Expert Code Grokking with Vim

Vim hacks for penetration testers

David Thiel

iSEC Partners

January 21, 2015

Outline

Hello.

Vim is a useful tool.

It excels at source review.

It takes a while to learn.

Hopefully, this will make it faster.

Prerequisites

- Vim or MacVim
- Basic vi knowledge
  - If you don't have this, try: `http://www.openvim.com/tutorial.html`
- Git, for fetching plugins
- A basic `vimrc` and a `.vim` (or `c:\users\username\vimfiles`) directory

The Basics

basics

```
syntax on                " enable syntax highlighting
filetype plugin on       " filetype detection
filetype indent on
set number               " line numbers
set hidden               " allow invisible buffers
set ignorecase           " case-insensitive searching
set smartcase            " but be smart about it
set hlsearch             " highlight all search matches
set incsearch            " search incrementally
set t_Co=256             " use 256 colors in the terminal
set et                   " expand tabs to spaces
set tags=./tags          " ctags - we'll get into that
set guioptions=aAegiM " Turn off the GUI. Now. Especially while learning.
```

Basic Vim Concepts

Modes

- Two main modes: *command* (aka "normal") and *insert*
  - Command mode is for giving instructions to vim.
- Other modes:
  - ex mode: Entered by typing "`:`", this is vim's "command line"
  - Visual: Visually select blocks of text, by using `Shift-V` (line mode) `CTRL-v`, (block mode), or "`v`" plus operators[1][2]

[1] http://vimdoc.sourceforge.net/htmldoc/visual.html#visual-operators

[2] http://stackoverflow.com/a/1218429

Vim’s hierarchy of buffers

- A “buffer” holds the contents of files.
- A “window” is a portal to a buffer.
- A “tab” is a container of windows.
  - You may think you want to only use tabs, but you don’t.
  - Avoid them until you master buffers and windows.

Vim's hierarchy of buffers

- A "buffer" holds the contents of files.
- A "window" is a portal to a buffer.
- A "tab" is a container of windows.
    - You may think you want to only use tabs, but you don't.
    - Avoid them until you master buffers and windows.

Vim’s hierarchy of buffers

- A “buffer” holds the contents of files.
- A “window” is a portal to a buffer.
- A “tab” is a container of windows.
    - You may think you want to only use tabs, but you don’t.
    - Avoid them until you master buffers and windows.

Vim's hierarchy of buffers

- A "buffer" holds the contents of files.
- A "window" is a portal to a buffer.
- A "tab" is a container of windows.
    - You may think you want to only use tabs, but you don't.
    - Avoid them until you master buffers and windows.

Vim's hierarchy of buffers

- A "buffer" holds the contents of files.
- A "window" is a portal to a buffer.
- A "tab" is a container of windows.
  - You may think you want to only use tabs, but you don't.
  - Avoid them until you master buffers and windows.

Basic Vim Concepts

Buffers

- Use `:ls` to show buffers. Many people also use a buffer manager such as BufTabs.
- Load a new buffer without viewing it with `:badd`.
- Note: The `:quit` command means to close a window. `:bdelete` / `:bd` means to delete/close a buffer.

Basic Vim Concepts

Windows

- Opening/closing windows:
  - `CTRL-w s` or `:split (filename)` — new horizontal split
  - `CTRL-w v` or `:vsplit (filename)` — new vertical split
  - `CTRL-w o` or `:only` — close all other windows
  - `CTRL-w c` — close window
- Navigating/moving windows:
  - `CTRL-w h/j/k/l` — change to window in that direction
  - `CTRL-w H/J/K/L` — move current window to (direction) side of the screen
  - `CTRL-w r` — rotate windows

Basic Vim Concepts

Jumps

- When you “jump” to another part of a file, your old position is stored in the jumplist
- Things that make jumps:
  - Jumping to a search result
  - Changing to a new buffer
  - Jumping to symbol definition
- Navigate your jumplist with `(CTRL-o)` and `(CTRL-i)`

Basic Vim Concepts

Changes

- Similar to jumps, but for lines that were changed
- See change list with `:changes`
- `g;` goes to the position of the last change
- `g,` goes back up the change list

Basic Vim Concepts
Named Registers

- Named registers can be used for storing lots of things
- IMO, the most useful is using registers for “complex repeats”, kind of an insta-macro
- Usually, Last change is repeated with “`.`”
- Complex repeats allow repeating very complex command sequences
- Usage:
  - `qa` starts recording into register “a”
  - Perform whatever complex sequence of commands and movements you feel like
  - When finished, hit `q` again.
  - To execute the contents of this register, call `@a`
  - `@@` to repeat the last executed register

Basic Vim Concepts

Other Registers

- There are also “numbered registers”
  - Used for remembering yanks/deletes
  - By default, register 1 has your most recent yank/delete
  - Access yank-before-last by `"2p`, and so on
- The `/` register: holds last search pattern
- The `_` register: blackhole — delete things to this register with `"_d` to have them not affect your delete/yank history
- There’s a lot more you can do with registers; check `http://blog.sanctum.geek.nz/advanced-vim-registers/`

Basic Vim Concepts

Undos

- Vim carries undo actions in a tree.
- You can make a change, undo it, make another change. You have now branched.
- This is hard to understand, but take my word for it.
- `u` and `(CTRL-r)` undo and redo along the *main branch*.
- `g+` and `g-` move *forward and backward in time*.
- Gundo can help you visualize this. See the screencast.

Basic Vim Concepts

Marks

- “marks” are pointers to specific locations in specific files.
- Simple usage:
    - `ma` to make mark “a”
    - `'a` to jump to the line mark “a” is on
    - `` `a `` to jump to the exact position of mark “a”
    - `:delm a` to delete mark “a”
    - Use “A” instead of “a” and this will make a cross-file mark — you can jump to it at any time, regardless of whether you’re editing that file at the moment
    - You can use motions with marks: `d'a` deletes from your current position to mark “a”
- But wait! There are other, cooler things.

Basic Vim Concepts

Marks

- Try using the `:marks` command.
- Note there are some special marks:
  - `.` — location of last change
  - `'` — the place you were before your last jump
  - `0` — the location and file you were at when you last quit vim (it’s a stack: you can also use 1-9)
- Check out `:help mark-motions` for more

Sessions and Views

Views

- Vim has a concept of "views", which specify where you last were in a file
- You can configure a filetype to do this thusly:

Saving and loading Python views

```
augroup python
au BufEnter *.py,*.pyw set smartindent smarttab nospell
au BufWinLeave *.py mkview
au BufWinEnter *.py silent loadview
augroup end
```

- Note that you can screw yourself up with this; if you notice some change in your vimrc isn't taking effect, try nuking the file in your viewdir.

Sessions and Views

Sessions

- This does save potentially sensitive data (filenames)
- I recommend storing it outside of your .vim directory:

Change viewdir

```
set viewdir=$HOME/.views
```

- You can also save your whole vim session state with `:mksession`
- This writes your state out to `Session.vim` in the cwd
- Includes all your open files, panels, etc
- Restore with `:so Session.vim`

Interlude

Putting things together

- visually select a word: `vaw`
- visually select a sentence: `vas`
- visually select a paragraph: `vap`
- visually select a block of C code: `va{`
- visually select from here to a search string: `v/someword`

- Now, think what happens if you use "`y`", "`c`" or "`d`" here...
- Before *or* after: `vapy` == `yap`
- You see how your life is changing now

Note: using "`i`" (meaning "inner") instead of "`a`" is probably more common in practice. Try both.

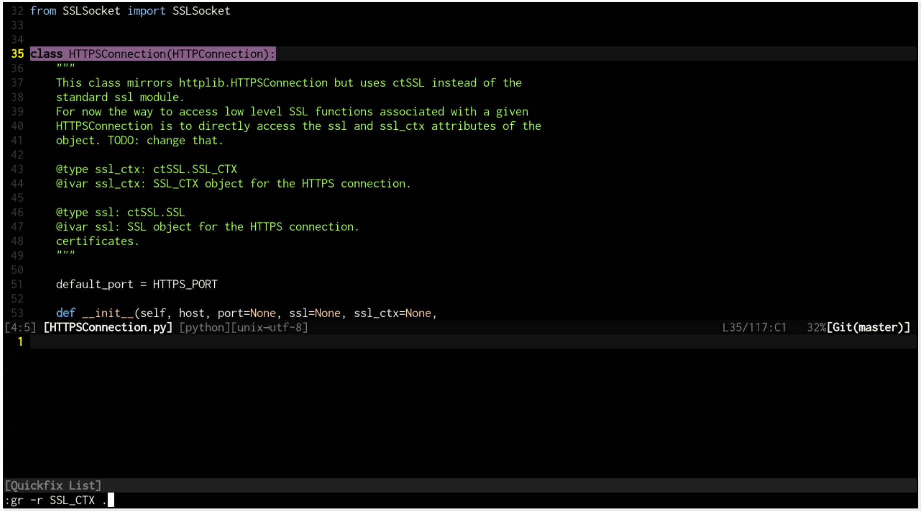
The Quickfix window

Your best friend

- Quickfix takes lists of files and line numbers and lets you jump among them
- Usage: `:copen` and `:cclose`
    - Or make a toggle, see my vimrc at the end
- Load a file into it with `:cf /tmp/filename`
- Use `:cn` to jump to next fix (you should map this)
- Another way to populate it is the `:grep` command

Quickfix

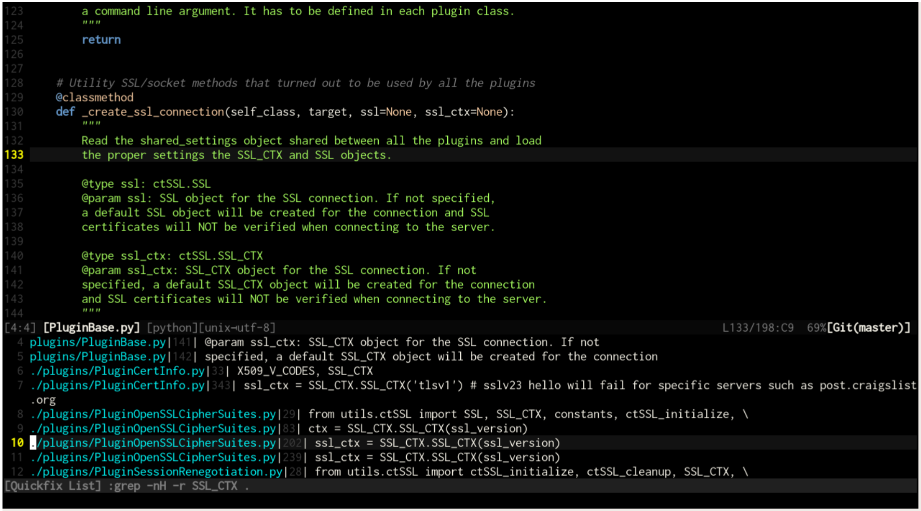
Using grep/vimgrep



```
from SSLSocket import SSLSocket


class HTTPSConnection(HTTPConnection):
    """
    This class mirrors httplib.HTTPSConnection but uses ctSSL instead of the
    standard ssl module.
    For now the way to access low level SSL functions associated with a given
    HTTPSConnection is to directly access the ssl and ssl_ctx attributes of the
    object. TODO: change that.

    @type ssl_ctx: ctSSL.SSL_CTX
    @ivar ssl_ctx: SSL_CTX object for the HTTPS connection.

    @type ssl: ctSSL.SSL
    @ivar ssl: SSL object for the HTTPS connection.
    certificates.
    """

    default_port = HTTPS_PORT

    def __init__(self, host, port=None, ssl=None, ssl_ctx=None,
[4:5] [HTTPSConnection.py] [python][unix-utf-8]                    L35/117:C1    32%[Git(master)]
  1

[Quickfix List]
:gr -r SSL_CTX .
```

Quickfix

Using grep/vimgrep



```
        a command line argument. It has to be defined in each plugin class.
        """
        return


    # Utility SSL/socket methods that turned out to be used by all the plugins
    @classmethod
    def _create_ssl_connection(self_class, target, ssl=None, ssl_ctx=None):
        """
        Read the shared_settings object shared between all the plugins and load
        the proper settings the SSL_CTX and SSL objects.

        @type ssl: ctSSL.SSL
        @param ssl: SSL object for the SSL connection. If not specified,
        a default SSL object will be created for the connection and SSL
        certificates will NOT be verified when connecting to the server.

        @type ssl_ctx: ctSSL.SSL_CTX
        @param ssl_ctx: SSL_CTX object for the SSL connection. If not
        specified, a default SSL_CTX object will be created for the connection
        and SSL certificates will NOT be verified when connecting to the server.
        """
[4:4] [PluginBase.py] [python][unix-utf-8]                                     L133/198:C9  69%[Git(master)]
plugins/PluginBase.py|141| @param ssl_ctx: SSL_CTX object for the SSL connection. If not
plugins/PluginBase.py|142| specified, a default SSL_CTX object will be created for the connection
./plugins/PluginCertInfo.py|33| X509_V_CODES, SSL_CTX
./plugins/PluginCertInfo.py|343| ssl_ctx = SSL_CTX.SSL_CTX('tlsv1') # sslv23 hello will fail for specific servers such as post.craigslist
.org
./plugins/PluginOpenSSLCipherSuites.py|29| from utils.ctSSL import SSL, SSL_CTX, constants, ctSSL_initialize, \
./plugins/PluginOpenSSLCipherSuites.py|83| ctx = SSL_CTX.SSL_CTX(ssl_version)
./plugins/PluginOpenSSLCipherSuites.py|202| ssl_ctx = SSL_CTX.SSL_CTX(ssl_version)
./plugins/PluginOpenSSLCipherSuites.py|239| ssl_ctx = SSL_CTX.SSL_CTX(ssl_version)
./plugins/PluginSessionRenegotiation.py|28| from utils.ctSSL import ctSSL_initialize, ctSSL_cleanup, SSL_CTX, \
[Quickfix List] :grep -nH -r SSL_CTX .
```

