

# Expert Code Grokking with Vim

Vim hacks for penetration testers

David Thiel

November 21, 2017

# Outline

- 1 Must-haves
  - The Basics
  - Exploring
- 2 Auditing tools
  - Ctags
  - Tagbar
  - Cscope
  - Completion
  - Snippets
- 3 Source control
  - Fugitive
  - gv

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
- 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` directory

# The Basics

---

## basics

---

<b>syntax on</b>	" enable syntax highlighting
<b>filetype</b> plugin <b>on</b>	" filetype detection
<b>filetype</b> indent <b>on</b>	
<b>set number</b>	" line numbers
<b>set hidden</b>	" allow invisible buffers
<b>set ignorecase</b>	" case-insensitive searching
<b>set smartcase</b>	" but be smart about it
<b>set hlsearch</b>	" highlight all search matches
<b>set incsearch</b>	" search incrementally
<b>set et</b>	" expand tabs to spaces
<b>set tags=./tags</b>	" ctags - we'll get into that

---

# 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<sup>12</sup>

---

<sup>1</sup><http://vimdoc.sourceforge.net/html/doc/visual.html#visual-operators>

<sup>2</sup><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 `I` 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`
- Smoother session integration can be had with <https://github.com/tpope/vim-obsession>

# 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
- Can use anything that specifies line info; I use `graudit` with some custom regexes



# Quickfix

## Using grep/vimgrep

```
32 from SSLSocket import SSLSocket
33
34
35 class HTTPSConnection(HTTPConnection):
36     """
37     This class mirrors httplib.HTTPSConnection but uses ctSSL instead of the
38     standard ssl module.
39     For now the way to access low level SSL functions associated with a given
40     HTTPSConnection is to directly access the ssl and ssl_ctx attributes of the
41     object. TODO: change that.
42
43     @type ssl_ctx: ctSSL.SSL_CTX
44     @ivar ssl_ctx: SSL_CTX object for the HTTPS connection.
45
46     @type ssl: ctSSL.SSL
47     @ivar ssl: SSL object for the HTTPS connection.
48     certificates.
49     """
50
51     default_port = HTTPS_PORT
52
53     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

```

123     a command line argument. It has to be defined in each plugin class.
124     """
125     return
126
127
128     # Utility SSL/socket methods that turned out to be used by all the plugins
129     @classmethod
130     def _create_ssl_connection(self_class, target, ssl=None, ssl_ctx=None):
131         """
132         Read the shared_settings object shared between all the plugins and load
133         the proper settings the SSL_CTX and SSL objects.
134
135         @type ssl: ctSSL.SSL
136         @param ssl: SSL object for the SSL connection. If not specified,
137         a default SSL object will be created for the connection and SSL
138         certificates will NOT be verified when connecting to the server.
139
140         @type ssl_ctx: ctSSL.SSL_CTX
141         @param ssl_ctx: SSL_CTX object for the SSL connection. If not
142         specified, a default SSL_CTX object will be created for the connection
143         and SSL certificates will NOT be verified when connecting to the server.
144         """
[4:4] [PluginBase.py] [python][unix=utf-8] L133/198:C9 69%[Git(master)]
145     plugins/PluginBase.py|141| @param ssl_ctx: SSL_CTX object for the SSL connection. If not
146     plugins/PluginBase.py|142| specified, a default SSL_CTX object will be created for the connection
147     ./plugins/PluginCertInfo.py|33| X509_V_CODES, SSL_CTX
148     ./plugins/PluginCertInfo.py|343| ssl_ctx = SSL_CTX.SSL_CTX('tlsv1') # sslv23 hello will fail for specific servers such as post.craigslist
149     .org
150     ./plugins/PluginOpenSSLCipherSuites.py|29| from utils.ctSSL import SSL, SSL_CTX, constants, ctSSL_initialize, \
151     plugins/PluginOpenSSLCipherSuites.py|83| ctx = SSL_CTX.SSL_CTX(ssl_version)
152     ./plugins/PluginOpenSSLCipherSuites.py|202| ssl_ctx = SSL_CTX.SSL_CTX(ssl_version)
153     ./plugins/PluginOpenSSLCipherSuites.py|239| ssl_ctx = SSL_CTX.SSL_CTX(ssl_version)
154     ./plugins/PluginSessionRenegotiation.py|28| from utils.ctSSL import ctSSL_initialize, ctSSL_cleanup, SSL_CTX, \
[Quickfix List] :grep -nH -r SSL_CTX .

```

# File Browsing

netrw

- Use splits and the `:Explore` family
- Usage:
  - `:Explore` — open a file browser in your current window (if unmodified — otherwise, split first)
  - `:Lexplore` — toggle a file drawer on the left
  - 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
- You may want <https://github.com/tpope/vim-vinegar>

# Plugin management

## vim-plug

- There are several plugin managers
- The correct one is vim-plug:
  - <https://github.com/junegunn/vim-plug>

