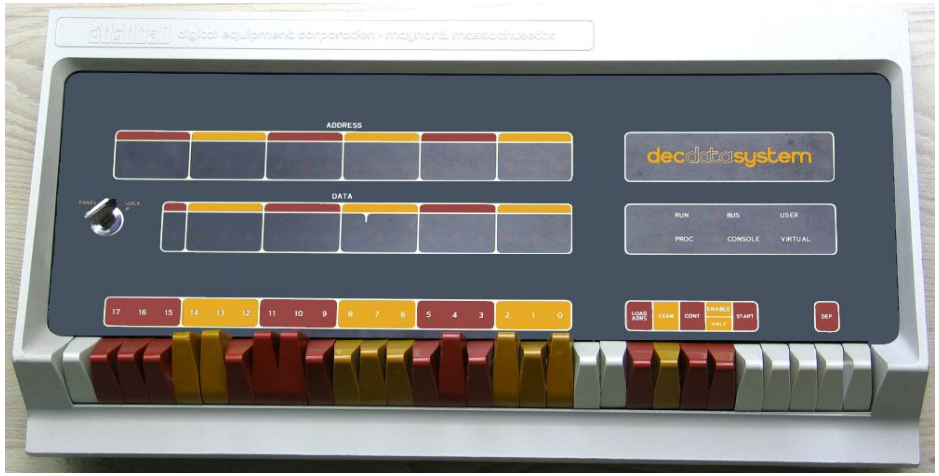
well not like you know them

History – PDP 11 front panel



- PDP 11/20 front panel
- used to manually switch bits in registers
- can we at least "see" some results?

History – PDP 11 front panel

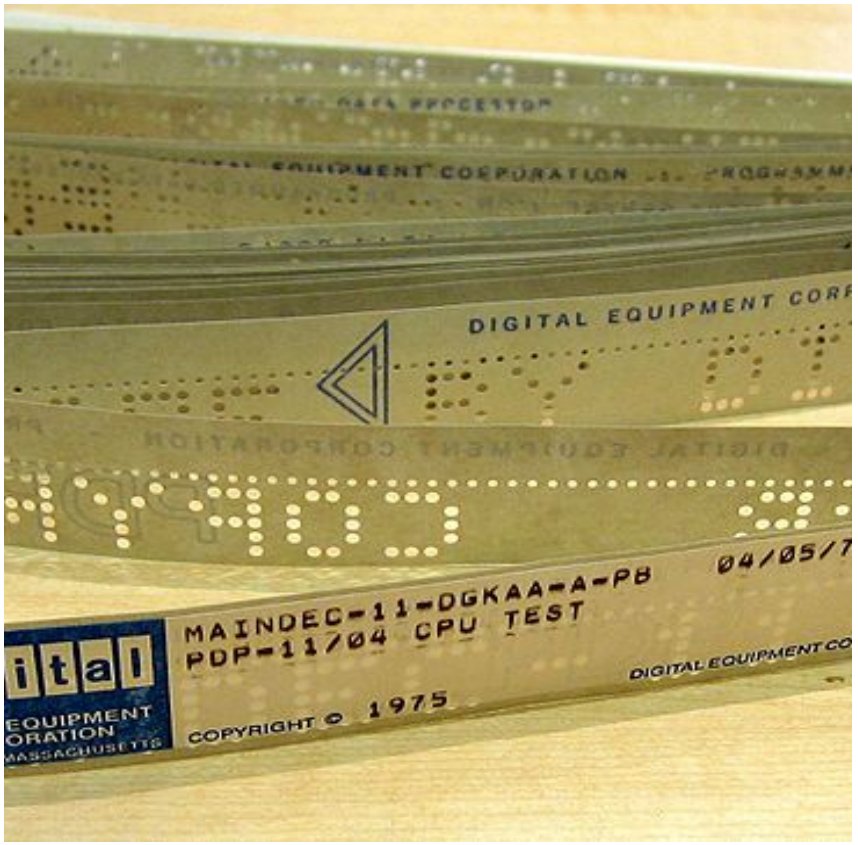


- sure this one has LEDs
- but if you "load" a driver you could use a DecWriter

https://commons.wikimedia.org/wiki/File:Decwriter.jpg

History – PDP 11 tape reader

paper tape reader



paper tape with driver in machine code

History – PDP 11 DecWriter

History – PDP 11 DecWriter in action

History – summary editors

- 1971 ed: line editor Ken Thompson (g/re/p)
- 1976 em: ed for mortals
- 1977 ex: Bill Joy extended version of
  - em -> en -> ex
- ex had a visual mode for new CRT terminals with shorthand "vi" in console
- copied several features from Bravo editor
- 1978 include in BSD