File Browsing

netrw

- Previously, I had recommended NERDTree for this task
- But I have been enlightened
- Just use splits and the :Explore family
- Usage:
  - `:Explore` — open a file browser in your current window (if unmodified — otherwise, split first)
  - `:Sexplore` — open a browser in a split (horizontal)
  - `:Vexplore` — open a browser in a split (left-vertical)
  - And so on, `:help Explore`
- Use `mb` bookmark files for later examination or common use
- `qb` to query bookmarks
- See my .vimrc at the end for a function to toggle :Vexplore on and off like a file drawer

Quick file opening

Ctrl-P

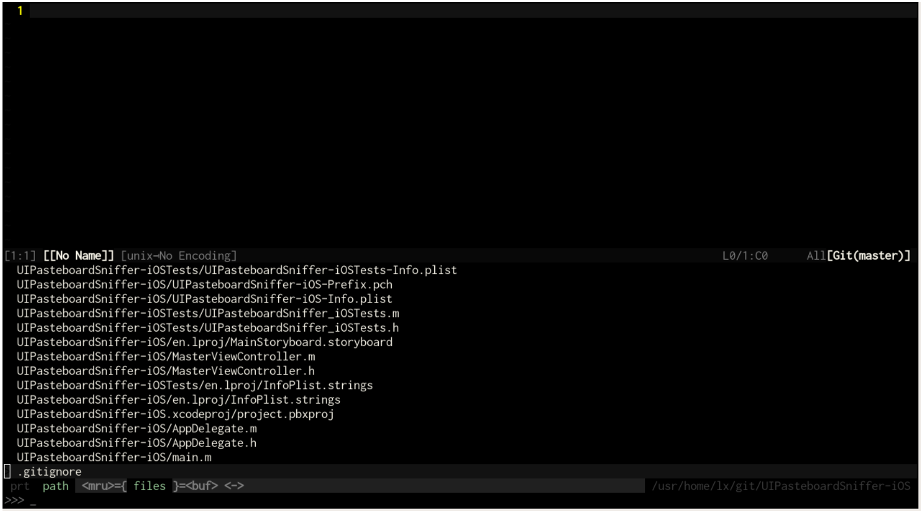
- Sometimes, you don't want to browse, you want a certain file
- What if I have a giant codebase and don't know where my file is?
- When you know what you want, regardless of where or what it is, use Ctrl-P:

`https://github.com/kien/ctrlp.vim`

- Can also search open buffers, recently used files, etc.

Quick file opening

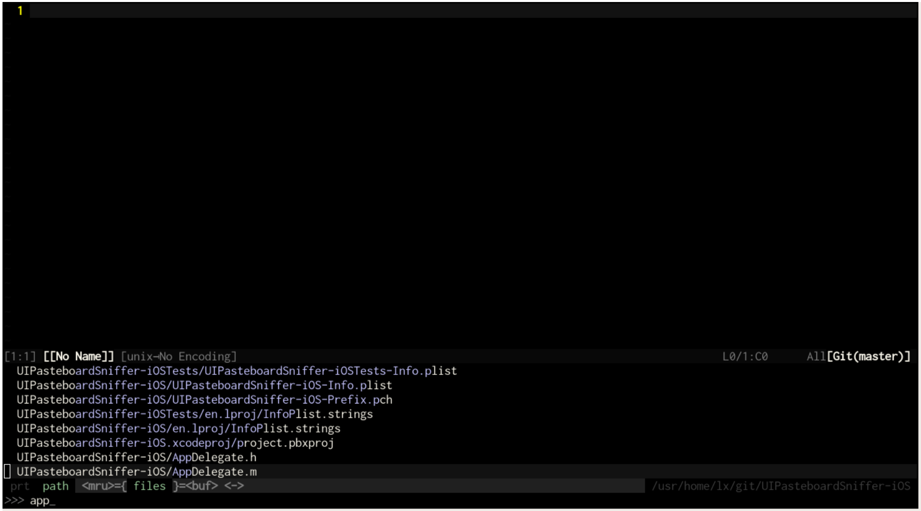
Ctrl-P in action



```
  1

[1:1] [[No Name]] [unix-No Encoding]                                   L0/1:C0        All[Git(master)]
  UIPasteboardSniffer-iOSTests/UIPasteboardSniffer-iOSTests-Info.plist
  UIPasteboardSniffer-iOS/UIPasteboardSniffer-iOS-Prefix.pch
  UIPasteboardSniffer-iOS/UIPasteboardSniffer-iOS-Info.plist
  UIPasteboardSniffer-iOSTests/UIPasteboardSniffer_iOSTests.m
  UIPasteboardSniffer-iOSTests/UIPasteboardSniffer_iOSTests.h
  UIPasteboardSniffer-iOS/en.lproj/MainStoryboard.storyboard
  UIPasteboardSniffer-iOS/MasterViewController.m
  UIPasteboardSniffer-iOS/MasterViewController.h
  UIPasteboardSniffer-iOSTests/en.lproj/InfoPlist.strings
  UIPasteboardSniffer-iOS/en.lproj/InfoPlist.strings
  UIPasteboardSniffer-iOS.xcodeproj/project.pbxproj
  UIPasteboardSniffer-iOS/AppDelegate.m
  UIPasteboardSniffer-iOS/AppDelegate.h
  UIPasteboardSniffer-iOS/main.m
> .gitignore
 prt  path  <mru>={ files }=<buf> <->                              /usr/home/lx/git/UIPasteboardSniffer-iOS
>>> _
```

Quick file opening

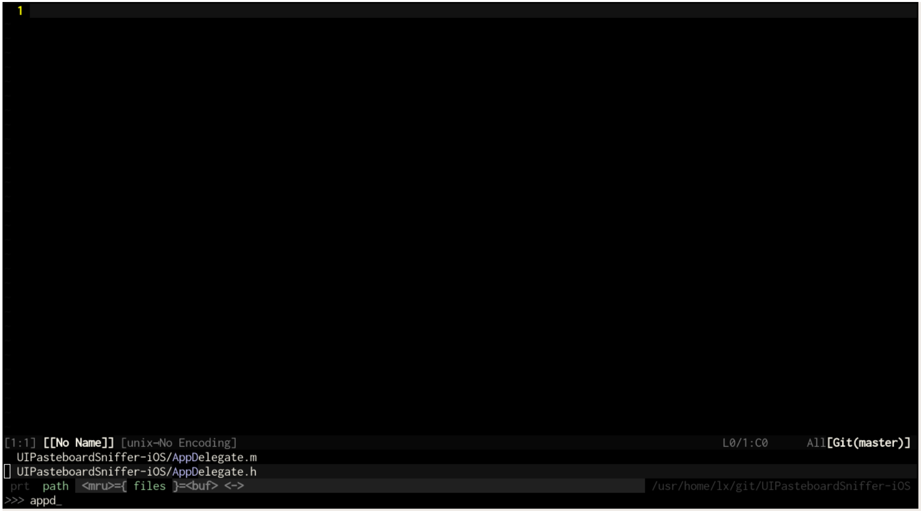
Ctrl-P in action



```
[1:1] [[No Name]] [unix-No Encoding]                                              L0/1:C0      All[Git(master)]
  UIPasteboardSniffer-iOSTests/UIPasteboardSniffer-iOSTests-Info.plist
  UIPasteboardSniffer-iOS/UIPasteboardSniffer-iOS-Info.plist
  UIPasteboardSniffer-iOS/UIPasteboardSniffer-iOS-Prefix.pch
  UIPasteboardSniffer-iOSTests/en.lproj/InfoPlist.strings
  UIPasteboardSniffer-iOS/en.lproj/InfoPlist.strings
  UIPasteboardSniffer-iOS.xcodeproj/project.pbxproj
  UIPasteboardSniffer-iOS/AppDelegate.h
  UIPasteboardSniffer-iOS/AppDelegate.m
 prt  path  <mru>={ files }=<buf> <->                          /usr/home/lx/git/UIPasteboardSniffer-iOS
>>> app_
```

Quick file opening

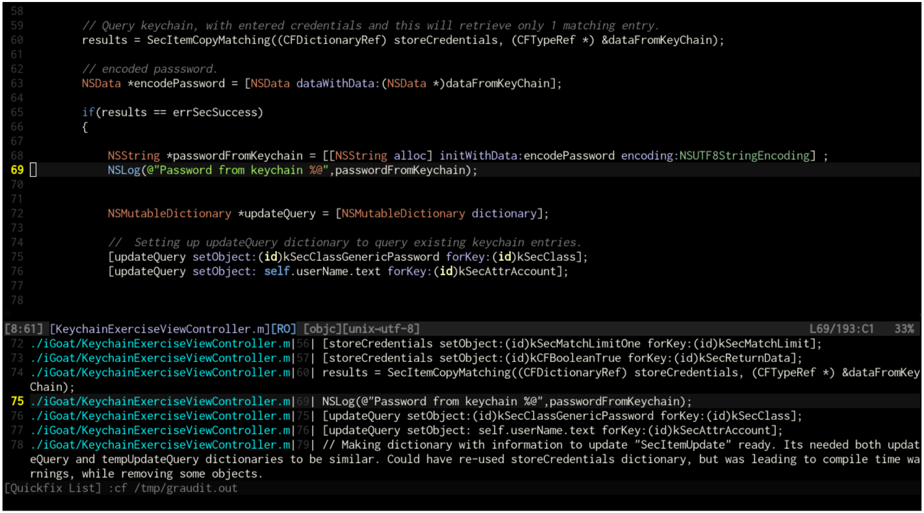
Ctrl-P in action



```
  1
[1:1] [[No Name]] [unix-No Encoding]                                                          L0/1:C0      All[Git(master)]
  UIPasteboardSniffer-iOS/AppDelegate.m
  UIPasteboardSniffer-iOS/AppDelegate.h
 prt  path  <mru>={ files }=<buf> <->                                       /usr/home/lx/git/UIPasteboardSniffer-iOS
>>> appd_
```

graudit

graudit in quickfix



```

        // Query keychain, with entered credentials and this will retrieve only 1 matching entry.
        results = SecItemCopyMatching((CFDictionaryRef) storeCredentials, (CFTypeRef *) &dataFromKeyChain);

        // encoded password.
        NSData *encodePassword = [NSData dataWithData:(NSData *)dataFromKeyChain];

        if(results == errSecSuccess)
        {

            NSString *passwordFromKeychain = [[NSString alloc] initWithData:encodePassword encoding:NSUTF8StringEncoding] ;
            NSLog(@"Password from keychain %@",passwordFromKeychain);


            NSMutableDictionary *updateQuery = [NSMutableDictionary dictionary];

            //  Setting up updateQuery dictionary to query existing keychain entries.
            [updateQuery setObject:(id)kSecClassGenericPassword forKey:(id)kSecClass];
            [updateQuery setObject: self.userName.text forKey:(id)kSecAttrAccount];


[8:61] [KeychainExerciseViewController.m][RO] [objc][unix-utf-8]                                   L69/193:C1    33%
./iGoat/KeychainExerciseViewController.m|56| [storeCredentials setObject:(id)kSecMatchLimitOne forKey:(id)kSecMatchLimit];
./iGoat/KeychainExerciseViewController.m|57| [storeCredentials setObject:(id)kCFBooleanTrue forKey:(id)kSecReturnData];
./iGoat/KeychainExerciseViewController.m|60| results = SecItemCopyMatching((CFDictionaryRef) storeCredentials, (CFTypeRef *) &dataFromKey
Chain);
./iGoat/KeychainExerciseViewController.m|69| NSLog(@"Password from keychain %@",passwordFromKeychain);
./iGoat/KeychainExerciseViewController.m|75| [updateQuery setObject:(id)kSecClassGenericPassword forKey:(id)kSecClass];
./iGoat/KeychainExerciseViewController.m|76| [updateQuery setObject: self.userName.text forKey:(id)kSecAttrAccount];
./iGoat/KeychainExerciseViewController.m|79| // Making dictionary with information to update "SecItemUpdate" ready. Its needed both updat
eQuery and tempUpdateQuery dictionaries to be similar. Could have re-used storeCredentials dictionary, but was leading to compile time wa
rnings, while removing some objects.
[Quickfix List] :cf /tmp/graudit.out
```