```
vim-plug install  
curl -fLo ~/.vim/autoload/plug.vim --create-dirs \\  
https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim
```

```
vim-plug vimrc  
call plug#begin('~/.vim/plugged')  
Plug 'Raimondi/delimitMate'  
Plug 'SirVer/ultisnips' | Plug 'honza/vim-snippets'  
call plug#end()
```

# Quick file opening

## FZF

- 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 FZF:

<https://github.com/junegunn/fzf> <https://github.com/junegunn/fzf.vim>

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

# Quick file opening

## ZFZ in action

```

41 // Generate data for all pids in the vector.
42 // If there are comparison constraints this could apply the operator
43 // before generating the process structure.
44 for (const auto& pid : pids) {
45     *procs = procstat_getprocs(*pstat, KERN_PROC_PID, std::stoi(pid), &cnt);
46     if (*procs == nullptr) {
47         TLOG << "Problem retrieving processes";
48         procstat_close(*pstat);
49         *pstat = nullptr;
50         return 0;
51     }
52 }
53 } else {
54     // Get all PIDs.
55     *procs = procstat_getprocs(*pstat, KERN_PROC_PROC, 0, &cnt);
56     if (*procs == nullptr) {
57         TLOG << "Problem retrieving processes";
58         procstat_close(*pstat);
59         *pstat = nullptr;
60         return 0;
61     }
62 }

```

[2/2] [osquery/tables/system/freebsd/procstat.cpp] [cpp] L1/77:C1 Top[Git(master)]

```

..query/tables/CMakeFiles/osquery_events_tables.dir/build.m...
..query/tables/CMakeFiles/osquery_events_tables.dir/progress...
..query/tables/CMakeFiles/osquery_events_tables.dir/depend...
..query/tables/CMakeFiles/osquery_events_tables.dir/flags.m...
..query/tables/CMakeFiles/osquery_events_tables.dir/build.m...
osquery/tables/networking/freebsd/process_open_sockets.cpp
osquery/tables/system/freebsd/process_open_files.cpp
osquery/tables/system/freebsd/pkg_packages.cpp
osquery/tables/system/freebsd/syctl_utils.cpp
osquery/tables/system/freebsd/os_version.cpp
osquery/tables/system/freebsd/fbsd_kwods.cpp
osquery/tables/networking/freebsd/routes.cpp
osquery/tables/system/freebsd/processes.cpp
osquery/tables/system/freebsd/procstat.cpp
docs/wiki/installation/install-freebsd.md
osquery/tables/system/freebsd/mounts.cpp
osquery/tables/system/freebsd/procstat.h
osquery/tables/system/freebsd/groups.cpp
osquery/tables/system/freebsd/users.cpp
osquery/events/freebsd/kernel_util.cpp
osquery/events/freebsd/fs_events.h
specs/freebsd/pkg_packages.table
specs/freebsd/fbsd_kwods.table
tools/provision/freebsd.sh
89/5110
~/g/osquery/freebsd

```

```

/*
 * Copyright (c) 2014-present, Facebook, Inc.
 * All rights reserved.
 *
 * This source code is licensed under the BSD-style license
 * LICENSE file in the root directory of this source tree,
 * or of patent rights can be found in the PATENTS file in th
 *
 */

#include <string>

#include <stdlib.h>
#include <unistd.h>
#include <limits.h>
#include <paths.h>

/// Required defines for libprocstat.
#include <sys/socket.h>
#include <sys/sysctl.h>
#include <sys/param.h>
#include <sys/queue.h>
#include <sys/user.h>

```

# 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)`
- You can also use FZF for this:

---