- licensed under BSD
- 1991 "Vi IMproved": vim Bram Boolenaar

History – using ed writing


```
$> ed
a
this is my first line
wow that is fun
.
w test.txt
38
q
$> more test.txt
this is my first line
wow that is fun
```

notice the sketchy definition of "fun"

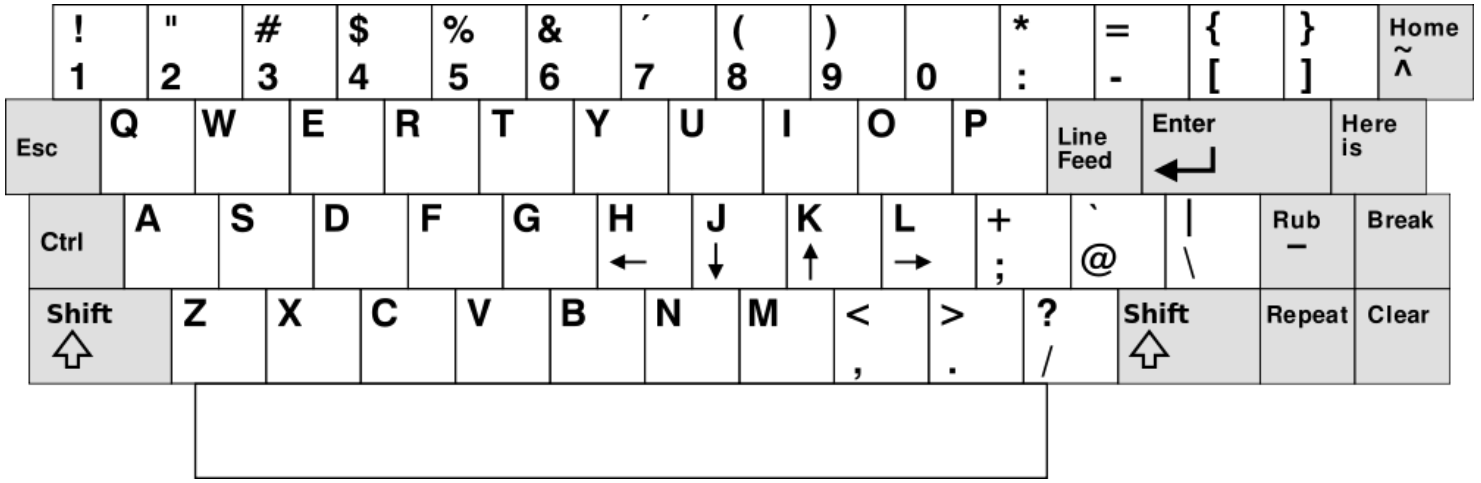
History – using ed editing


```
$> ed test.txt
38
a
wow yet another line.
.
w
60
q
$> more test.txt
this is my first line
wow that is fun
wow yet another line.
```

please stop the torture

History – CRT Terminal

- ADM-3A 1975
  - 12 inch
  - 12 lines 80 char wide
  - no arrows
  - no caps-lock
  - convenient esc



Motivation for vim

- almost every unix system has it
- very fast, especially for large files
- doesn't require a real GUI (X-Forwarding or VNC)
- distraction free editing (no mouse)
- flexible, expandable using scripting features