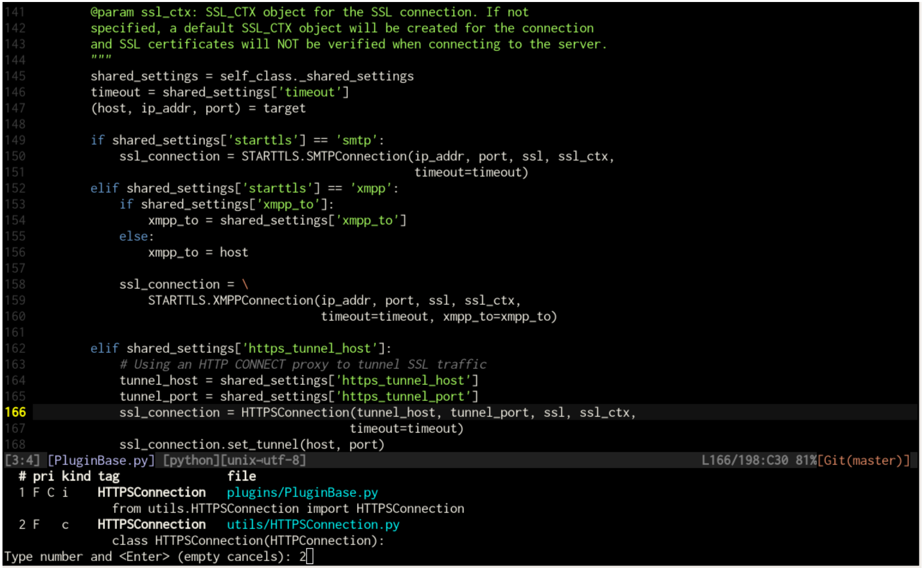
Exuberant Ctags

Tokenize all the things

`http://ctags.sourceforge.net/`

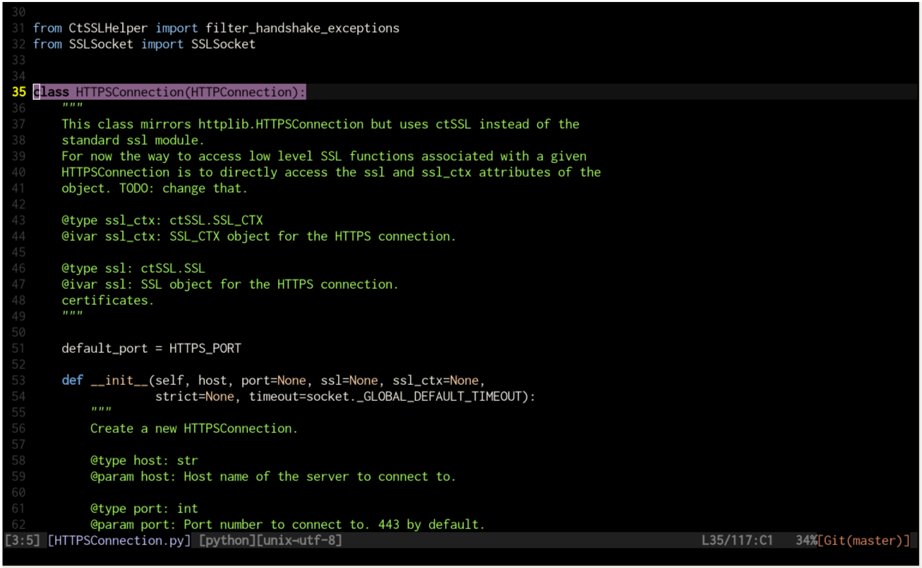
- Generates an index of symbols
- Usage: at root of source tree (on command line), `ctags -R .`
- In vim, `:set tags=./tags` or whatever path you choose
- When cursor is on a function/method/whatever, `(CTRL-])` jumps to its definition
  - If ambiguous, a select list is displayed
- Return to your previous location with `(CTRL-t)`
- To open the tag in a "Preview" window, use `(CTRL-w ])`
  - Close with `(CTRL-w z)`

Ctags usage: jump to definition



```
        @param ssl_ctx: SSL_CTX object for the SSL connection. If not
        specified, a default SSL_CTX object will be created for the connection
        and SSL certificates will NOT be verified when connecting to the server.
        """
        shared_settings = self_class._shared_settings
        timeout = shared_settings['timeout']
        (host, ip_addr, port) = target

        if shared_settings['starttls'] == 'smtp':
            ssl_connection = STARTTLS.SMTPConnection(ip_addr, port, ssl, ssl_ctx,
                                                     timeout=timeout)
        elif shared_settings['starttls'] == 'xmpp':
            if shared_settings['xmpp_to']:
                xmpp_to = shared_settings['xmpp_to']
            else:
                xmpp_to = host

            ssl_connection = \
                STARTTLS.XMPPConnection(ip_addr, port, ssl, ssl_ctx,
                                        timeout=timeout, xmpp_to=xmpp_to)

        elif shared_settings['https_tunnel_host']:
            # Using an HTTP CONNECT proxy to tunnel SSL traffic
            tunnel_host = shared_settings['https_tunnel_host']
            tunnel_port = shared_settings['https_tunnel_port']
            ssl_connection = HTTPSConnection(tunnel_host, tunnel_port, ssl, ssl_ctx,
                                            timeout=timeout)
            ssl_connection.set_tunnel(host, port)
[3:4] [PluginBase.py] [python][unix-utf-8]                              L166/198:C30 81%[Git(master)]
  # pri kind tag               file
  1 F C i    HTTPSConnection   plugins/PluginBase.py
               from utils.HTTPSConnection import HTTPSConnection
  2 F   c    HTTPSConnection   utils/HTTPSConnection.py
               class HTTPSConnection(HTTPConnection):
Type number and <Enter> (empty cancels): 2
```

Ctags usage: successful symbol lookup



```
from CtSSLHelper import filter_handshake_exceptions
from SSLSocket import SSLSocket


class HTTPSConnection(HTTPConnection):
    """
    This class mirrors httplib.HTTPSConnection but uses ctSSL instead of the
    standard ssl module.
    For now the way to access low level SSL functions associated with a given
    HTTPSConnection is to directly access the ssl and ssl_ctx attributes of the
    object. TODO: change that.

    @type ssl_ctx: ctSSL.SSL_CTX
    @ivar ssl_ctx: SSL_CTX object for the HTTPS connection.

    @type ssl: ctSSL.SSL
    @ivar ssl: SSL object for the HTTPS connection.
    certificates.
    """

    default_port = HTTPS_PORT

    def __init__(self, host, port=None, ssl=None, ssl_ctx=None,
                 strict=None, timeout=socket._GLOBAL_DEFAULT_TIMEOUT):
        """
        Create a new HTTPSConnection.

        @type host: str
        @param host: Host name of the server to connect to.

        @type port: int
        @param port: Port number to connect to. 443 by default.
[3:5] [HTTPSConnection.py] [python][unix-utf-8]                  L35/117:C1    34%[Git(master)]
```

Tagbar

Intelligent symbol browsing

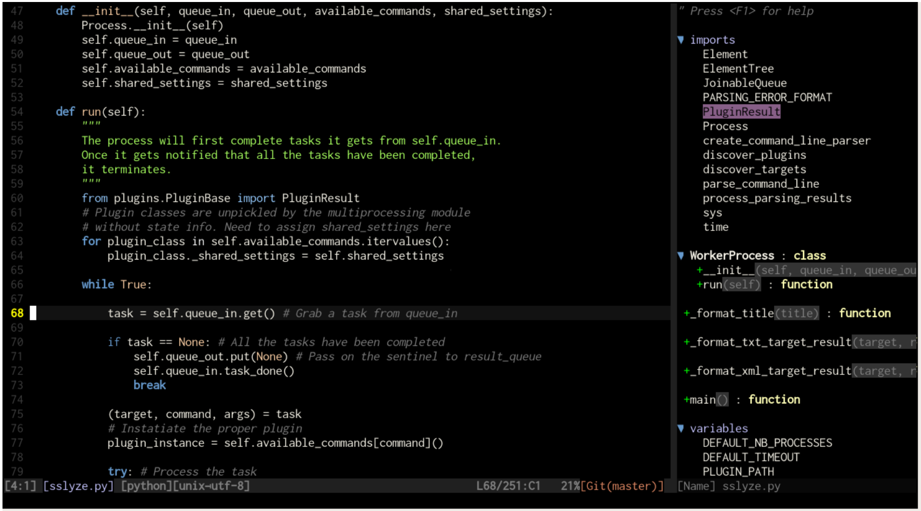
- Uses ctags to generate a symbol list
- Smart about identifying different symbol types
- Scope-aware
- Displays basic function signatures

TagBar mappings

```
map <silent> <F10> :TagbarToggle<CR>
nnoremap <silent> <F10> :TagbarToggle<CR>
```

Tagbar

Example Java tagbar

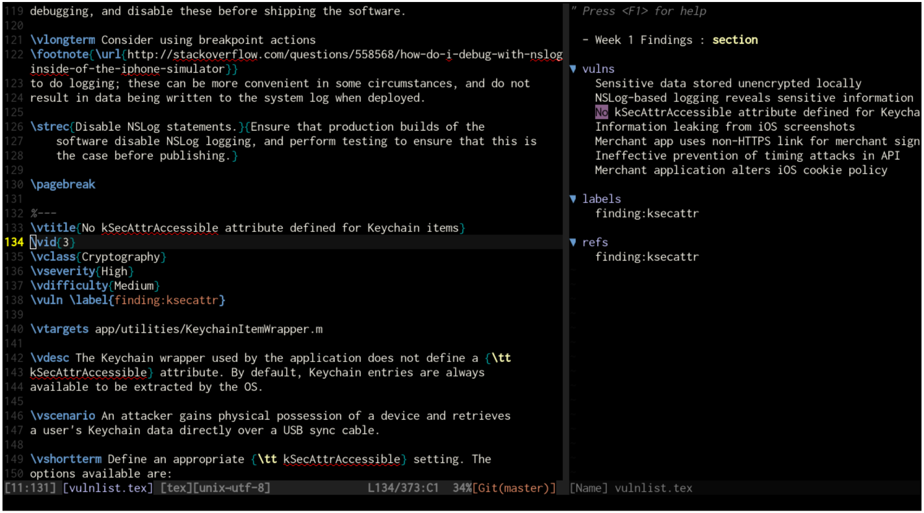


```
    def __init__(self, queue_in, queue_out, available_commands, shared_settings):
        Process.__init__(self)
        self.queue_in = queue_in
        self.queue_out = queue_out
        self.available_commands = available_commands
        self.shared_settings = shared_settings

    def run(self):
        """
        The process will first complete tasks it gets from self.queue_in.
        Once it gets notified that all the tasks have been completed,
        it terminates.
        """
        from plugins.PluginBase import PluginResult
        # Plugin classes are unpickled by the multiprocessing module
        # without state info. Need to assign shared_settings here
        for plugin_class in self.available_commands.itervalues():
            plugin_class._shared_settings = self.shared_settings

        while True:

            task = self.queue_in.get() # Grab a task from queue_in

            if task == None: # All the tasks have been completed
                self.queue_out.put(None) # Pass on the sentinel to result_queue
                self.queue_in.task_done()
                break

            (target, command, args) = task
            # Instatiate the proper plugin
            plugin_instance = self.available_commands[command]()

            try: # Process the task
[4:1] [sslyze.py] [python][unix-utf-8]                  L68/251:C1    21%[Git(master)]
```

```
" Press <F1> for help

▼ imports
    Element
    ElementTree
    JoinableQueue
    PARSING_ERROR_FORMAT
    PluginResult
    Process
    create_command_line_parser
    discover_plugins
    discover_targets
    parse_command_line
    process_parsing_results
    sys
    time

▼ WorkerProcess : class
   +__init__(self, queue_in, queue_ou
   +run(self) : function

 +_format_title(title) : function

 +_format_txt_target_result(target, r

 +_format_xml_target_result(target, r

 +main() : function

▼ variables
    DEFAULT_NB_PROCESSES
    DEFAULT_TIMEOUT
    PLUGIN_PATH
[Name] sslyze.py
```

