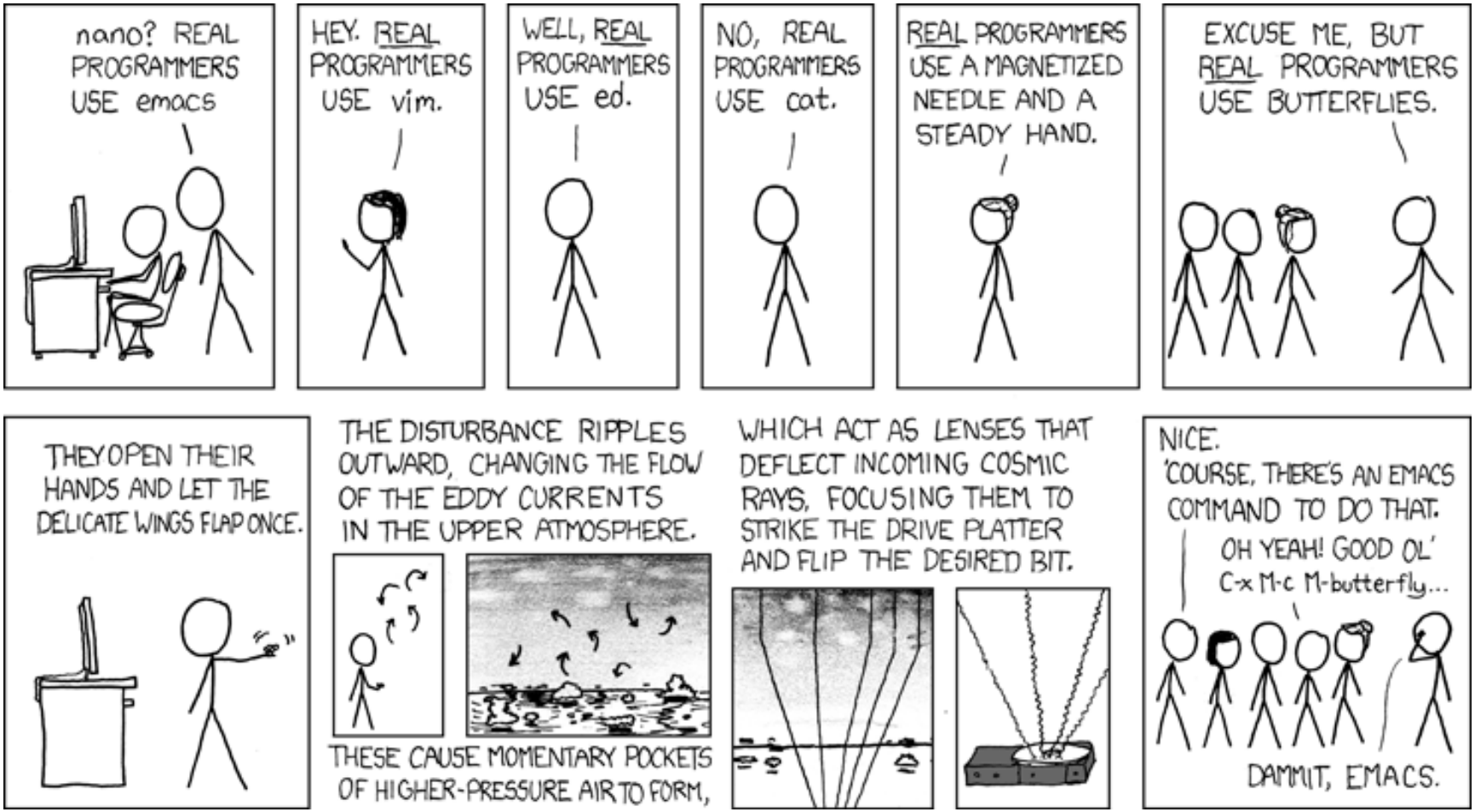
Text Editors for Programmers

Dr. Prasad Kulkarni
Michael Jantz
Jamie Robinson

Real Programmers



vim

- Based on vi
  - vi written in 1976 and has become standard on Unix machines
- Basic design principles:
  - Retains each permutation of typed keys to resolve commands
  - Smaller and faster editor – but with less capacity for customization
  - Uses distinct editing “modes”