```
_____ FZF tag jump _____  
nmap <C-]> :call fzf#vim#tags(expand('<cword>'), \  
    {'options': '--exact --select-1 --exit-0'})<CR>
```

---

# Ctags usage: jump to definition

```

141 @param ssl_ctx: SSL_CTX object for the SSL connection. If not
142 specified, a default SSL_CTX object will be created for the connection
143 and SSL certificates will NOT be verified when connecting to the server.
144 """
145 shared_settings = self.class._shared_settings
146 timeout = shared_settings['timeout']
147 (host, ip_addr, port) = target
148
149 if shared_settings['starttls'] == 'smtp':
150     ssl_connection = STARTTLS.SMTPConnection(ip_addr, port, ssl, ssl_ctx,
151                                             timeout=timeout)
152 elif shared_settings['starttls'] == 'xmpp':
153     if shared_settings['xmpp_to']:
154         xmpp_to = shared_settings['xmpp_to']
155     else:
156         xmpp_to = host
157
158     ssl_connection = \
159         STARTTLS.XMPPConnection(ip_addr, port, ssl, ssl_ctx,
160                                 timeout=timeout, xmpp_to=xmpp_to)
161
162 elif shared_settings['https_tunnel_host']:
163     # Using an HTTP CONNECT proxy to tunnel SSL traffic
164     tunnel_host = shared_settings['https_tunnel_host']
165     tunnel_port = shared_settings['https_tunnel_port']
166     ssl_connection = HTTPSConnection(tunnel_host, tunnel_port, ssl, ssl_ctx,
167                                     timeout=timeout)
168     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(HTTPSConnection):

```

Type number and <Enter> (empty cancels): 2



# Ctags usage: successful symbol lookup

```
30
31 from CtSSLHelper import filter_handshake_exceptions
32 from SSLSocket import SSLSocket
33
34
35 class HTTPSConnection(HTTPConnection):
36     """
37     This class mirrors httplib.HTTPSConnection but uses ctSSL instead of the
38     standard ssl module.
39     For now the way to access low level SSL functions associated with a given
40     HTTPSConnection is to directly access the ssl and ssl_ctx attributes of the
41     object. TODO: change that.
42
43     @type ssl_ctx: ctSSL.SSL_CTX
44     @ivar ssl_ctx: SSL_CTX object for the HTTPS connection.
45
46     @type ssl: ctSSL.SSL
47     @ivar ssl: SSL object for the HTTPS connection.
48     certificates.
49     """
50
51     default_port = HTTPS_PORT
52
53     def __init__(self, host, port=None, ssl=None, ssl_ctx=None,
54                  strict=None, timeout=socket._GLOBAL_DEFAULT_TIMEOUT):
55         """
56         Create a new HTTPSConnection.
57
58         @type host: str
59         @param host: Host name of the server to connect to.
60
61         @type port: int
62         @param port: Port number to connect to. 443 by default.
63
64 [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

```

47 def __init__(self, queue_in, queue_out, available_commands, shared_settings):
48     Process.__init__(self)
49     self.queue_in = queue_in
50     self.queue_out = queue_out
51     self.available_commands = available_commands
52     self.shared_settings = shared_settings
53
54 def run(self):
55     """
56     The process will first complete tasks it gets from self.queue_in.
57     Once it gets notified that all the tasks have been completed,
58     it terminates.
59     """
60     from plugins.PluginBase import PluginResult
61     # Plugin classes are unpickled by the multiprocessing module
62     # without state info. Need to assign shared_settings here
63     for plugin_class in self.available_commands.itervalues():
64         plugin_class._shared_settings = self.shared_settings
65
66     while True:
67
68         task = self.queue_in.get() # Grab a task from queue_in
69
70         if task == None: # All the tasks have been completed
71             self.queue_out.put(None) # Pass on the sentinel to result_queue
72             self.queue_in.task_done()
73             break
74
75         (target, command, args) = task
76         # Instantiate the proper plugin
77         plugin_instance = self.available_commands[command]()
78
79         try: # Process the task

```

```

" 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_out)
    +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

```

[4:1] [sslyze.py] [python][unix-utf-8] L68/251:C1 21%[Git(master)] [Name] sslyze.py

# Cscope

## Serious symbol mangling

[http://cscope.sourceforge.net/cscope\\_vim\\_tutorial.html](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