- however: one of the steepest learning curves you may encounter

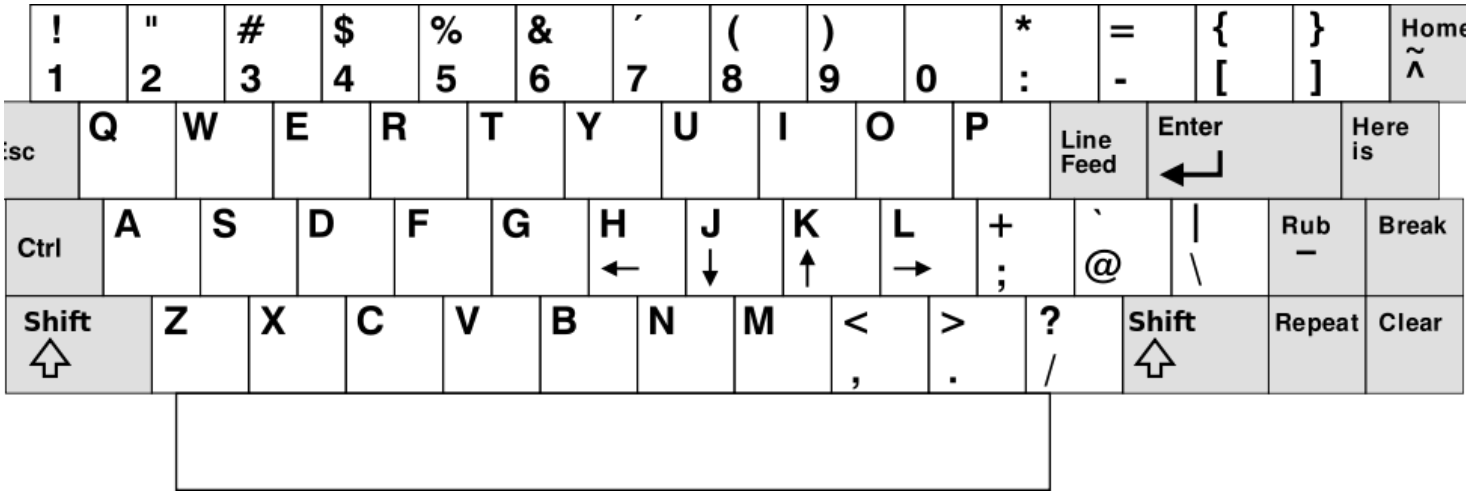
Fundamentals – two mode editor

- vim is a two mode editor
  - insert mode: insert characters as usual under cursor position
  - normal mode - everything else like: deleting, replacing, searching, file operations, switching to insert mode, ...
- ESC always returns you to normal mode

Fundamentals – safely escape from vim

- enter :q
  - if you forget the ":" you are in macro record mode
  - vim wants to know a shorthand for this macro, you may enter any character e.g. "s"
  - finish macro recording pressing "q“
- enter :q! if you made any changes (quit but don't write)
- change $EDITOR if a programm/script force you into vim ("nano" is a good beginners choice)

Fundamentals – moving



movement with arrow keys was not standardized

- H J K L will always work (as displayes above)
- touch typing "homerow" allows for minimal hand movement

Fundamentals – programmer virtues

According to Larry Wall(1), the original author of the Perl programming language, there are three great virtues of a programmer; Laziness, Impatience and Hubris

- Laziness: The quality that makes you go to great effort to reduce overall energy expenditure. It makes you write labor-saving programs that other people will find useful and document what you wrote so you don't have to answer so many questions about it.
- Impatience: The anger you feel when the computer is being lazy. This makes you write programs that don't just react to your needs, but actually anticipate them. Or at least pretend to.
- Hubris: The quality that makes you write (and maintain) programs that other people won't want to say bad things about.

(1) Quoted from "Programming Perl", 2nd Edition, O'Reilly & Associates, 1996

Fundamentals – vim is a developer tool

TECHNISCHE UNIVERSITÄT DARMSTADT

- editors at the time were made for programmers only
- an editor has to tend to your needs
- many times you have to repeat yourself

- vim allows you to repeat the last action (minimacro) using the dot command (undoing it is "u")

vim ex01_code.txt
a = 2.0
b = 14
c = 'this'
~ <- this shows you that there is nothing in file left to display

DEMO ex01_code.txt : append „;“ to every line using "a;" -> "j$." -> "j$."

Fundamentals – first commands

TECHNISCHE UNIVERSITÄT DARMSTADT

key	command
i	insert before position
a	append after position
ESC	return to normal mode
Shift + i	insert before first position in line
Shift + a	insert after last position in line
x	delete character
d_	delete until movement
ex. d2l	delete next 2 characters
ex. dd	delete entire line

DEMO ex02_delete.txt

Fundamentals – more movements

TECHNISCHE UNIVERSITÄT DARMSTADT

key	command
w	move to beginning of next word
b	move to beginning of last word
e	move to end of next word
ge	move to end of last word
$	end of line
^ 0	start of line (^: first letter) (0: first position)
G	end of file

almost every command can be repeated preceeding a number

key	command
10w	go 10 words forward
12G	goto line 12 of text
d10G	delete the next 10 lines

DEMO:
ex03_movement.txt

Fundamentals – managing files

TECHNISCHE UNIVERSITÄT DARMSTADT

	command
:w	write to disk
:w file.txt	write to disk using filename file.txt
:e file.txt	open file.txt in current window
:r file.txt	read something from file.txt and put it into the current file
:!ls	key(e.g. here ls, general form: „:!<command>“
:r!ls	read output from command in shell into file (example ls) „:r!<command>“