Using Vim on a Simple Example

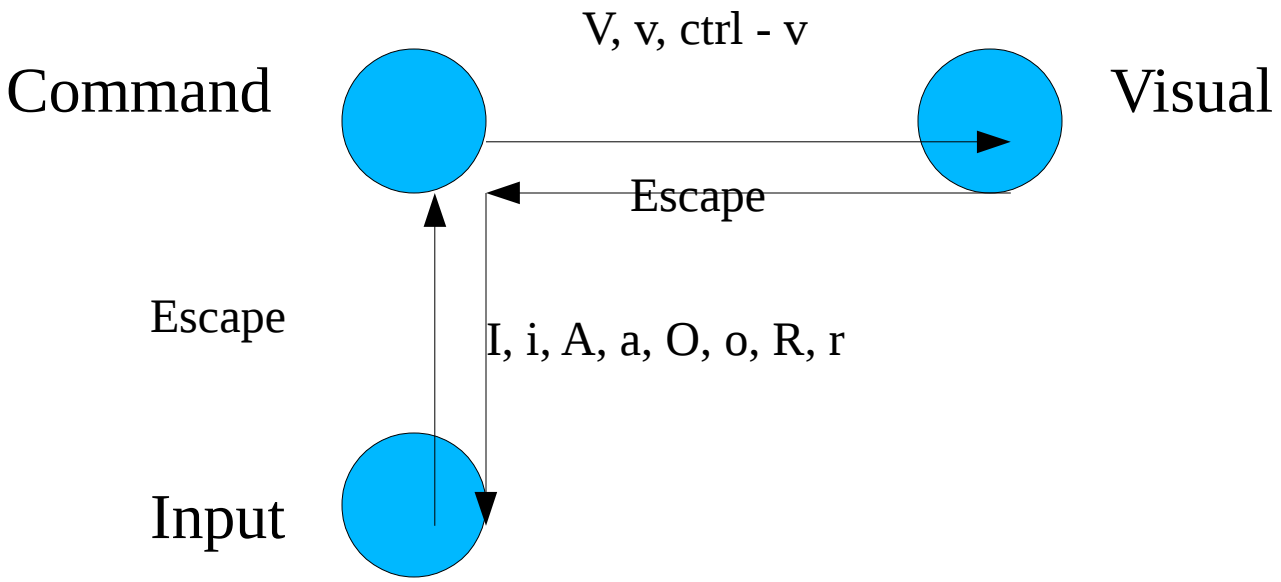
- You should have received two source files (simple.c and simple.h), a Makefile, and a dot_vimrc file from the lab website.
  - Save *dot_vimrc* as *.vimrc* in your home directory
  - Use mv to rename the file
    - mv dot_vimrc ~/.vimrc
- “dot_vimrc”
  - A collection of vim commands run each time you start vim
  - Used to set mappings / options that are not otherwise set by default.

Using Vim to Create & Edit a File

- Start a session
  - vim simple.c
- Press 'i' to enter insert mode
  - Now type any text you want
- 'Esc' to enter command mode
  - ':wq' to write changes and exit the session

Vim – Modes of Operation

- Command Mode
- Input Mode
- Visual Mode

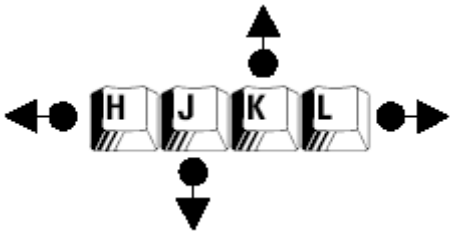


Essential Commands

- :e *file*
  - Edit *file* in a new buffer
- :w
  - Save any modifications to the current buffer.
- :q
  - Quit Vim. If you have modifications you do not want to save, use :q!
- u, <c-r>
  - Undo, redo

Command Mode: Navigation

- Reopen simple.c
  - Use j, k, l, and h to navigate around the file as shown. This may take awhile get used to, but is very nice once you have it down.
  - For faster page scrolling, use <c-b> and <c-f> for page up and page down.
    - I've mapped these commands to spacebar and backspace in my .vimrc



Input Mode

- The following commands switch to input mode:
  - i – characters inserted just before the cursor position
  - I – characters inserted at the beginning of the line
  - a – characters inserted just after the cursor position
  - A – characters appended to the end of the line
  - o – characters inserted in a new line below the cursor
  - O – characters inserted in a new line above the cursor
  - C – Often overlooked, deletes the line after the cursor position and start inserting characters at this position
- After you're done editing, press Escape to go back to command mode, and :w to write the changes

Common Editor Commands

- Cut/copy/paste in command mode:
  - dd – cut a line of text
  - yy – copy (“yank”) a line of text
  - P/p – paste a line of text above / below the cursor position
- Commands in Vim can be applied to multiple lines by typing the number of lines you want before the command:
  - “12dd” cuts 12 lines of text
  - “4j” moves the cursor down 4 lines

Common Editor Commands (cont).