- I recommend the plugin from [brookhong/cscope.vim](#)
- Query a symbol with `(\)` 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

```

5 @class NSArray, AVAssetInternal;
6
7 @interface AVAsset : NSObject <NSCopying, AVAsynchronousKeyValueLoading> {
8     AVAssetInternal *_assetInternal;
9 }
10
11 @property(readonly) NSArray * availableChapterLocales;
12 @property(readonly) struct { long long x1; int x2; unsigned int x3; long long x4; } duration;
13 @property(readonly) float preferredRate;
14 @property(readonly) float preferredVolume;
15 @property(readonly) struct CGAffineTransform { float x1; float x2; float x3; float x4; float x5; float x6; } preferredTransform;
16 @property(readonly) struct CGSize { float x1; float x2; } naturalSize;
17
18 + (id)assetWithURL:(id)arg1 figPlaybackItem:(struct OpaqueFigPlaybackItem { })arg2 trackIDs:(id)arg3 dynamicBehavior:(BOOL)arg4;
19 + (id)assetWithURL:(id)arg1;
20
21 - (BOOL)isEqual:(id)arg1;
22 - (unsigned int)hash;
23 - (id)copyWithZone:(struct _NSZone { })arg1;
24 - (id)init;
25 - (void)dealloc;
26 - (int)unusedTrackID;
27
28 [2:1] [AVAsset.h] [objc][unix-utf-8] L13/85:C1 6%[Git(master)]
29 1 Frameworks/AVFoundation.framework/AVAsset.h|13| <<<unknown>>> @property(readonly) float preferredRate;
30 2 Frameworks/AVFoundation.framework/AVAsset.h|66| <<<unknown>>> - (float)preferredRate;
31 3 Frameworks/AVFoundation.framework/AVAssetInspector.h|11| <<<unknown>>> @property(readonly) float preferredRate;
32 4 Frameworks/AVFoundation.framework/AVAssetInspector.h|47| <<<unknown>>> - (float)preferredRate;
33 5 Frameworks/AVFoundation.framework/AVFigAssetInspector.h|44| <<<unknown>>> - (float)preferredRate;
34 6 Frameworks/AVFoundation.framework/AVFormatReaderInspector.h|34| <<<unknown>>> - (float)preferredRate;
35
36 [Quickfix List] :cs find t preferredRate

```

# Completion

## Cruise control for completion

- 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`
- Lots of completion mechanisms available, some very heavy, some very light
- VimCompletesMe makes it just work with tab
  - <https://github.com/ajh17/VimCompletesMe/>
- Maybe also check out deoplete, YCM



# Snippets

You will forget how to program forever

- Spits out code snippets from abbreviations
- Easy to define new ones
- Several options, I suggest: <https://github.com/SirVer/ultisnips>
- Note that the default binding is tab, conflicts with tab-completion plugins (VCM)
  - Remap to something else

# Snippets

## 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}
```

---

# Snippets

## Complete a “for” loop

```

54 cCommandLine[sizeof(cCommandLine) - 1] = '\\0';
55
56 if(!EnableDebugPriv()) return 1;
57
58 StartupInfo.cb = sizeof(STARTUPINFO);
59 StartupInfo.lpReserved = NULL;
60 StartupInfo.lpDesktop = NULL;
61 StartupInfo.lpTitle = NULL;
62 StartupInfo.dwFlags = STARTF_USESHOWWINDOW;
63 StartupInfo.wShowWindow = SW_SHOWDEFAULT;
64 StartupInfo.cbReserved2 = 0;
65 StartupInfo.lpReserved2 = NULL;
66
67 //Start MMC
68 //if(!CreateProcess(NULL, "c:\\windows\\system32\\mmc.exe c:\\windows\\system32\\certmgr.msc", NULL, NULL, FALSE, 0, NULL, NULL, &StartupInfo
, &ProcessInfo)) {
69     if(!CreateProcess(NULL, cCommandLine, NULL, NULL, FALSE, 0, NULL, NULL, &StartupInfo, &ProcessInfo)) {
70         printf("CreateProcess error, %d\\n", GetLastError());
71         return 1;
72     }
73
74     //stupid sleep to allow mmc.exe's process space to be fully created before injecting thread
75     Sleep(1000);
76
77     for[
78
79     memset(&Params, 0, sizeof(PARAMS));
80     strcpy_s(Params.DllFileName, cDllFileName);
81     strcpy_s(Params.FunctionName, "Run");
82     Params.LoadLibraryPtr = LoadLibrary;
83     Params.GetProcAddressPtr = GetProcAddress;
84     Params.ExitThreadPtr = ExitThread;
85
[2:3] [jailbreak.cpp][+] [cpp][dos-utf-8] L77/239:C8 25%[Git(master)]
-- INSERT --

```

# Snippets

## Snippet success