Tagbar

Customized iSEC report Tagbar



```
debugging, and disable these before shipping the software.

\vlongterm Consider using breakpoint actions
\footnote{\url{http://stackoverflow.com/questions/558568/how-do-i-debug-with-nslog
inside-of-the-iphone-simulator}}
to do logging; these can be more convenient in some circumstances, and do not
result in data being written to the system log when deployed.

\strec{Disable NSLog statements.}{Ensure that production builds of the
    software disable NSLog logging, and perform testing to ensure that this is
    the case before publishing.}

\pagebreak

%---
\vtitle{No kSecAttrAccessible attribute defined for Keychain items}
\vid{3}
\vclass{Cryptography}
\vseverity{High}
\vdifficulty{Medium}
\vuln \label{finding:ksecattr}

\vtargets app/utilities/KeychainItemWrapper.m

\vdesc The Keychain wrapper used by the application does not define a {\tt
kSecAttrAccessible} attribute. By default, Keychain entries are always
available to be extracted by the OS.

\vscenario An attacker gains physical possession of a device and retrieves
a user's Keychain data directly over a USB sync cable.

\vshortterm Define an appropriate {\tt kSecAttrAccessible} setting. The
options available are:
[11:131] [vulnlist.tex] [tex][unix-utf-8]          L134/373:C1   34%[Git(master)]
```

```
" Press <F1> for help

 - Week 1 Findings : section

▼ vulns
   Sensitive data stored unencrypted locally
   NSLog-based logging reveals sensitive information
   No kSecAttrAccessible attribute defined for Keycha
   Information leaking from iOS screenshots
   Merchant app uses non-HTTPS link for merchant sign
   Ineffective prevention of timing attacks in API
   Merchant application alters iOS cookie policy

▼ labels
   finding:ksecattr

▼ refs
   finding:ksecattr

[Name] vulnlist.tex
```

Cscope

Serious symbol mangling

`http://cscope.sourceforge.net/cscope_vim_tutorial.html`

- Similar to ctags, but lets you do much more:
  - Show all callers of a function
  - Show all instances of a tag
  - Open include files
- Usage: `find . -name "*.[ch]" > cscope.files && cscope -b`
- Or whatever type of file you want to index

Cscope

Cheat sheet

- Install cscope_maps.vim from the tutorial
- Query a symbol with `(CTRL-\)` followed by one of the below:

```
cscope query types
"   's'   symbol: find all references to the token under cursor
"   'g'   global: find global definition(s) of the token under cursor
"   'c'   calls:  find all calls to the function name under cursor
"   't'   text:   find all instances of the text under cursor
"   'e'   egrep:  egrep search for the word under cursor
"   'f'   file:   open the filename under cursor
"   'i'   includes: find files that include the filename under cursor
"   'd'   called: find functions that function under cursor calls
```

Cscope
Configuration

- By default, a numbered list pops up, similar to multiple matches for ctags
- I prefer to have most of them be in the quickfix list, so:

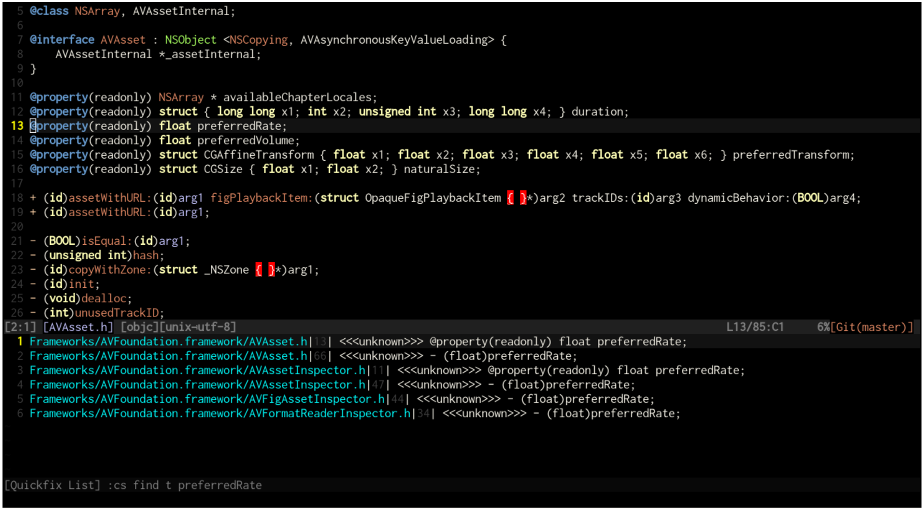
cscope quickfix types

```
set cscopequickfix=s-,c-,d-,i-,t-,e-
```

- "-" means to make a new quickfix list, "+" means to append

Cscope

Querytype “t” in the quickfix



```
@class NSArray, AVAssetInternal;

@interface AVAsset : NSObject <NSCopying, AVAsynchronousKeyValueLoading> {
    AVAssetInternal *_assetInternal;
}

@property(readonly) NSArray * availableChapterLocales;
@property(readonly) struct { long long x1; int x2; unsigned int x3; long long x4; } duration;
@property(readonly) float preferredRate;
@property(readonly) float preferredVolume;
@property(readonly) struct CGAffineTransform { float x1; float x2; float x3; float x4; float x5; float x6; } preferredTransform;
@property(readonly) struct CGSize { float x1; float x2; } naturalSize;

+ (id)assetWithURL:(id)arg1 figPlaybackItem:(struct OpaqueFigPlaybackItem { }*)arg2 trackIDs:(id)arg3 dynamicBehavior:(BOOL)arg4;
+ (id)assetWithURL:(id)arg1;

- (BOOL)isEqual:(id)arg1;
- (unsigned int)hash;
- (id)copyWithZone:(struct _NSZone { }*)arg1;
- (id)init;
- (void)dealloc;
- (int)unusedTrackID;
[2:1] [AVAsset.h] [objc][unix-utf-8]                                                    L13/85:C1      6%[Git(master)]
1 Frameworks/AVFoundation.framework/AVAsset.h|13| <<<unknown>>> @property(readonly) float preferredRate;
2 Frameworks/AVFoundation.framework/AVAsset.h|66| <<<unknown>>> - (float)preferredRate;
3 Frameworks/AVFoundation.framework/AVAssetInspector.h|11| <<<unknown>>> @property(readonly) float preferredRate;
4 Frameworks/AVFoundation.framework/AVAssetInspector.h|47| <<<unknown>>> - (float)preferredRate;
5 Frameworks/AVFoundation.framework/AVFigAssetInspector.h|44| <<<unknown>>> - (float)preferredRate;
6 Frameworks/AVFoundation.framework/AVFormatReaderInspector.h|34| <<<unknown>>> - (float)preferredRate;

[Quickfix List] :cs find t preferredRate
```

graudit
Mega-securitygrep

- Simple third-party grep wrapper, works like rats or flawfinder
- But we have good dbs for it
- Usage:

```
graudit -d objc -c0 -z . > /tmp/graudit.out
```

- `:copen`, `:cf /tmp/graudit.out`

CCTree

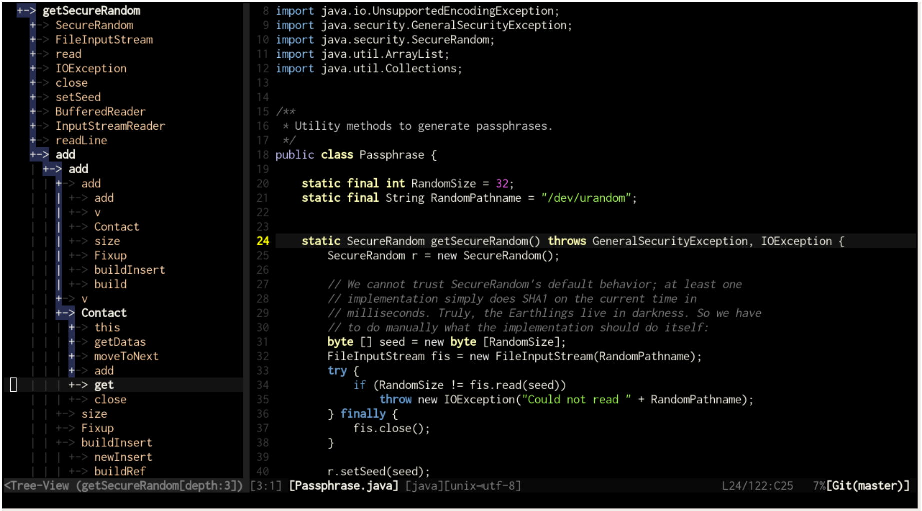
Now we get all crazy

`http://www.vim.org/scripts/script.php?script_id=2368`

- Builds fancy database from cscope output
- Creates symbol call trees
- Can do forward or reverse
- Usage: `:CCTreeLoadDB`, select your cscope.out file
- Has some keybindings for generating graphs that I always forget, so just use `:CCTreeTraceForward` and `:CCTreeTraceReverse`

CCTree

:CCTreeTraceForward

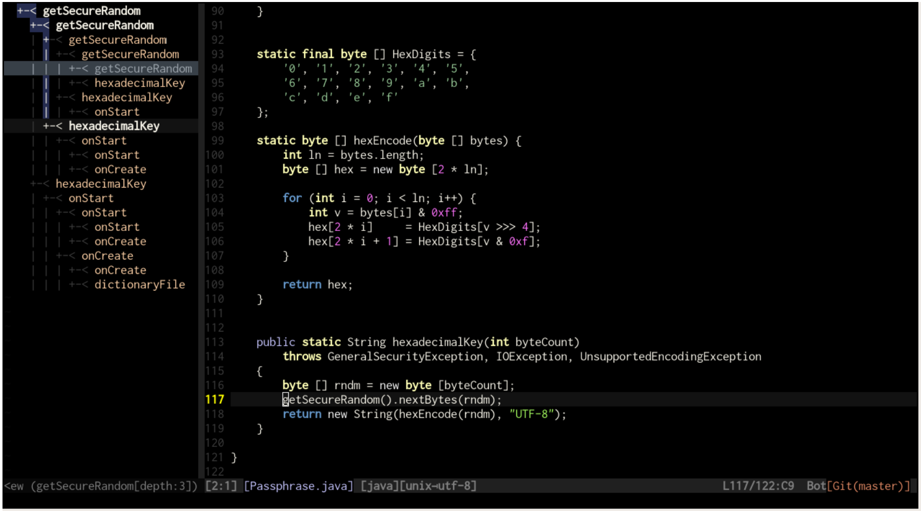


```
+-> getSecureRandom
  +-> SecureRandom
  +-> FileInputStream
  +-> read
  +-> IOException
  +-> close
  +-> setSeed
  +-> BufferedReader
  +-> InputStreamReader
  +-> readLine
  +-> add
  |  +-> add
  |  |  +-> add
  |  |  |  +-> add
  |  |  |  +-> v
  |  |  |  +-> Contact
  |  |  |  +-> size
  |  |  |  +-> Fixup
  |  |  |  +-> buildInsert
  |  |  |  +-> build
  |  |  +-> v
  |  |  +-> Contact
  |  |  |  +-> this
  |  |  |  +-> getDatas
  |  |  |  +-> moveToNext
  |  |  |  +-> add
  |  |  |  +-> get
  |  |  |  +-> close
  |  |  +-> size
  |  |  +-> Fixup
  |  |  +-> buildInsert
  |  |  |  +-> newInsert
  |  |  |  +-> buildRef
<Tree-View (getSecureRandom[depth:3])
```

```java
import java.io.UnsupportedEncodingException;
import java.security.GeneralSecurityException;
import java.security.SecureRandom;
import java.util.ArrayList;
import java.util.Collections;


/**
 * Utility methods to generate passphrases.
 */
public class Passphrase {

    static final int RandomSize = 32;
    static final String RandomPathname = "/dev/urandom";


    static SecureRandom getSecureRandom() throws GeneralSecurityException, IOException {
        SecureRandom r = new SecureRandom();

        // We cannot trust SecureRandom's default behavior; at least one
        // implementation simply does SHA1 on the current time in
        // milliseconds. Truly, the Earthlings live in darkness. So we have
        // to do manually what the implementation should do itself:
        byte [] seed = new byte [RandomSize];
        FileInputStream fis = new FileInputStream(RandomPathname);
        try {
            if (RandomSize != fis.read(seed))
                throw new IOException("Could not read " + RandomPathname);
        } finally {
            fis.close();
        }

        r.setSeed(seed);
[3:1] [Passphrase.java] [java][unix-utf-8]                    L24/122:C25   7%[Git(master)]
```

CCTree

:CCTreeTraceReverse



```
+-< getSecureRandom
  +-< getSecureRandom
  | +-< getSecureRandom
  | | +-< getSecureRandom
  | | | +-< getSecureRandom
  | | | +-< hexadecimalKey
  | | +-< hexadecimalKey
  | | | +-< onStart
  | +-< hexadecimalKey
  | | +-< onStart
  | | | +-< onStart
  | | | +-< onCreate
  +-< hexadecimalKey
  | +-< onStart
  | | +-< onStart
  | | | +-< onStart
  | | | +-< onCreate
  | | +-< onCreate
  | | | +-< onCreate
  | | | +-< dictionaryFile
```

```java
    }


    static final byte [] HexDigits = {
        '0', '1', '2', '3', '4', '5',
        '6', '7', '8', '9', 'a', 'b',
        'c', 'd', 'e', 'f'
    };

    static byte [] hexEncode(byte [] bytes) {
        int ln = bytes.length;
        byte [] hex = new byte [2 * ln];

        for (int i = 0; i < ln; i++) {
            int v = bytes[i] & 0xff;
            hex[2 * i]     = HexDigits[v >>> 4];
            hex[2 * i + 1] = HexDigits[v & 0xf];
        }

        return hex;
    }


    public static String hexadecimalKey(int byteCount)
        throws GeneralSecurityException, IOException, UnsupportedEncodingException
    {
        byte [] rndm = new byte [byteCount];
        getSecureRandom().nextBytes(rndm);
        return new String(hexEncode(rndm), "UTF-8");
    }

}
```

```
<ew (getSecureRandom[depth:3]) [2:1] [Passphrase.java] [java][unix-utf-8]          L117/122:C9  Bot[Git(master)]
```

SuperTab

Cruise control for completion

https://github.com/ervandew/supertab

- Vim's default code completion mechanism is rather complex
- Ties hands into knots, requires multiple keystrokes
- Worth looking into for some things, *e.g.* CTRL-X CTRL-L
- SuperTab makes it just work with tab
- Obviously that can cause problems sometimes...