DEMO: ex04_documentation.txt: save it as ex05_manual.txt

Fundamentals – search and replace

TECHNISCHE UNIVERSITÄT DARMSTADT

key	command
/	search for string forwards (? search backwards) (/string)
n	goto next result in search (N goto next result backward direction)
*	search word under cursor (# search word under cursor backwards)
g*	same as before but with partial matches
r	replace one character by the one following (R till EOL)
c	change (e.g. cw word as movement) (C change till EOL)
s	substitute char and go into insert mode (S substitute entire LINE)

DEMO: ex05_manual.txt

Fundamentals – copy & paste

TECHNISCHE UNIVERSITÄT DARMSTADT

Key	command
d	delete text, also stores it in clipboard
p	paste text in clipboard after cursor (P before cursor)
y	yank movement indicator sets range (yy whole line)
+	goto next line, - last line (y2+ copy next 2 lines)
:reg	list of registers
"ay	yank text into register a
"ap	paste text from register a after cursor
:wv, :rv	write / read viminfo file to sync registers

registers 0-9 are automatically filed, a-z are named and stored in .viminfo

DEMO ex06_copy.txt

Fundamentals – visual mode

key	command
v	start visual selection; close using operation or ESC (v mov y)
V	line wise visual mode
<Ctrl>-v	blockwise visual mode
o	switch end of selection
<Ctrl>-f	scroll page forward (<CTRL>-b scroll backward)

- visual mode using the mouse can be enabled on terminals that support it
- avoid visual mode as much as possible or you'll be spending a lot of time using cursor keys

DEMO ex07_visual.txt

Fundamentals – buffer

TECHNISCHE UNIVERSITÄT DARMSTADT

key	command
:ls!	list available buffers
:b	switch to buffer either by name or number or regexp
:badd	open new file and add to buffers
:bdel	destroy buffer

easily switch between file, no need for an application with several tabs or several open windows

DEMO: simply use two files

Fundamentals – windows

TECHNISCHE UNIVERSITÄT DARMSTADT

key	command
:new	new window from file (or empty) split horizontally (also <CTRL-w> s)
:vnew	same as above split vertically (also <CTRL-w> v)
<Ctrl-w> mov	move to other window in direction
<Ctrl-w> +	resize window vertically e.g. "10 <Ctrl-w> -"
<Ctrl-w> <	resize window horizontally e.g. "10 <Ctrl-w> <"
<Ctrl-w> =	resize all to equal size
<Ctrl-w> _	maximize current window height (width |)
<Ctrl-w> o	make current window the only one
<Ctrl-w> c	close current window

DEMO: vim –o ex08_header.txt ex08_source.c

Fundamentals – tabs

TECHNISCHE UNIVERSITÄT DARMSTADT

key	command
:tabedit	open file in new tab
:tabclose	close current tab
:tabonly	close all other tabs
gt	next tab
gT	previous tab
:tabmove	move tab to front (0), numbered pos (2) or end ()

DEMO: vim –p ex08_header.txt ex08_source.c

Fundamentals – macros

TECHNISCHE UNIVERSITÄT DARMSTADT

key	command
q	record named macro (and quit hitting q another time)
@	playback macro
<Ctrl-A>	increment current number (<Ctrl-X> decrement)
:reg	see macro in named register

ex-command	command
:%s/a/b/gc	search and replace a by b with confimation globally

DEMO: ex09_macro.txt

Plugins - installation

- vimball : vim myplugin.vba; :source %
- download zip/tar/... and extract folder contents of
  - doc plugin .. to ~/.vim folder
- install https://github.com/tpope/vim-pathogen.git
  - and extract all plugins to ~/.vim/bundles/... they will be detected automatically

Plugins

- delimitMate : automaticly add closing brackets
- supertab : autoexpand when pressing tab
- NERD_tree : file browser
- ack : ack search
- Tabular : draw nice tables with ascii art
- voom : two-pane outliner supporting many formats
- taglist : automatically search expressions in ctags file
- vimwiki : markdown style wiki for vim

- ...

Further resources

http://vim-adventures.com

http://www.vim.org/scripts/

:help help

:vimtutor

https://github.com/jmoon018/PacVim

What should you have learned?

TECHNISCHE UNIVERSITÄT DARMSTADT

List of things you should know:

- How to use vim effectively
- Know standard commands (opening, closing, moving, yanking….)
- What features does vim provide?
- Where can it be really usefull?
- Why is vim superior to emacs?
- Hint: If you want to impress the professor learn also about emacs