```

54  cCommandLine[sizeof(cCommandLine) - 1] = '\0';
55
56  if(!EnableDebugPriv()) return 1;
57
58  StartupInfo.cb = sizeof(STARTUPINFO);
59  StartupInfo.lpReserved = NULL;
60  StartupInfo.lpDesktop = NULL;
61  StartupInfo.lpTitle = NULL;
62  StartupInfo.dwFlags = STARTF_USESHOWWINDOW;
63  StartupInfo.wShowWindow = SW_SHOWDEFAULT;
64  StartupInfo.cbReserved2 = 0;
65  StartupInfo.lpReserved2 = NULL;
66
67  //Start MMC
68  //if(!CreateProcess(NULL, "c:\\windows\\system32\\mmc.exe c:\\windows\\system32\\certmgr.msc", NULL, NULL, FALSE, 0, NULL, NULL, &StartupInfo
, &ProcessInfo)) {
69      if(!CreateProcess(NULL, cCommandLine, NULL, NULL, FALSE, 0, NULL, NULL, &StartupInfo, &ProcessInfo)) {
70          printf("CreateProcess error, %d\n", GetLastError());
71          return 1;
72      }
73
74      //stupid sleep to allow mmc.exe's process space to be fully created before injecting thread
75      Sleep(1000);
76
77      for (i = 0; i < count; i++) {
78          /* code */
79      }
80
81      memset(&Params, 0, sizeof(PARAMS));
82      strcpy_s(Params.DllFileName, cDllFileName);
83      strcpy_s(Params.FunctionName, "Run");
84      Params.LoadLibraryPtr = LoadLibrary;
85      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
- Check out: <http://vimcasts.org/episodes>
- Short story: `:Gw`, `:Gr`, `:Gstatus`, `:Gcommit`, `:Gdiff`

# Fugitive

:Gstatus

```

1 On branch master
2 # Changes not staged for commit:
3 #   (use "git add <file>..." to update what will be committed)
4 #   (use "git checkout -- <file>..." to discard changes in working directory)
5 #
6 #    modified:   .vim/spell/en.utf-8.add.spl
7 #    modified:   .vim/vimchat.vim
8 #    modified:   .vimrc
9 #    modified:   .xombrero.conf
10 #
11 # Untracked files:
12 #   (use "git add <file>..." to include in what will be committed)
13 [3:12] [index][Preview][-][R0] [gitcommit][unix-utf-8] L1/15:C1 Top[Git(master)]
14 if $DISPLAY != ""
15     set mouse=a           " Turn this off for console-only mode
16     set selectmode+=mouse " Allow the mouse to select
17 endif
18 set et                    " expand tabs
19 set diffopt+=iwhite       " ignore whitespace in diffs
20 set cursorline            " highlight the line the cursor is on
21 set t_Co=256              " use 256 colors
22 set hidden
23 set novb
24 set number
25 set viewdir=$HOME/.views " keep view states out of my .vim
26 set pumheight=15         " trim down the completion popup menu
27 set shortmess+=atIoT     " save space in status messages
28 set scrolloff=2          " 3 lines of buffer before scrolling
29 set ignorecase            " case insensitive searches
30 set smartcase             " unless you type uppercase explicitly
31 set wildmode=list:longest " shows a list of candidates when tab-completing
32 set hlsearch              " highlight all search matches
33 set nojoinspaces          " disallow two spaces after a period when joining
34 [3:11] [.vimrc] [vim][unix-utf-8] L60/287:C9 15%[Git(master)]