Supertab config

```
let g:SuperTabContextFileTypeExclusions = ['make']
```

LaTeX-Box

Lightweight LaTeX in vim

https://github.com/LaTeX-Box-Team/LaTeX-Box

- Considerably less weird than vim-LaTeX
- Provides basic niceties like:
  - Background compilation with latexmk
  - begin/end block matching
  - Smart indentation
  - Completion for basic LaTeX environments

LATEX-Box config

```
                              Latex-Box
set wildignore+=*.idx,~*,*.aux,*.dvi,*.bbl,*.blg,*.orig,*.toc,*.fls,*.ind
set wildignore+=*.loc,*.gz,*.latexmain,*.tv,*.ilg,*.lltr,*.lov,*.lstr

imap <buffer> [[ \begin{
imap <buffer> ]] <Plug>LatexCloseCurEnv

let g:LatexBox_latexmk_options = "-pdflatex=lualatex"
let g:LatexBox_viewer = "evince"
augroup latex
    au BufEnter *.tex,*.sty set spell filetype=tex
    au BufEnter *.tex,*.sty set textwidth=78 smartindent
    au BufEnter *.tex,*.sty syntax spell toplevel
    au BufWinLeave *.tex,*.sty mkview
    au BufWinEnter *.tex,*.sty silent loadview
augroup end
```

SnipMate

You will forget how to program forever

`https://github.com/msanders/snipmate.vim`

`https://github.com/honza/snipmate-snippets`

- Spits out code snippets from abbreviations
- Easy to define new ones
- I recommend using the original SnipMate plus the expanded snippets library, as listed above
- Though if you’re starting from scratch, you might consider using ultisnips: `https://github.com/SirVer/ultisnips`

SnipMate

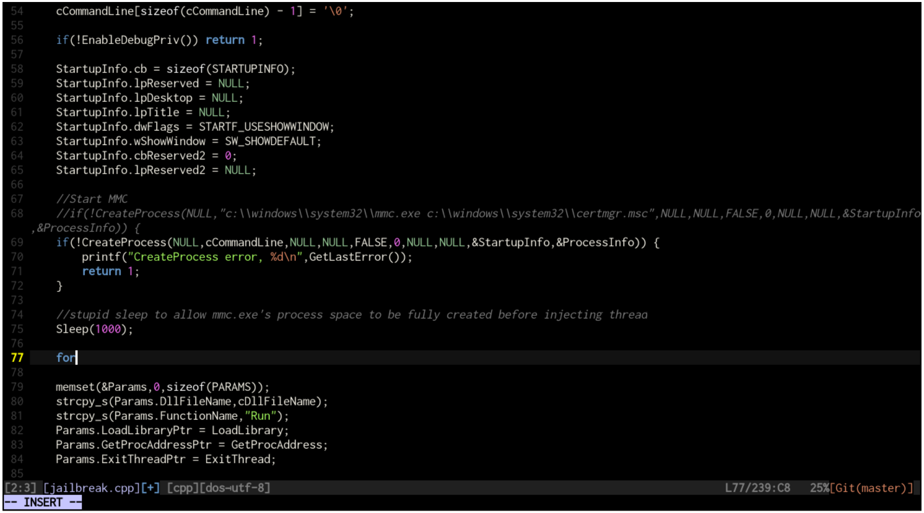
Example snippets

tex.snippets

```
snippet fig
\begin{figure}[ht]
\center \includegraphics[width=1.0\textwidth]{images/${1}}
\caption{${2}}
\end{figure}

snippet lst
\begin{lstlisting}[style=code,language=${1},numbers=none,caption={${2}}]

\end{lstlisting}
```

SnipMate

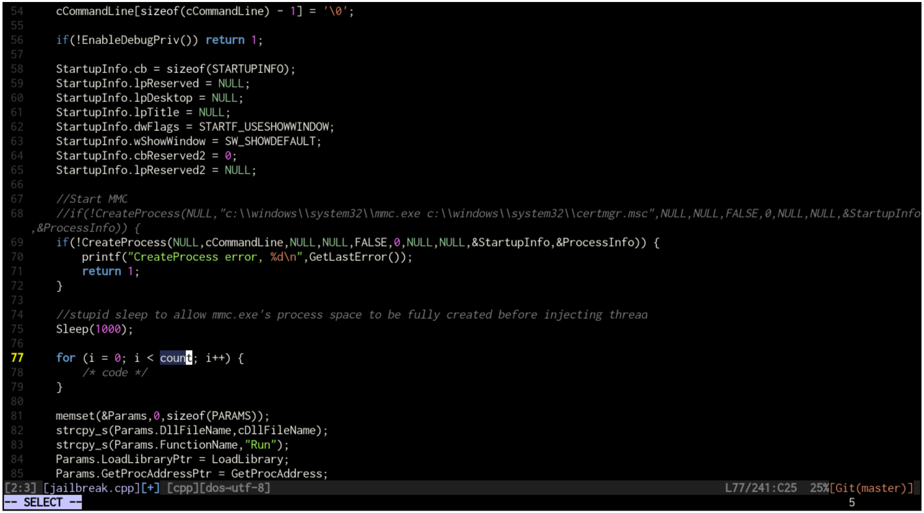
Complete a “for” loop



```
    cCommandLine[sizeof(cCommandLine) - 1] = '\0';

    if(!EnableDebugPriv()) return 1;

    StartupInfo.cb = sizeof(STARTUPINFO);
    StartupInfo.lpReserved = NULL;
    StartupInfo.lpDesktop = NULL;
    StartupInfo.lpTitle = NULL;
    StartupInfo.dwFlags = STARTF_USESHOWWINDOW;
    StartupInfo.wShowWindow = SW_SHOWDEFAULT;
    StartupInfo.cbReserved2 = 0;
    StartupInfo.lpReserved2 = NULL;

    //Start MMC
    //if(!CreateProcess(NULL,"c:\\windows\\system32\\mmc.exe c:\\windows\\system32\\certmgr.msc",NULL,NULL,FALSE,0,NULL,NULL,&StartupInfo
,&ProcessInfo)) {
    if(!CreateProcess(NULL,cCommandLine,NULL,NULL,FALSE,0,NULL,NULL,&StartupInfo,&ProcessInfo)) {
        printf("CreateProcess error, %d\n",GetLastError());
        return 1;
    }

    //stupid sleep to allow mmc.exe's process space to be fully created before injecting thread
    Sleep(1000);

    for

    memset(&Params,0,sizeof(PARAMS));
    strcpy_s(Params.DllFileName,cDllFileName);
    strcpy_s(Params.FunctionName,"Run");
    Params.LoadLibraryPtr = LoadLibrary;
    Params.GetProcAddressPtr = GetProcAddress;
    Params.ExitThreadPtr = ExitThread;

[2:3] [jailbreak.cpp][+] [cpp][dos-utf-8]                                          L77/239:C8    25%[Git(master)]
-- INSERT --
```

SnipMate

Snippet success


```
    cCommandLine[sizeof(cCommandLine) - 1] = '\0';

    if(!EnableDebugPriv()) return 1;

    StartupInfo.cb = sizeof(STARTUPINFO);
    StartupInfo.lpReserved = NULL;
    StartupInfo.lpDesktop = NULL;
    StartupInfo.lpTitle = NULL;
    StartupInfo.dwFlags = STARTF_USESHOWWINDOW;
    StartupInfo.wShowWindow = SW_SHOWDEFAULT;
    StartupInfo.cbReserved2 = 0;
    StartupInfo.lpReserved2 = NULL;

    //Start MMC
    //if(!CreateProcess(NULL,"c:\\windows\\system32\\mmc.exe c:\\windows\\system32\\certmgr.msc",NULL,NULL,FALSE,0,NULL,NULL,&StartupInfo
,&ProcessInfo)) {
    if(!CreateProcess(NULL,cCommandLine,NULL,NULL,FALSE,0,NULL,NULL,&StartupInfo,&ProcessInfo)) {
        printf("CreateProcess error, %d\n",GetLastError());
        return 1;
    }

    //stupid sleep to allow mmc.exe's process space to be fully created before injecting thread
    Sleep(1000);

    for (i = 0; i < count; i++) {
        /* code */
    }

    memset(&Params,0,sizeof(PARAMS));
    strcpy_s(Params.DllFileName,cDllFileName);
    strcpy_s(Params.FunctionName,"Run");
    Params.LoadLibraryPtr = LoadLibrary;
    Params.GetProcAddressPtr = GetProcAddress;
[2:3] [jailbreak.cpp][+] [cpp][dos-utf-8]                                L77/241:C25  25%[Git(master)]
-- SELECT --                                                                        5
```

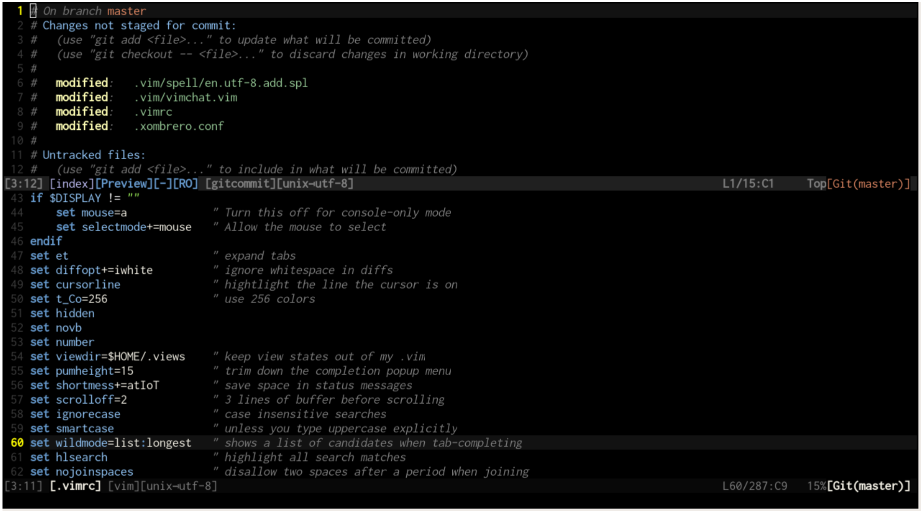
Fugitive

git in vim

- Do all your git magic from within vim
- Often way nicer than command line
- Navigate diffs
- Check in/check out/stage/commit
- Way too big to get into here
- Check out: `http://vimcasts.org/episodes`
- Short story: `:Gw`, `:Gr`, `:Gstatus`, `:Gcommit`, `:Gdiff`

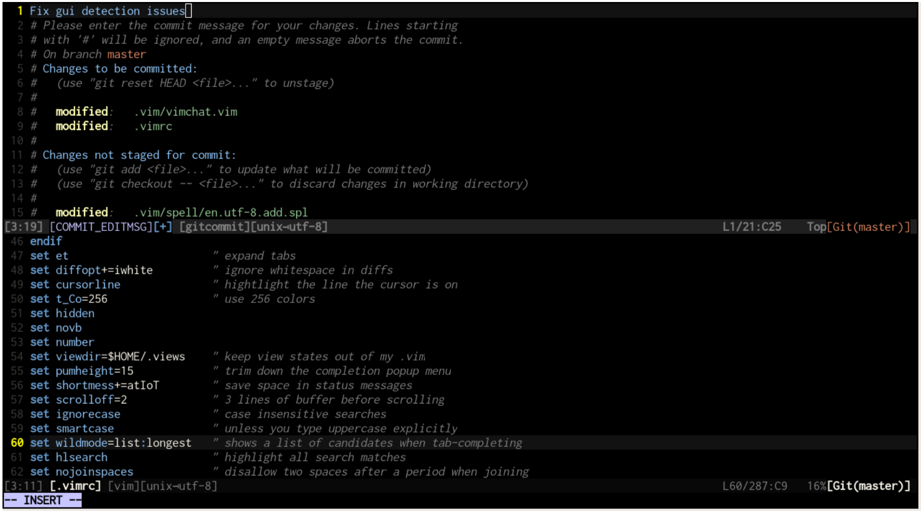
Fugitive

:Gstatus



```
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#   modified:   .vim/spell/en.utf-8.add.spl
#   modified:   .vim/vimchat.vim
#   modified:   .vimrc
#   modified:   .xombrero.conf
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
[3:12] [index][Preview][-][RO] [gitcommit][unix-utf-8]                    L1/15:C1     Top[Git(master)]
if $DISPLAY != ""
    set mouse=a               " Turn this off for console-only mode
    set selectmode+=mouse     " Allow the mouse to select
endif
set et                        " expand tabs
set diffopt+=iwhite           " ignore whitespace in diffs
set cursorline                " hightlight the line the cursor is on
set t_Co=256                  " use 256 colors
set hidden
set novb
set number
set viewdir=$HOME/.views      " keep view states out of my .vim
set pumheight=15              " trim down the completion popup menu
set shortmess+=atIoT          " save space in status messages
set scrolloff=2               " 3 lines of buffer before scrolling
set ignorecase                " case insensitive searches
set smartcase                 " unless you type uppercase explicitly
set wildmode=list:longest     " shows a list of candidates when tab-completing
set hlsearch                  " highlight all search matches
set nojoinspaces              " disallow two spaces after a period when joining
[3:11] [.vimrc] [vim][unix-utf-8]                                         L60/287:C9   15%[Git(master)]
```