- gq<motion command> - Format a block of code to comply with textwidth setting
  - <motion command> is any of the commands to move the cursor (i.e. j, k, h, and l)
  - See example in simple.c
- == - Format a block of code to correspond to tabbing conventions
  - See example in simple.c

Searching

- */word* – Search for occurrences of *word*
  - Cursor jumps to the next occurrence of *word*
  - n/N – jump to the next / previous occurrence of word
  - ?*word* – search initially jumps to previous occurrence of *word.*
- :set ic – ignore case
- th – toggle search highlighting

Find/Replace

- :s /*search_for*/*replace_with*/
- Variations
  - :s /*s*/*r*/g – Replace every occurrence on the line (not just the first)
  - :%s /*s*/*r*/g – Replace every occurrence in the current buffer
  - :s /*s*/*r*/g 12 – Replace for the next 12 lines
  - :s /*s*/*r*/gc – Replace, but get confirmation before doing so
  - :s /*s*/*r*/gi – Ignore case when searching for *s*.

Setting the Mark

- ma – Sets the mark a to the current cursor position
  - a is not unique, any alphanumeric character can be used.
- Now, pressing **`a** in command mode returns you to the position marked by a.
  - Helpful for getting back to hard to find sections of code quickly
  - See the example in simple.c that shows how it can be used with the find/replace command to comment out large sections of code.
- da – Deletes the mark a.

Visual Mode

- V/v enter into visual mode
- Allows user to visually select text for commands.
- Navigate in visual mode as in command mode (g,j,h,k)
- Issue commands with selected text ('y' to yank, 'd' to cut, etc.)
- `esc` exits visual mode

Buffers

- Vim allows you to edit multiple files in one session using buffers
  - <c-w> v to split the screen vertically
  - <c-w> s to split the screen horizontally
  - <c-w> w to switch to the other screen
  - :hide close current window
  - :only keep only this window open
  - :b <buffer-number> open buffer-number
  - :V/Sex – splits the screen vertically or horizontally and opens a file explorer in the new screen.
  - Select simple.h to open it in the new screen.

Installing Buffer Explorer

- Vim has built-in commands to work with its open buffers, but there is a plugin that allows you to visualize and navigate the open buffers.
  - Goto http://vim.sourceforge.net/scripts/script.php?script_id=42
  - Download the latest version of bufexplorer.zip and extract it
  - In your home directory, if a .vim/ directory does not exist, create it:
    - ls -a .v*
    - If .vim/ is not present, do mkdir .vim
  - Now, move the contents of the extracted bufexplorer folder into .vim/:
    - mv bufexplorer/doc/ bufexplorer/plugin/ .vim/
  - Inside your vim session do:
    - :helptag ~/.vim/doc/
  - Quit and reopen vim

Buffer Explorer

- \be – Opens the buffer explorer in the current screen. Allows you to navigate (as in command mode) and select a buffer.
- Also can press the number of the buffer to select a buffer.

Tagging the Source

- Big advantage to Vim is its integration with a source code tagging program.
- Inside a terminal, goto the directory of the simple source and type:
  - ctags -R
- Should create a file named tags. Now, reopen simple.c in Vim.

Using Tags with Vim

- <c-]> - With your cursor over a variable, jump to the declaration of that variable.
- <c-t> - Having jumped to a declaration, go back to the spot you jumped from
  - You can use <c-]> multiple times before using <c-t>. The functionality operates like pushing and popping frames on a stack.
- Extremely helpful for browsing and learning large programs.

Colors

- Colorschemes can be downloaded from:
  - http://www.cs.cmu.edu/~maverick/VimColorSchemeTest/
- Current default colorschemes for EECS machines are in:
  - /usr/share/vim/vim72/colors/
- Set a new colorscheme with:
  - :colorscheme name

Vim Resources

- Vim Tips Wiki:
  - http://vim.wikia.com/wiki/Main_Page
- Vim Cookbook
  - http://www.oualline.com/vim-cook.html
- Slashdot comments discussing Vim tips:
  - http://ask.slashdot.org/article.pl?sid=08/11/06/206213
- For everything else, just use Google.