```

# Fugitive

:Gcommit

```

1 Fix gui detection issues
2 # Please enter the commit message for your changes. Lines starting
3 # with '#' will be ignored, and an empty message aborts the commit.
4 # On branch master
5 # Changes to be committed:
6 #   (use "git reset HEAD <file>..." to unstage)
7 #
8 #   modified:   .vim/vimchat.vim
9 #   modified:   .vimrc
10 #
11 # Changes not staged for commit:
12 #   (use "git add <file>..." to update what will be committed)
13 #   (use "git checkout -- <file>..." to discard changes in working directory)
14 #
15 #   modified:   .vim/spell/en.utf-8.add.spl
[3:19] [COMMIT_EDITMSG][+] [gitcommit][unix-utf-8] L1/21:C25 Top[Git(master)]
46 endif
47 set et                " expand tabs
48 set difft+ iwwhite    " ignore whitespace in diffs
49 set cursorline        " highlight the line the cursor is on
50 set t_Co=256          " use 256 colors
51 set hidden
52 set novb
53 set number
54 set viewdir=$HOME/.views " keep view states out of my .vim
55 set pumheight=15       " trim down the completion popup menu
56 set shortmess+=atIoT   " save space in status messages
57 set scrolloff=2        " 3 lines of buffer before scrolling
58 set ignorecase         " case insensitive searches
59 set smartcase          " unless you type uppercase explicitly
60 set wildmode=list:longest " shows a list of candidates when tab-completing
61 set hlsearch           " highlight all search matches
62 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

```

80 \end{frame}
81
82 \section{Auditing tools}
83 \subsection{Ctags}
84 \begin{frame}[fragile]{Exuberant Ctags}
85 \url{http://ctags.sourceforge.net/}
86
87 \begin{itemize}
88 \item Generates an index of symbols
89 \item Usage: at root of source tree, {\tt ctags -R .}
90 \item In vim, {\tt set tags=./tags} or whatever path you
91 \item When cursor is on a function/method/whatever, {\tt
92 its definition
93 \item If ambiguous, a select list is displayed
94 \end{itemize}
95 \end{frame}
96
97 \begin{frame}[fragile]{Ctags usage: jump to definition}
98 \begin{figure}[ht]
+ 99 +-- 4 lines: \center \includegraphics[width=1.0\textwidth]{img
103 \begin{frame}[fragile]{Ctags usage: successful symbol lookup}
104 \begin{figure}[ht]
105 \center \includegraphics[width=1.0\textwidth]{images/cta
106 \end{figure}
107 \end{frame}
108
109
110
111 \end{frame}
112
113 \section{Auditing tools}
114 \subsection{Ctags}
115 \begin{frame}[fragile]{Exuberant Ctags} Tokenize all the things
116 \url{http://ctags.sourceforge.net/}
117
118 \begin{itemize}
119 \item Generates an index of symbols
120 \item Usage: at root of source tree, {\tt ctags -R .}
121 \item In vim, {\tt set tags=./tags} or whatever path you
122 \item When cursor is on a function/method/whatever, {\tt
123 jumps to its definition
124 \item If ambiguous, a select list is displayed
125 \end{itemize}
126 \end{frame}
127
128 \begin{frame}[fragile]{Ctags usage: jump to definition}
129 \begin{figure}[ht]
130 + 131 +-- 4 lines: \center \includegraphics[width=1.0\textwidth]{img
132 \begin{frame}[fragile]{Ctags usage: successful symbol lookup}
133 \begin{figure}[ht]
134 \center \includegraphics[width=1.0\textwidth]{images/cta
135 \end{figure}
136 \end{frame}
137
138 \subsection{Tagbar}
139 \begin{frame}[fragile]{Tagbar}{Intelligent symbol browsing}
140 \begin{itemize}
141 \item Uses ctags to generate a symbol list
142 \item Smart about identifying different symbol types
143 \item Scope-aware
144 \item Displays basic function signatures
145 \end{itemize}
146 \end{frame}
147
148
149
150
151

```

[10:90] <action.tex> [tex][unix-utf-8] L90/261:C1 34% [Git:0(master)] [10:88] <notation.tex> [tex][unix-utf-8] L125/441:C1 28% [Git(master)]





Your non-GUI git GUI

- Plugin for Fugitive
- Commit browsing
- Perusing file history
- Jumping into vimdiff
- <https://github.com/junegunn/gv.vim>

Thiel



## File mode

2 o/t/y/yara.cpp o/t/y/yara.cpp

