Vi IMproved improved

The Guide to Make VIM Awesome

# whoami

This is me

- Jaime - @jgeigerm
- CSEC 4th year, total dork
- “You’re my faaaavorite” -Corinne
- KΔP, RC3, CCDC, SPARSA, CTF, WiC, SI, ACDC, WTF, R2D2
- I want to meet everyone in this room
- http://havefuninsi.de
- Bullet intentionally left blank

Vim is hard

- Large learning curve
  - So many alternatives - nano, sublime, atom
  - So many keys to remember! - C, A, i, r, R, hjkl, ^w
  - A whole new language to learn - :wq, :%s///, :bnext, :!
- Why bother?
  - It’s on every system ever
  - It’s easy to restore your configuration so it is the same everywhere with little setup

Modes

- Modes - the same keys do different things in different modes
  - Normal
  - :Command or set vimrc setting
    - With colon is a command mode command, without is .vimrc setting
  - Insert
  - Visual
- Commands and keys are color coded in this presentation
- Hit escape to get out of a mode back to normal mode

Basic Editing and Navigation

- i - insert
- I - insert at beginning of line
- A - append to line
- C - replace the rest of the line after the cursor
- r<char> - replace the letter under the cursor with <char>
- o/O - insert on new line below/above
- x - delete letter under cursor
- dd - delete whole line
    - d#d - delete # of lines
- ^w - delete word
- <</>> - decrease/increase indent

- arrows/hjkl - move cursor
- f<char> - forward search for <char>
- F<char> - backward search for <char>
- w/b - jump to beginning of the next/previous word
- e - jump to the end of a word
- ^u/^d - page up/down
- gg - top of file
- G - bottom of file
- % - jump to corresponding item (closing brace from opening brace and back)

Copying/Cutting/Pasting

- yy, cc - copy line (yank), cut line (and go into insert mode)
- p/P - paste before/after
- v - go into visual mode
  - ^v - visual block mode
  - V - visual line mode
- y - yank (copy) selection
- x - cut selection
- c - cut selection and then go into insert mode
- d - delete selection
- </> - decrease/increase indent of selection

Pasting from System Clipboard

- Things get messed up when you use the system paste
- :set paste - i - paste - :set nopaste
- nnoremap <key_to_map> :set nopaste!<CR>
    - .vimrc toggle for paste mode, I use <leader>p

Command Mode

- Set options - ex. :set number :set syntax :set wrap (line numbers, syntax highlighting, & line wrapping)
  - Unset by prefixing with no - ex. :set nowrap (turn off line wrap)
- Read, write, and open
  - :r file.txt, :w file.txt, :o file.txt
  - Tab completed, sometimes
- Run/read from/write to shell commands - :!ls -lah, :r !ls, :w !wc
- Run python - :python print(“hi”)
- Abbreviations work - :py print(“hi”), :set nu

Searching

- / - search down, ? - search up
- n/N - next result, previous result
- set ignorecase - ignores the case of the search
- set smartcase - won’t be case sensitive unless you are
- set hlsearch - highlights matches
- set incsearch - highlights match as you type (like web browser)
- nnoremap <silent> <C-l> :nohl<CR><C-l> - clear highlighting with Ctrl-l

Spaces, not tabs

- Tabs are annoying. Stop using them.
- `set expandtab` - uses spaces, not tabs
- `set shiftwidth=4` - << or >> shift this much, also auto indenting
- `set softtabstop=4` - number of spaces to use instead of tab
- `set tabstop=8` - if you do have a tab, this is how many columns it will take up
- `set smarttab` - aligns tab/backspace with the nearest shiftwidth

Buffers, not tabs

- Vim buffers are meant for editing different files
  - Use tabs as little as possible
    - One use case is different arrangement of buffers
- When used with vim-bufferline and ctrlp.vim plugins they become very efficient
- Use your leader and split, here’s how I do it …

Buffers, not tabs

- set hidden - allow unsaved buffers to be out of focus
- nnoremap <leader>. :bnext<CR> - next tab map to ,.
- nnoremap <leader>m :bprev<CR> - previous tab map to ,m
  - This makes more sense when you look at your keyboard
- nnoremap <leader>q :bnext <BAR> bd#<CR> - close current buffer*
- nnoremap <leader>t :enew<CR> - new buffer in new tab (scratch)
- nnoremap <leader>l :ls<CR> - lists buffers

~/.vimrc

- File where defaults for vim are defined
    - This is where the magic happens
- Define plugins, key combos, and every behavior of vim
- If you put it on github, you can just clone it and go
- First line should be:
    - `set nocompatible`

Splits

- Vertical and horizontal splitting
  - :split and :vsplit
- ^w^w - switch focussed split
- ^w^r - swap two panes
- ^w[h|j|k|l] - select pane [left|down|up|right]
- ^w[H|J|K|L] - move pane [left|down|up|right]
- Mouse support allows you to just drag to resize and click to select

Mapping Keys

- map/noremap - maps a command universally
- [n|v|i]map/[n|v|i]noremap - maps a command in a specific mode
- *noremap will map non-recursively, *map will map recursively
  - nmap j gg, nmap W j - W will recurse to gg
  - nnoremap j gg, nnoremap W j - will not recurse to gg will just be j
- Command mode commands can be mapped to keys
  - nmap <C-q> :wq<CR>
  - nnoremap <C-C> :bnext <BAR> bd#<CR>
    - Two commands on one line