Screen

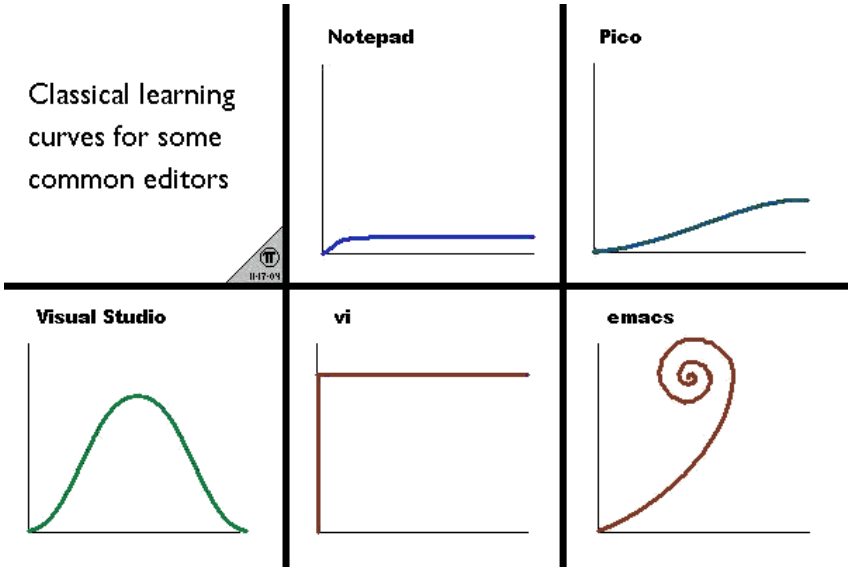
- Screen is a 'terminal multiplexer'
  - Allows users to access multiple separate terminal sessions from within a single terminal window or remote terminal session.

Screen Essential Commands

- Commands outside screen:
  - Create a screen: bash> screen
  - List screens: bash> screen -ls
  - Attach to existing screen: bash> screen -D -R
- Commands within screen:
  - Create a new terminal: <c-a> + c
  - Cycle through terminals: <c-a> + n, <c-a> + p
  - Kill terminal: <c-d>
  - Detach from screen: <c-a> d

Editor Comparisons

- Vim vs. Emacs
  - Vim is primarily an editor
  - Emacs is a Lisp interpreter running an editor
- Wikipedia: Editor War



Emacs Properties

- Single mode editor
  - Each key is a command to add the letter to the buffer
  - Key combos shortcut commonly used commands
    - “C-x C-s” to save a file to disk
    - “C-x C-c” to exit Emacs
  - Access all commands with “M-x <command name>”
- Highly extensible and versatile
  - Plugins for almost any functionality
  - Emacs Lisp (Elisp) programming language
  - More than just a text editor:
    - M-x flyspell-prog-buffer, M-x shell, M-x life, M-x list-packages

.emacs

- Similar to .vimrc
  - An Elisp script run at start up
  - Used for configuring Emacs options and attaching plugins
- Follow the instructions for the dot_vimrc file on the dot_emacs file while changing the file names where necessary

Emacs Navigation and Command Shortcuts

- Command shortcut syntax:
  - C- means hold Ctrl key
  - M- means hold Alt key or press Esc key
- Forward 1 character: C-f
- Backward 1 character: C-b
- Go to previous line: C-p
- Go to next line: C-n
- Forward 1 word: M-f
- Backward 1 word: M-b
- Page up: M-v
- Page down: C-v
- Go to top of buffer: M-<
- Go to bottom of buffer: M->
- Go to line “n”: M-g g “n”
- Repeat following command “n” times: <Esc> “n” command
- Repeat last command:

  C-x z [z z z …]
- All commands can be accessed with M-x “command name”

Basic Emacs Command Shortcuts

- Start Emacs from the command line:
  - emacs
  - emacs “file name”
- Close session: C-x C-c
- Suspend session: C-z
- Open file: C-x C-f
- Save file: C-x C-s
- Undo = Redo: C-_ or C-x u
- Abort command: C-g
- Begin mark region: C-<space>
- Copy region: M-w
- Kill region (Cut): C-w
- Yank (Paste): C-y
- Search buffer contents: C-s
- Select buffer: C-x C-b
- Previous buffer: C-x <left>
- Next buffer: C-x <right>
- Window - split:
  - Vertically: C-x 2
  - Horizontally: C-x 3
- Window - cycle cursor: C-x o
- GNU Reference Card