```

1 2017-05-26 bf2457ff Address YARA hardcoded home folder issu
2 2017-04-28 4372785d Refactor build logic to allow optional:
3 2016-02-21 9a54af29 Bump sqlite to 3.11.0 (Teddy Reed)
4 2016-02-11 21c2237e [osquery] Update copyright headers to n
5 2015-10-21 1d9695ac eliminated some warnings from Clang 3.7
6 2015-09-20 d042967f Fix YARA sigfile caching (Teddy Reed)
7 2015-09-02 a1403334 [fix #1390] query pack re-org (Mike Arp
8 2015-08-13 68d7a6e0 Speedup type conversions, yara, and 10.
9 2015-07-27 698e226b Add tags and strings columns to YARA ta
10 2015-06-03 a1059248 Move specs to a top-level path, add que
11 2015-05-29 6558f605 Implement process related tables on Fre
12 2015-05-23 5969ae4f Clean up TLS-version from OpenSSL detec
13 2015-04-29 546d2981 Move yara relative paths to /etc/osquer
14 2015-04-26 67bf0992 YARA tests, SQL matching, sigfile loadi
15 2015-04-26 fcd66c4b Move yara out of core/SDK into addition
16 2015-04-26 a9f66fa3 Major YARA refactor and enhancements (W

```

```

1 tree a052c0553f3b24e2f9014acba5b5e8f9881f44e8
2 parent eefccf27b1c2c1ba26125ff4334bde22ed27b3c8
3 author Teddy Reed <teddy@prosaunce.org> Thu Aug 13 18:04:03
  2015 -0700
4 committer Teddy Reed <teddy@prosaunce.org> Thu Aug 13 18:04:
  03 2015 -0700
5
6 Speedup type conversions, yara, and 10.10 symbols at
  runtime
7
8
9 diff --git a/include/osquery/tables.h b/include/osquery/
  tables.h
10 index b7a6369b..5832c532 100644
11 --- a/include/osquery/tables.h
12 +++ b/include/osquery/tables.h
13 @@ -10,6 +10,7 @@
14
15 #pragma once
16
17 #include <deque>
18 #include <map>
19 #include <memory>
20 #include <vector>
21 @@ -77,7 +78,7 @@ namespace osquery {
22 /// Helper alias for TablePlugin names.
23 typedef std::string TableName;
24 typedef std::vector<std::pair<std::string, std::string> >
  TableColumns;
25 -typedef std::map<std::string, std::vector<std::string> >

```

# Precise Code Tracking

- Precise code tracking: <https://github.com/d0c-s4vage/pct-vim>
- Add todos, findings, annotations, or mark reviewed
- Show a report of all of the above

## Precise Code Tracking

```
[1:1] [[No Name]] L0/1:C0 All[Git(master)]
[!] Could not find the database, where should it be created?
[!] Annotation location options:
[!] 0 - /Users/det/git/osquery/pct.sqlite
[!] 1 - /Users/det/git/pct.sqlite
[!] 2 - /Users/det/pct.sqlite
[!] 3 - /Users/pct.sqlite
[!] 4 - /pct.sqlite
[!] 5 - /pct.sqlite
Where would you like to create the database? (0-5): 0[!]
[+] Using database at /Users/det/git/osquery/pct.sqlite
[✓] found annotations database at /Users/det/git/osquery/pct.sqlite
Press ENTER or type command to continue
```

# Precise Code Tracking

```
19 namespace osquery {
20 namespace tables {
21
22 Mutex grpEnumerationMutex;
23
24 QueryData genGroups(QueryContext& context) {
25     QueryData results;
RR 26     struct group* grp = nullptr;
RR 27     std::set<long> groups_in;
RR 28
RR 29     WriteLock lock(grpEnumerationMutex);
RR 30     setgrent();
31     while ((grp = getgrent()) != nullptr) {
32         if (std::find(groups_in.begin(), groups_in.end(), grp->gr_gid) ==
33             groups_in.end()) {
?? 34         Row r;
35             r["gid"] = INTEGER(grp->gr_gid);
36             r["gid_signed"] = INTEGER((int32_t)grp->gr_gid);
37             r["groupname"] = TEXT(grp->gr_name);
38             results.push_back(r);
39             groups_in.insert(grp->gr_gid);
40         }
41     }
```

osquery/tables/system/freebsd/groups.cpp - 10%, 0 fndg, 1 todo, 1 note

[+] NOTE (34): TODO Add comments

# Suggested Reading

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

# Questions?

Check out my setup at

<https://github.com/lxcode/dotfiles>