Your Leader

- let mapleader = “,”
  - You can make it whatever you want
  - Can now be referred to as <leader> in .vimrc
  - Used for command mappings - one common key base
  - Combined with another key to run an action
- ex. nnoremap <leader>s :set nospell!<CR>
  - Spell check toggle with ,s (or whatever my leader is defined as)

Da Mouse

- You can scroll and click and highlight things with the mouse :O
- set mouse=a

Colors

- Run tput colors in your shell
  - If it isn’t 256 you’re doing it wrong
- set t_Co=256
- colorscheme <the colorscheme you chose>
- Bundle 'flazz/vim-colorschemes' - massive repo of colorschemes
- :colorscheme random - selects a random scheme and applies it
- When changing color schemes in a session remember that some settings carry over from the previously applied scheme
- Hack: set background=dark or set background=light

Swap/Undo Space

- Vim will create .swp files in case it quits unexpectedly
  - Current dir unless you tell it otherwise
- Vim will not save your history beyond exits
  - Unless you tell it otherwise - set undofile

```
if isdirectory(expand('$HOME/.vim-bak'))
    if &directory =~# '^\.,'
       set directory^=$HOME/.vim-bak/swap//
    endif
    if exists('+undodir') && &undodir =~# '^\.\%(,\|$\)'
        set undodir^=$HOME/.vim-bak/undo//
    endif
endif
```

HELP?!?!?

- :help <THE THING YOU WANT HELP WITH>
- At least the docs for vim are really, really good
- Google helps too
  - Usually if you want to be able to do it, it already exists

Bundles/Plugins

- vundle - plugin manager
  - mkdir -p ~/.vim/bundle
  - git clone https://github.com/gmarik/vundle ~/.vim/bundle/vundle
  - vim ~/.vimrc
    - Bundle ‘VundleVim/Vundle.vim’
    - Bundle ‘user/bundle’
      - user is github username, bundle is the repo name
  - :BundleUpdate to install/update bundles
- I use vundle, but you can also use pathogen or vam

Bundle: NerdTree

- In vim file browser
- Bundle 'scrooloose/nerdtree'

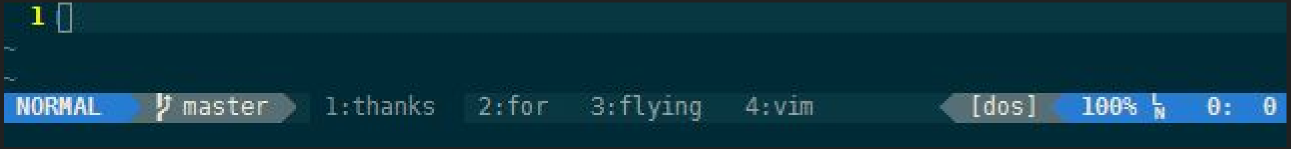
```
function! NERDTreeToggleOrFocus()
    if expand("%") =~ "NERD_tree"
        :NERDTreeToggle
    else
        call NERDTreeFocus()
    endif
endfunction
nmap <leader>n :call NERDTreeToggleOrFocus()<CR>
```

Bundle: delimitMate

- Brace, bracket, quote, and parentheses matcher
- Bundle 'Raimondi/delimitMate'

Bundles: vim-airline and vim-bufferline

- Awesome bottom and top bars, can be customized heavily
- I have never gotten it to look as cool as this…



```
 1 
~
~
NORMAL   master   1:thanks   2:for   3:flying   4:vim        [dos]  100%   0:  0
```

- Bundle 'bling/vim-airline’
- Bundle 'bling/vim-bufferline'

Bundle: syntastic

- In vim syntax checking for many languages
- Tells you the lines you have errors on
- Bundle 'scrooloose/syntastic'

Bundle: vim-gitgutter

- Shows you which lines have been added/deleted/modified relative to the currently active git repository
- Does it right in the gutter (near the numbers)
- Bundle 'airblade/vim-gitgutter'

Bundle: ctrlp.vim

- Fuzzy search for files in the current directory or git project
- Great for quickly switching between files in buffers
- Bundle 'kien/ctrlp.vim'
- let g:ctrlp_working_path_mode = 'r'
  - Entire git project will be included in search

Bundles: ultisnips and vim-snippets

- Snippets of code you can insert on the fly
- main + ctrl-g =

```
int main(int argc, char *argv[])
{
    | <- cursor starts here
    return 0;
}
```

- Lots of predefined snippets for many languages
- Bundle '[SirVer/ultisnips]'
- Bundle '[honza/vim-snippets]'

Bundles: python-mode and jedi-vim

- Python extensions for vim
- Auto-completion, special keys, etc.
  - <leader>b to insert python breakpoint!
- Research them more if you are interested, their configs can get long!
- Bundle 'davidhalter/jedi-vim'
- Bundle 'klen/python-mode'

Other Goodies

- :set spell - Spell checking!
- :%s/thing1/thing2/gc - regex find and replace
    - replaces thing1 with thing2 globally (g) and confirms each time (c)
- set number vs. set relativenumber
    - Two line number styles to fit your taste
- .vimrc can have comments with "
- Other bundles:
    - youcompleteme, tabular, vim-git, vim-easymotion, tagbar, vim-multiple-cursors, undotree, rainbow_parentheses

Example .vimrc

https://github.com/jgeigerm/vim-basic