Editing with vi

Or more fun than you thought you'd have without a mouse

COMP 2400, Fall 2008

Prof. Chris GauthierDickey

UNIVERSITY OF
DENVER

The vi editor

- The standard interactive editor on Unix systems is called 'vi'.
- How do you open a file?
  - vi file1 file2
    - This allows you to edit two files, first file1, then file2

Command Mode

- When you start vi, you begin in command mode
  - All commands are entered from the keyboard--no mouse here!
- vi has two other primary modes, insert mode for free-form text input, and ex-mode for running commands on the text

Movement Commands

- Movement: You can move the cursor around using the arrow keys
  - In the old days, we used h (left), j (up), k (down), and l (right)
- 0 moves to the beginning of a line, $ moves to the end
- <ctrl>-F and <ctrl>-B move forward and backwards a page

More Movement

- w moves you to the start of the next word
- e moves you to the last character of the next word
- b moves you to the previous word
- Note that vi separates all words by punctuation
  - Use the capitalized versions: W, E, B for whole words

And more movement

- Use (and) to jump to prior and next sentences
- Use { and } to jump to prior and next paragraphs

Saving and Quitting

- Typing : followed by a letter executes what is called 'ex' mode.
- :q will quit vi, if you have nothing to save
  - :q! will quit vi, throwing away changes
- :w will save the current file you're editing
- :x (or :wq) will save the file and quit vi

More Files

- :w filename will write the contents to the given file name
- :sp filename will open another file in a split window
  - <ctrl>-w <ctrl>-w will switch between windows--all commands only affect the active window

UNIVERSITY OF DENVER

My dog closed my terminal!

- Don't worry, vi keeps a backup and will warn you when you try to edit the file again
- vi -r will list the files that can be recovered
- vi -r <filename> will recover the edited file when your terminal was closed

UNIVERSITY OF DENVER

Simple editing

- x deletes the character the cursor is on
- J (capitalized) joins the next line with the current line
- r followed by a character replaces the character under the cursor
- dd deletes the current line

More editing

- d plus a movement command will delete in that manner
  - dw deletes up to the next word, db the prior word
  - d) deletes to the end of the sentence, d} deletes to the end of the paragraph
- . will repeat a command. For example, dd. will delete two lines

Undo!

- Finally, the command u will undo prior edits
- <ctrl>-r will redo the last undo

Insert Mode

- To enter insert mode:
  - i will let you enter text at the cursor
  - a will move the cursor forward one and let you enter text
- To exit insert mode:
  - <esc> will put you back in command mode

More Insert mode

- You can now type in free-form, backspace, delete, and enter will work as expected
- More insert commands:
  - A will start appending text to the current line
  - I will start adding text at the start of the current line

Edit/Insert

- o will insert a blank line below your current line, move the cursor and place you in insert mode
- O will insert a blank line above your current line, move the cursor and put you in insert mode
- cc will replace the current line with a new line and put you in insert mode

Edits/inserts

- c0 deletes everything from the cursor to the start of the line and puts you in insert mode
- c$ deletes everything from the cursor to the end of the line and puts you in insert mode
- c (for change) is like the d command, you can use it with movements to edit and insert

Combos!

- We've seen two combo commands already: d and c
- Using a number prior to a command repeats it that many times
- 100dd deletes 100 lines, 12w jumps forward 12 words

Special c and d combos

- You can combine c and d with a number and command:
  - d2j will delete the current and next two lines
  - d3w will delete the next 3 words
  - Experiment!

Searching

- Sometimes you need to find something:
  - /<regexp> will search forward for the regular expression in the text and put the cursor on the expression
  - /<enter> will repeat the last search
  - ?<regexp> will search backwards

UNIVERSITY OF DENVER

Moving to visual mode

- Hitting 'v' will switch vi (on vim, elvis, gvim, etc versions) to visual mode. visual mode was added later, old versions of vi won't have it
- v will let text be highlighted through the use of movement commands
- Hitting 'v' while in visual mode will drop you back to command mode

Copy/Cut

- Once in visual mode:
  - y will copy the text (y for yank!)
  - d will cut the text
- Either of these commands will drop you back to command mode
- p/P will insert the text which was copied/cut from y or d either after/before the cursor

Replacing text

- Replacing text requires the ex-mode
  - :s/<regexp>/replacement/<enter>
  - :s/<regexp>/replacement/g will replace all occurrences on the line
  - :%s/<regexp>/replacement/g will replace all in the file
  - :%s/<regexp>/replacement/gc will ask for confirmation

Miscellaneous

- For source files, vi can auto-indent for you
  - If it's not on, type :set autoindent
  - <ctrl>-d will indent to the left one level
  - <ctrl>-t will indent to the right one level
  - :set tabstop=4 sets the indention to 4 per level

- :set sm will show matching {, (, or [when programming
- :syntax on will turn on syntax highlighting (ie, color)