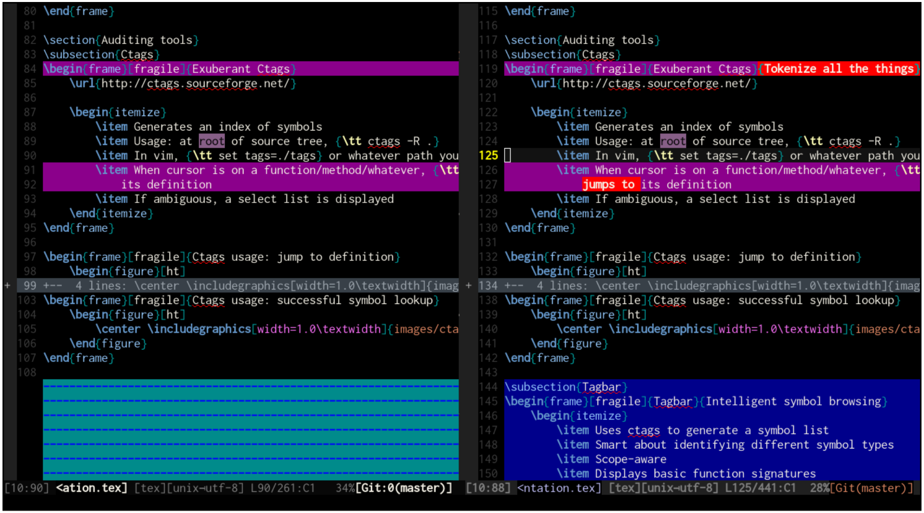
Fugitive
:Gcommit



```
Fix gui detection issues
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   .vim/vimchat.vim
#       modified:   .vimrc
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   .vim/spell/en.utf-8.add.spl
[3:19] [COMMIT_EDITMSG][+] [gitcommit][unix-utf-8]                                      L1/21:C25     Top[Git(master)]
endif
set et                          " expand tabs
set diffopt+=iwhite             " ignore whitespace in diffs
set cursorline                  " hightlight the line the cursor is on
set t_Co=256                    " use 256 colors
set hidden
set novb
set number
set viewdir=$HOME/.views        " keep view states out of my .vim
set pumheight=15                " trim down the completion popup menu
set shortmess+=atIoT            " save space in status messages
set scrolloff=2                 " 3 lines of buffer before scrolling
set ignorecase                  " case insensitive searches
set smartcase                   " unless you type uppercase explicitly
set wildmode=list:longest       " shows a list of candidates when tab-completing
set hlsearch                    " highlight all search matches
set nojoinspaces                " disallow two spaces after a period when joining
[3:11] [.vimrc] [vim][unix-utf-8]                                                       L60/287:C9    16%[Git(master)]
-- INSERT --
```

Fugitive

:Gdiff



```
\end{frame}

\section{Auditing tools}
\subsection{Ctags}
\begin{frame}[fragile]{Exuberant Ctags}
    \url{http://ctags.sourceforge.net/}

    \begin{itemize}
        \item Generates an index of symbols
        \item Usage: at root of source tree, {\tt ctags -R .}
        \item In vim, {\tt set tags=./tags} or whatever path you
        \item When cursor is on a function/method/whatever, {\tt
            its definition
        \item If ambiguous, a select list is displayed
    \end{itemize}
\end{frame}

\begin{frame}[fragile]{Ctags usage: jump to definition}
    \begin{figure}[ht]
+--  4 lines: \center \includegraphics[width=1.0\textwidth]{imag
\begin{frame}[fragile]{Ctags usage: successful symbol lookup}
    \begin{figure}[ht]
        \center \includegraphics[width=1.0\textwidth]{images/cta
    \end{figure}
\end{frame}

-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
[10:90] <ation.tex] [tex][unix-utf-8] L90/261:C1   34%[Git:0(master)]
```

```
\end{frame}

\section{Auditing tools}
\subsection{Ctags}
\begin{frame}[fragile]{Exuberant Ctags}{Tokenize all the things}
    \url{http://ctags.sourceforge.net/}

    \begin{itemize}
        \item Generates an index of symbols
        \item Usage: at root of source tree, {\tt ctags -R .}
        \item In vim, {\tt set tags=./tags} or whatever path you
        \item When cursor is on a function/method/whatever, {\tt
            jumps to its definition
        \item If ambiguous, a select list is displayed
    \end{itemize}
\end{frame}

\begin{frame}[fragile]{Ctags usage: jump to definition}
    \begin{figure}[ht]
+--  4 lines: \center \includegraphics[width=1.0\textwidth]{imag
\begin{frame}[fragile]{Ctags usage: successful symbol lookup}
    \begin{figure}[ht]
        \center \includegraphics[width=1.0\textwidth]{images/cta
    \end{figure}
\end{frame}

\subsection{Tagbar}
\begin{frame}[fragile]{Tagbar}{Intelligent symbol browsing}
    \begin{itemize}
        \item Uses ctags to generate a symbol list
        \item Smart about identifying different symbol types
        \item Scope-aware
        \item Displays basic function signatures
[10:88] <ntation.tex] [tex][unix-utf-8] L125/441:C1  28%[Git(master)]
```

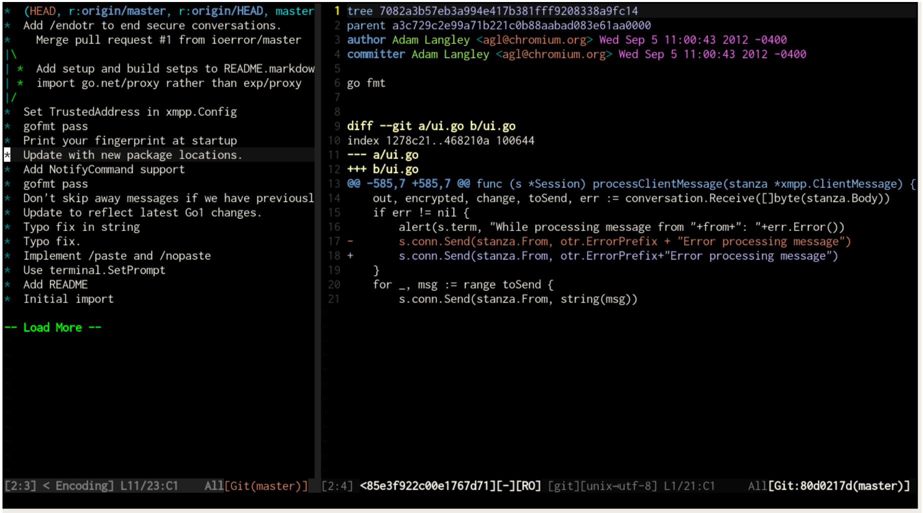
Gitv

Your non-GUI git GUI

- Plugin for Fugitive
- Commit browsing
- Perusing file history
- Jumping into vimdiff

Gitv

Browser mode

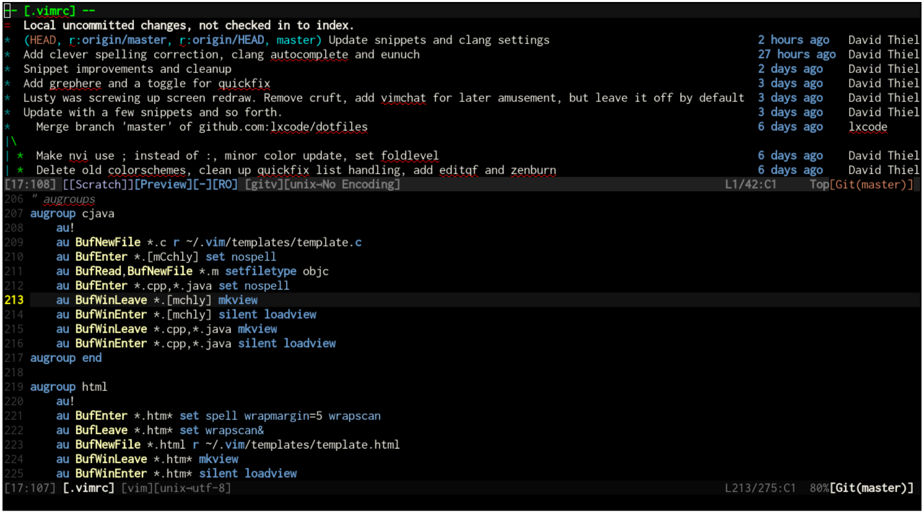


```
*  (HEAD, r:origin/master, r:origin/HEAD, master
*  Add /endotr to end secure conversations.
*    Merge pull request #1 from ioerror/master
|\
| *  Add setup and build setps to README.markdow
| *  import go.net/proxy rather than exp/proxy
|/
*  Set TrustedAddress in xmpp.Config
*  gofmt pass
*  Print your fingerprint at startup
*  Update with new package locations.
*  Add NotifyCommand support
*  gofmt pass
*  Don't skip away messages if we have previousl
*  Update to reflect latest Go1 changes.
*  Typo fix in string
*  Typo fix.
*  Implement /paste and /nopaste
*  Use terminal.SetPrompt
*  Add README
*  Initial import

-- Load More --

[2:3] < Encoding] L11/23:C1     All[Git(master)]
```

```
tree 7082a3b57eb3a994e417b381fff9208338a9fc14
parent a3c729c2e99a71b221c0b88aabad083e61aa0000
author Adam Langley <agl@chromium.org> Wed Sep 5 11:00:43 2012 -0400
committer Adam Langley <agl@chromium.org> Wed Sep 5 11:00:43 2012 -0400

go fmt


diff --git a/ui.go b/ui.go
index 1278c21..468210a 100644
--- a/ui.go
+++ b/ui.go
@@ -585,7 +585,7 @@ func (s *Session) processClientMessage(stanza *xmpp.ClientMessage) {
     out, encrypted, change, toSend, err := conversation.Receive([]byte(stanza.Body))
     if err != nil {
         alert(s.term, "While processing message from "+from+": "+err.Error())
-        s.conn.Send(stanza.From, otr.ErrorPrefix + "Error processing message")
+        s.conn.Send(stanza.From, otr.ErrorPrefix+"Error processing message")
     }
     for _, msg := range toSend {
         s.conn.Send(stanza.From, string(msg))

[2:4] <85e3f922c00e1767d71][-][RO] [git][unix-utf-8] L1/21:C1     All[Git:80d0217d(master)]
```

Gitv

File mode



```
-- [.vimrc] --
= Local uncommitted changes, not checked in to index.
* (HEAD, r:origin/master, r:origin/HEAD, master) Update snippets and clang settings                          2 hours ago   David Thiel
* Add clever spelling correction, clang autocomplete and eunuch                                              27 hours ago  David Thiel
* Snippet improvements and cleanup                                                                           2 days ago    David Thiel
* Add grephere and a toggle for quickfix                                                                     3 days ago    David Thiel
* Lusty was screwing up screen redraw. Remove cruft, add vimchat for later amusement, but leave it off by default 3 days ago David Thiel
* Update with a few snippets and so forth.                                                                   3 days ago    David Thiel
*   Merge branch 'master' of github.com:lxcode/dotfiles                                                      6 days ago    lxcode
|\
| * Make nvi use ; instead of :, minor color update, set foldlevel                                           6 days ago    David Thiel
| * Delete old colorschemes, clean up quickfix list handling, add editqf and zenburn                        6 days ago    David Thiel
[17:108] [[Scratch]][Preview][-][RO] [gitv][unix-No Encoding]                                 L1/42:C1      Top[Git(master)]
" augroups
augroup cjava
    au!
    au BufNewFile *.c r ~/.vim/templates/template.c
    au BufEnter *.[mCchly] set nospell
    au BufRead,BufNewFile *.m setfiletype objc
    au BufEnter *.cpp,*.java set nospell
    au BufWinLeave *.[mchly] mkview
    au BufWinEnter *.[mchly] silent loadview
    au BufWinLeave *.cpp,*.java mkview
    au BufWinEnter *.cpp,*.java silent loadview
augroup end

augroup html
    au!
    au BufEnter *.htm* set spell wrapmargin=5 wrapscan
    au BufLeave *.htm* set wrapscan&
    au BufNewFile *.html r ~/.vim/templates/template.html
    au BufWinLeave *.htm* mkview
    au BufWinEnter *.htm* silent loadview
[17:107] [.vimrc] [vim][unix-utf-8]                                                            L213/275:C1   80%[Git(master)]
```

Suggested Reading

- :help jump
- :help buffer
- :help window-move-cursor
- :help folds

Other things you may like

- statline: https://github.com/millermedeiros/vim-statline
- tag signatures: http://www.vim.org/scripts/script.php?script_id=2714
- yankring: http://www.vim.org/scripts/script.php?script_id=1234
    - Or yankstack: https://github.com/maxbrunsfeld/vim-yankstack
- cocoa.vim: https://github.com/msanders/cocoa.vim
- clang-complete: http://github.com/Rip-Rip/clang_complete
- gnupg.vim: https://github.com/jamessan/vim-gnupg
- GrepHere: https://github.com/vim-scripts/GrepHere
- editqf: https://github.com/jceb/vim-editqf
- Check out my setup at https://github.com/lxcode/dotfiles

Questions?

vimrc

```
" Keymappings {{{
" Make space clear highlighted searches
nmap <silent> <space> :noh<CR>
"left/right arrows to switch buffers in normal mode
map <right> :bn<cr>
map <left> :bp<cr>
map <home> :rewind<cr>
map <end> :last<cr>
map g<Tab> :bn<CR>
nnoremap <C-Tab> gt
" Make Y behave like C and D
nnoremap Y y$
" Use , in addition to \ for the leader
let mapleader = ","
nmap \ ,
nmap <space> ,
" save my pinky
nore ; :
" auto-format the current paragraph
nnoremap __ gwip
nnoremap -- :call WrapMerge()<CR>
" Get rid of jumping behavior when using these search functions
nnoremap * *<c-o>
nnoremap # #<c-o>
" Clear search pattern with \\
map <silent> <Leader>\ :noh<CR>
" correct spelling
nmap <F1> [s1z=<C-o>
imap <F1> <Esc>[s1z=<C-o>a
map <F8> :w<CR> :!make<CR>
map <silent> <F9> :call ToggleVExplorer()<CR>
nnoremap <silent> <F10> :TagbarToggle<CR>
set pastetoggle=<F11>
" jump to next quickfix item
map <F12> :cn<CR>
" preview the tag under the cursor
nmap <C-p> :exe "ptag" expand("<cword>")<CR>
nnoremap <silent> <C-c> :call QuickfixToggle()<cr>
" Window movement
nnoremap <C-j> <C-W>w
nnoremap <C-k> <C-W>W
" Keep selected blocks selected when shifting
vmap > >gv
vmap < <gv
nmap <Leader>x :call system("cd `dirname %` && urxvt")<CR>
" Change to the directory of the current file
nmap cd :lcd %:h \| :pwd<CR>
```

```
" Delete a vuln
" This works when I type it, but not here...
nmap dav ?%<CR>2d/%---\|\\vtitle<CR>
nmap <Leader>fw :StripWhitespace<CR>
" Quick exits
nmap zz ZZ
" }}}

" Settings {{{
syntax on
filetype plugin on
filetype indent on
helptags ~/.vim/doc

if has('gui')
    set gcr=n:blinkon0          " don't blink the cursor in normal mode
    set guioptions=aAegiM       " get rid of useless stuff in the gui
    if has("gui_macvim")
        set guifont=Inconsolata:h18
        set clipboard=unnamed
        noremap <Leader>zo :set guifont=Inconsolata:h4<CR>
        noremap <Leader>zi :set guifont=Inconsolata:h18<CR>
    else
        set guifont=Inconsolata\ 14
    endif
endif
if has('gui_running')
    set ballooneval
    set balloondelay=100
endif
if $DISPLAY != ""
    "set cursorline          " I like this, but damn is it slow
    set mouse=a             " Turn this off for console-only mode
    set selectmode+=mouse    " Allow the mouse to select
    set ttymouse=xterm2
endif
set et                      " expand tabs
set diffopt+=iwhite,vertical,filler   " ignore whitespace in diffs
set hidden                  " allow hidden buffers
set novb t_vb=              " no visual bell
set nonu                    " line numbers
set viewdir=$HOME/.views    " keep view states out of my .vim
set pumheight=15            " trim down the completion popup menu
set shortmess+=atIoT        " save space in status messages
set scrolloff=3             " 3 lines of buffer before scrolling
set ignorecase              " case insensitive searches
set smartcase               " unless you type uppercase explicitly
set smarttab                " use shiftwidth instead of tab stops
set wildmode=longest,list   " shows a list of candidates when tab-completing
set wildmenu                " use a more functional completion menu when tab-completing
set encoding=utf-8          " always use utf-8
set hlsearch                " highlight all search matches
set foldcolumn=0            " I never use this.
set nojoinspaces            " disallow two spaces after a period when joining
if version >= 704
    set formatoptions=qjnrtlmnc " auto-formatting style
else
    set formatoptions=qnrtlmnc  " auto-formatting style minus j
endif
set autoindent
set shiftround              " Round to the nearest shiftwidth when shifting
set linebreak               " When soft-wrapping long lines, break at a word
set comments-=s1:/*,mb:*,ex:*/
set comments+=fb:*,b:\\item
set formatlistpat=^\\s*\\([0-9]\\+\\\|[a-z]\\)[\\].:)}]\\s\\+
if has("macunix")
    set grepprg=grep\ -R\ --exclude=\"*.aux\"\ --exclude=\"tags\"\ --exclude=\"*scope.out\"\ --color=
else
    set grepprg=bsdgrep\ -R\ --exclude=\"*.aux\"\ --exclude=\"tags\"\ --exclude=\"*scope.out\"\ --col
endif
set cpoptions=BFt
set completeopt=menuone,longest
set tags=tags;/                 " use first tags file in a directory tree
set nobackup                    " ugh, stop making useless crap
set nowritebackup               " same with overwriting
set directory=/tmp              " litter up /tmp, not the CWD
set nomodeline                  " modelines are dumb
set tabstop=4 shiftwidth=4
set backspace=indent,eol,start
set ruler                       " show position in file
set title
set titlestring=%t%(\ %M%)%(\ (%{expand(\"%:p:h\")})%)%(\ %a%)
set ttimeout
set ttimeoutlen=100             " Make it so Esc enters Normal mode right away
set helpheight=0                " no minimum helpheight
set incsearch                   " search incrementally
set showmatch                   " show the matching terminating bracket
set suffixes=.out               " set priority for tab completion
set wildignore+=*.bak,~*,*.o,*.aux,*.dvi,*.bbl,*.blg,*.orig,*.toc,*.fls
set wildignore+=*.loc,*.gz,*.tv,*.ilg,*.lltr,*.lov,*.lstr,*.idx,*.pdf
set wildignore+=*.fdb_latexmk,*.ind,*.cg,*.tdo,*.log,*.latexmain,*.out
set sidescroll=1                " soft wrap long lines
set lazyredraw ttyfast          " go fast
set errorfile=/tmp/errors.vim
set cscopequickfix=s-,c-,d-,i-,t-,e-        " omfg so much nicer
set foldlevelstart=2            " the default level of fold nesting on startup
set cryptmethod=blowfish    " in case I ever decide to use vim -x
set autoread                " Disable warning about file change to writable
set conceallevel=0          " Don't hide things by default
"set updatecount=100 updatetime=3600000        " saves power on notebooks

"if exists('&autochdir')
"    " Change directory to first open file
"    set autochdir
"    set noautochdir
"endif

" colors
set t_Co=256                " use 256 colors
colorscheme lx-256-dark
" }}}

" Plugins {{{
" 33ms startup penalty!
source ~/.vim/ftplugin/man.vim

" netrw {{{
let g:netrw_liststyle=3
let g:netrw_browse_split=4
let g:netrw_winsize=25
let g:netrw_banner=0
let g:netrw_list_hide='\(^\|\s\s\)\zs\.\S\+' "hide files by default
let g:netrw_sort_sequence = '[\/]$,*,\%(' . join(map(split(&suffixes, ','), 'escape(v:val, ".*$~")'),
" }}}

" quickfixsigns {{{
let g:quickfixsigns_classes=['qfl', 'loc', 'marks', 'vcsdiff', 'breakpoints']
" Disable display of the ' and . marks, so the gutter will be disabled until
" manually set marks or quickfix/diff info is present.
let g:quickfixsigns#marks#buffer = split('abcdefghijklmnopqrstuvwxyz', '\zs')
let g:quickfixsign_use_dummy = 0
let g:quickfixsigns#vcsdiff#highlight = {'DEL': 'QuickFixSignsDiffDeleteLx', 'ADD': 'QuickFixSignsDif
" }}}

" buftabs {{{
let g:buftabs_only_basename=1
" }}}

" clever-f {{{
let g:clever_f_mark_char_color="PreProc"
let g:clever_f_smart_case=1
" }}}

" Indentlines {{{
```

```
nmap \|\| :IndentLinesToggle<CR>
let g:indentLine_faster = 1
let g:indentLine_enabled = 0
" }}}

" Limelight {{{
let g:limelight_conceal_ctermfg = 240
let g:limelight_conceal_guifg = '#777777'
let g:limelight_default_coefficient = 0.7
" }}}

" latex-box {{{
let g:tex_flavor="latex"
let g:tex_no_error = 1
let g:tex_conceal= ""
let g:tex_comment_nospell = 1
"let g:LatexBox_latexmk_options = "--disable-write18 --file-line-error --interaction=batchmode -pdfla
let g:LatexBox_latexmk_options = "-xelatex --disable-write18 --file-line-error --interaction=batchmod
" Work around the fact that cmdline macvim doesn't support server mode
if has("gui_macvim")
    let g:LatexBox_latexmk_async = 1
else
    if has("macunix")
        let g:LatexBox_latexmk_async = 1
    else
        let g:LatexBox_latexmk_async = 0
    endif
endif
if has("macunix")
    let g:LatexBox_viewer = "open"
else
    let g:LatexBox_viewer = "evince"
endif
let g:LatexBox_split_side = "rightbelow"
let g:LatexBox_quickfix = 0
let g:LatexBox_show_warnings = 0
let g:LatexBox_ignore_warnings = [
            \ 'Underfull',
            \ 'Overfull',
            \ 'specifier changed to',
            \ 'Font shape',
            \ 'epstopdf',
            \ ]

let g:LatexBox_fold_parts=[
           \ "part",
           \ "chapter",
           \ "section",
          \ "subsection",
          \ "subsubsection",
          \ "vtitle"
          \ ]

augroup latex
    " The NoStarch style is a bit crufty and needs pdflatex
    au BufWinEnter book.tex let g:LatexBox_latexmk_options = "-interaction=batchmode -draftmode"
    au BufWinEnter book.tex let g:LatexBox_fold_envs = 1
    if &diff
        let g:LatexBox_Folding = 0
        let g:LatexBox_fold_preamble = 0
        let g:LatexBox_fold_envs = 0
    else
        let g:LatexBox_Folding = 1
        let g:LatexBox_fold_preamble = 1
        let g:LatexBox_fold_envs = 1
    endif
"    au BufWritePost *.tex Latexmk
    au BufWinLeave *.tex,*.sty mkview
    au BufWinEnter *.tex,*.sty silent loadview
    au FileType tex syntax spell toplevel
    au FileType tex set spell textwidth=78 smartindent
    au FileType tex set formatoptions+=w foldlevelstart=6
    au FileType tex imap <buffer> [[ \begin{
    au FileType tex imap <buffer> ]] <Plug>LatexCloseCurEnv
    au FileType tex imap <S-Enter> \pagebreak
    au FileType tex nmap tt i{\tt <Esc>wEa}<Esc>
    au FileType tex source ~/.vim/ftplugin/quotes.vim
augroup end
" }}}

" supertab {{{
let g:SuperTabContextFileTypeExclusions = ['make']
let g:SuperTabDefaultCompletionType = "context"
let g:SuperTabCompletionContexts = ['s:ContextText', 's:ContextDiscover']
let g:SuperTabContextTextOmniPrecedence = ['&omnifunc', '&completefunc']
let g:SuperTabContextDiscoverDiscovery =
    \ ["&completefunc:<c-x><c-u>", "&omnifunc:<c-x><c-o>"]

autocmd FileType *
            \  if &omnifunc != '' |
            \      let g:myfunc = &omnifunc |
            \  elseif &completefunc != '' |
            \      let g:myfunc = &completefunc |
            \  else |
            \      let g:myfunc = '' |
            \  endif |
            \  if g:myfunc != '' |
            \      call SuperTabChain(g:myfunc, "<c-p>") |
            \      call SuperTabSetDefaultCompletionType("<c-x><c-u>") |
            \  endif
" }}}

" cctree {{{
if has("macunix")
    let g:CCTreeSplitProgCmd="/opt/local/bin/gsplit"
else
    let g:CCTreeSplitProgCmd="/usr/local/bin/gsplit"
endif
" }}}

" rainbow {{{
map <Leader>r :RainbowToggle<CR>
" }}}

" vimchat {{{
let g:vimchat_otr = 1
let g:vimchat_statusicon = 0
let g:vimchat_showPresenceNotification = -1
let g:vimchat_pync_enabled = 1
"map g<Tab> gt
```

```
" }}}

" CtrlP {{{
let g:ctrlp_cmd = 'CtrlPMixed'
let g:ctrlp_map = '<C-e>'
let g:ctrlp_by_filename = 1
let g:ctrlp_working_path_mode = 0
let g:ctrlp_max_height = 30
let g:ctrlp_clear_cache_on_exit = 0
let g:ctrlp_extensions = ['buffertag']
map <Leader>e :CtrlP<CR>
map <Leader>m :CtrlPMRU<CR>
map <Leader>t :CtrlPTag<CR>
map <Leader>g :CtrlPBufTagAll<CR>
map <Leader>b :CtrlPBuffer<CR>
" CtrlP tjump
nnoremap <c-]> :CtrlPtjump<cr>
vnoremap <c-]> :CtrlPtjumpVisual<cr>
let g:ctrlp_tjump_shortener = ['/\(Users|home\)/lx', '~']
let g:ctrlp_tjump_only_silent = 1
" }}}

" statline {{{
let g:statline_fugitive=1
let g:statline_trailing_space=0
let g:statline_mixed_indent=0
let g:statline_filename_relative=1
" }}}

" clang {{{
let g:clang_complete_enable = 1
let g:clang_library_path='/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolch
let g:clang_user_options='-fblocks -isysroot /Applications/Xcode.app/Contents/Developer/Platforms/iPh
let g:clang_complete_copen = 1
let g:clang_snippets = 1
let g:clang_use_library = 1
" }}}

" tagbar {{{
let g:tagbar_iconchars = ['▸', '▾']
let g:tagbar_type_objc = {
    \ 'ctagstype' : 'ObjectiveC',
    \ 'kinds'     : [
        \ 'i:interface',
        \ 'I:implementation',
        \ 'p:Protocol',
        \ 'm:Object_method',
        \ 'c:Class_method',
        \ 'v:Global_variable',
        \ 'F:Object field',
        \ 'f:function',
        \ 'p:property',
        \ 't:type_alias',
        \ 's:type_structure',
        \ 'e:enumeration',
        \ 'M:preprocessor_macro',
    \ ],
    \ 'sro'        : ' ',
    \ 'kind2scope' : {
        \ 'i' : 'interface',
        \ 'I' : 'implementation',
        \ 'p' : 'Protocol',
        \ 's' : 'type_structure',
        \ 'e' : 'enumeration'
    \ },
    \ 'scope2kind' : {
        \ 'interface'      : 'i',
        \ 'implementation' : 'I',
        \ 'Protocol'       : 'p',
        \ 'type_structure' : 's',
        \ 'enumeration'    : 'e'
    \ }
```

```
\ }

let g:tagbar_type_tex = {
    \ 'ctagstype' : 'latex',
    \ 'kinds'     : [
        \ 's:sections',
        \ 'g:graphics',
        \ 'l:labels',
        \ 'r:refs:1',
        \ 'p:pagerefs:1',
        \ 'v:vulns',
        \ 'r:strecs',
        \ 'R:ltrecs'
    \ ],
    \ 'sort'    : 0,
\ }

let g:tagbar_type_markdown = {
        \ 'ctagstype' : 'markdown',
        \ 'kinds' : [
                \ 'h:Heading_L1',
                \ 'i:Heading_L2',
                \ 'k:Heading_L3'
        \ ]
\ }

let g:tagbar_type_scala = {
    \ 'ctagstype' : 'Scala',
    \ 'kinds'     : [
        \ 'p:packages:1',
        \ 'V:values',
        \ 'v:variables',
        \ 'T:types',
        \ 't:traits',
        \ 'o:objects',
        \ 'a:aclasses',
        \ 'c:classes',
        \ 'r:cclasses',
        \ 'm:methods'
    \ ]
\ }
" }}}

" }}}

" augroups {{{
augroup cjava
    au!
    au BufNewFile *.c r ~/.vim/templates/template.c
    au BufWinEnter *.[mCchly] set nospell number comments+=s1:/*,mb:*,ex:*/
    au BufWinEnter,BufNewFile *.m,*.xm,*.xmi setfiletype objc
    au BufWinEnter,BufNewFile *.m,*.xm,*.xmi let c_no_curly_error = 1
    au BufWinEnter *.cpp,*.java set nospell number
    au BufWinLeave *.[mchly] mkview
    au BufWinEnter *.[mchly] silent loadview
    au BufWinLeave *.cpp,*.java mkview
    au BufWinEnter *.cpp,*.java silent loadview
augroup end

augroup html
    au!
    au FileType html set spell wrapmargin=5 wrapscan number
    au FileType html set wrapscan&
    au BufNewFile *.html r ~/.vim/templates/template.html
    au BufWinLeave *.htm* mkview
    au BufWinEnter *.htm* silent loadview
augroup end

augroup python
    au FileType python set smartindent smarttab nospell number
    au BufWinLeave *.py mkview
    au BufWinEnter *.py silent loadview
```

```
augroup end

augroup markdown
    au BufWinEnter *.notes set filetype=markdown
    au BufWinLeave *.md,*.notes, mkview
    au BufWinEnter *.md,*.notes, silent loadview
    au BufWinEnter *.md,*.notes, imap <C-l> <C-t>
    au BufWinEnter *.md,*.notes, imap <C-h> <C-d>
    au BufWinEnter *.md,*.notes,*mutt*, imap >> <C-t>
    au BufWinEnter *.md,*.notes,*mutt*, imap << <C-d>
    au FileType markdown set spell
    au FileType markdown set textwidth=78 complete+=k comments+=b:-,b:+,b:*,b:+,n:>
augroup end

" Disable spellcheck on quickfix, switch between quickfix lists with the arrow
" keys
augroup quickfix
    au FileType qf, noremap ' <CR><C-W><C-P>j
    au FileType qf, set nospell number
    au FileType qf, nnoremap <silent> <buffer> <right> :cnew<CR>
    au FileType qf, nnoremap <silent> <buffer> <left> :col<CR>
    au FileType qf, setlocal statusline=\ %n\ \ %f%=L%l/%L\ %P
    au BufReadPost quickfix call GrepColors()
    au BufWinEnter quickfix call GrepColors()
    au BufWinEnter qf:list call GrepColors()
augroup end

augroup msdocs
    au BufReadCmd *.docx,*.xlsx,*.pptx call zip#Browse(expand("<amatch>"))
    au BufReadCmd *.odt,*.ott,*.ods,*.ots,*.odp,*.otp,*.odg,*.otg call zip#Browse(expand("<amatch>"))
augroup end

augroup misc
    au FileType netrw unmap <buffer> --
    au BufWinEnter *.applescript set filetype=applescript
    au BufWinEnter *.nmap, set syntax=nmap
    au BufWinEnter *.scala, set filetype=scala
    au BufWinEnter *.dtrace, set filetype=D
    au BufWinEnter *.less, set filetype=css
    au BufWinEnter *.fugitiveblame,*.diff, set nospell number
    au BufWinLeave *.txt,*.conf,.vimrc,*.notes mkview
    au BufWinEnter *.txt,*.conf,.vimrc,*.notes silent loadview
    au BufWinEnter .vimrc set foldmethod=marker
    au FileType make set diffopt-=iwhite
    au FileType vim set nospell
    au FileType mail set spell complete+=k nonu
    " par is much better at rewrapping mail
    au FileType mail if executable("par") | set formatprg=par | endif
    au FileType mail map <F8> :%g/^> >/d<CR>gg10j
    au FileType mail StripWhitespace
    au FileType mail,text let b:delimitMate_autoclose = 0
    au BufWinEnter *vimChatRoster, set foldlevel=1
    au BufWinEnter *.nse set filetype=lua
    " If a JS file has only one line, unminify it
    au FileType javascript if line('$')==1 | call Unminify() | endif
    au FileType help set nospell
    " What - like how does this even work
    au InsertLeave * hi! link CursorLine CursorLine
    au InsertEnter * hi! link CursorLine Normal
    " Disable the 'warning, editing a read-only file' thing that
    " hangs the UI
    au FileChangedRO * se noreadonly
augroup end

augroup syntax
    autocmd FileType css setlocal omnifunc=csscomplete#CompleteCSS
    autocmd FileType html setlocal omnifunc=htmlcomplete#CompleteTags
    autocmd FileType javascript setlocal omnifunc=javascriptcomplete#CompleteJS
"    autocmd FileType python setlocal omnifunc=pythoncomplete#Complete
    autocmd FileType xml setlocal omnifunc=xmlcomplete#CompleteTags
    autocmd FileType ruby setlocal omnifunc=rubycomplete#Complete
augroup end
```

```
" }}}

" Custom functions {{{
" Quickfix toggle
let g:quickfix_is_open = 0

function! QuickfixToggle()
    if g:quickfix_is_open
        cclose
        let g:quickfix_is_open = 0
        execute g:quickfix_return_to_window . "wincmd w"
    else
        let g:quickfix_return_to_window = winnr()
        bot copen
        let g:quickfix_is_open = 1
    endif
endfunction

" Toggle Vexplore
function! ToggleVExplorer()
  if exists("t:expl_buf_num")
      let expl_win_num = bufwinnr(t:expl_buf_num)
      if expl_win_num != -1
          let cur_win_nr = winnr()
```

```
            exec expl_win_num . 'wincmd w'
            close
            exec cur_win_nr . 'wincmd w'
            unlet t:expl_buf_num
        else
            unlet t:expl_buf_num
        endif
    else
        exec '1wincmd w'
        Vexplore
        let t:expl_buf_num = bufnr("%")
    endif
endfunction

" wrap nicely
function! WrapMerge()
    set formatoptions-=w
    exec "normal gwip"
    set formatoptions+=w
endfunction

" clear quickfix
command -bar Qfc call setqflist([])
```

```
" Read in cookiefiles
command -bar Cookies call ReadCookies()
function ReadCookies()
    call system("cp Cookies.binarycookies /tmp/")
    %!python $HOME/bin/BinaryCookieReader.py /tmp/Cookies.binarycookies
endfunction

" ex command for toggling hex mode - define mapping if desired
command -bar Hexmode call ToggleHex()

" helper function to toggle hex mode
function ToggleHex()
  " hex mode should be considered a read-only operation
  " save values for modified and read-only for restoration later,
  " and clear the read-only flag for now
  let l:modified=&mod
  let l:oldreadonly=&readonly
  let &readonly=0
  let l:oldmodifiable=&modifiable
  let &modifiable=1
  if !exists("b:editHex") || !b:editHex
    " save old options
    let b:oldft=&ft
    let b:oldbin=&bin
    " set new options
    setlocal binary " make sure it overrides any textwidth, etc.
    let &ft="xxd"
    " set status
    let b:editHex=1
    " switch to hex editor
    %!xxd
  else
    " restore old options
    let &ft=b:oldft
    if !b:oldbin
      setlocal nobinary
    endif
    " set status
    let b:editHex=0
    " return to normal editing
    %!xxd -r
  endif
  " restore values for modified and read only state
  let &mod=l:modified
  let &readonly=l:oldreadonly
  let &modifiable=l:oldmodifiable
endfunction
```

```
" I use this to highlight the match from grep, but keep quickfix syntax
" highlighting intact. This is for BSD grep.
command -bar GrepColors call GrepColors()
function GrepColors()
    set conceallevel=3
    set cocu=nv
    syn region ansiRed start="\e\[01;31m\e\[K"me=e-2 end="\e\[m"me=e-3 contains=ansiConceal
    syn match ansiConceal contained conceal    "\e\[\(\d*;\)*\d*m\e\[K"
    hi ansiRed    ctermfg=197   guifg=#FF005F  cterm=none         gui=none
    syn match ansiStop        conceal "\e\[m\e\[K"
    hi! link ansiStop NONE
endfunction

" Simple re-format for minified Javascript
command! Unminify call Unminify()
function! Unminify()
    %s/{\ze[^\r\n]/{\r/g
    %s/){/) {/g
    %s/};\?\ze[^\r\n]/\0\r/g
    %s/;\ze[^\r\n]/;\r/g
    %s/[^\s]\zs[=&|]\+\ze[^\s]/ \0 /g
    normal ggVG=
endfunction
command! -nargs=1 Graudit call Graudit(<f-args>)
function! Graudit(db)
    call system("$HOME/Tools/graudit/graudit -x 'cscope.*' -c0 -d " . a:db . " . | awk 'length($0) <
    copen
    cf /tmp/graudit.out
endfunction
" }}}
```