```
" Abbreviations {{{
abbr guys folks
abbr shruggie "~\_ ( )_/"
abbr ty Thanks,
        \<CR>David

" }}}

" Keymappings {{{
"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>

" Make Y behave like C and D
nnoremap Y y$

" Use , and space in addition to \ for the leader
let mapleader = ", "
nmap \ ,
nmap <space> ,

" save my pinky
nore ; :

" auto-format the current paragraph
```

```

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>
" fix spelling of last misspelled word
nmap <F1> [s1z=<C-o>
imap <F1> <Esc>[s1z=<C-o>a
nmap <Leader>fs [s1z=<C-o>
" Poor man's cscope - grep for symbol under cursor
nnoremap gr :grep '\b<cword>\b' *<CR>
" Clean up left side
nmap <F2> :set nonu foldcolumn=0<CR>:QuickfixsignsToggle<CR>
" Show netrw sidebar
map <silent> <F9> :Lexplore<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>
" Toggle the quickfix window
nnoremap <silent> <C-c> :call ToggleQuickfix()<cr>

```

```

" Window movement
nnoremap <C-j> <C-w>j
nnoremap <C-k> <C-w>k
nnoremap <C-h> <C-w>h
nnoremap <C-l> <C-l>k

" Keep selected blocks selected when shifting
vmap > >gv
vmap < <gv

" Move visual blocks up and down
vnoremap J :m '>+1<CR>gv=gv
vnoremap K :m '<-2<CR>gv=gv

" day+time / day+date
nmap <Leader>dt "=strftime("%c")<CR>P"
nmap <Leader>dd "=strftime("%y-%m-%d")<CR>P"

" Change to the directory of the current file
nmap <C-d> :lcd %:h \|| :pwd<CR>

" Fix whitespace
nmap <Leader>fw :StripWhitespace<CR>

" Quick exits
nmap zz ZZ

" }}}

" Settings {{{
syntax on

```

```

filetype plugin on
filetype indent on
"helptags ~/.vim/doc

if $DISPLAY != ""
    "set cursorline          " I like this, but damn is it slow
    set clipboard=unnamed
    set mouse=a              " Turn this off for console-only mode
    if !has('nvim')
        set ttymouse=xterm2
    endif
endif

set et                      " expand tabs
set diffopt+=iwhite,vertical,filler " ignore whitespace in diffs
set hidden                  " allow hidden buffers
set noerrorbells vb t_vb=  " no bells
set number                  " 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 wildignorecase          " same for directories and ex commands
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 foldcolumn=0 " I never use this.
set nojoinspaces " disallow two spaces after a period when joining
set formatoptions=qjnrtrlmnc " auto-formatting style
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\\|+
set grepprg=grep -R --exclude=*.aux --exclude=tags --exclude=*scope.out --color=always -nIH\
set cpoptions=Bft
set completeopt=menuone,longest
autocmd FileType *
    \ if &omnifunc == "" |
    \     setlocal omnifunc=syntaxcomplete#Complete |
    \ endif
set tags=tags,./tags
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 softtabstop=4
set backspace=indent,eol,start
set ruler                " show position in file
set title
set titlestring=%t%( \ %M%)(\ ({expand("\":p:h\"))})%( \ %a%)
set titleold=""
set ttimeout
set ttimeoutlen=100      " Make it so Esc enters Normal mode right away
if has('nvim')
    set ttimeoutlen=-1
endif
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 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=0     " the default level of fold nesting on startup
set autoread             " Disable warning about file change to writable
set conceallevel=0       " Don't hide things by default
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

```

```

" Use pipe cursor on insert

```

```

let &t_SI = "\<esc>[5 q"

```

```

let &t_SR = "\<esc>[3 q"

```

```

let &t_EI = "\<esc>[1 q"

```

```

" Same in neovim

```

```

let $NVIM_TUI_ENABLE_CURSOR_SHAPE = 1

```

```

" colors

```

```

if $TERM == 'xterm-kitty'

```

```

    if exists('$TMUX')

```

```

        set t_Co=256

```

```

    else

```

```

        set termguicolors

```

```

    endif

```

```

else

```

```

    set t_Co=256

```

```

endif

```

```

colorscheme lx-truecolor

```

```

" }}}

```

```

" Plugins

```

```
call plug#begin('~/.vim/plugged')
Plug 'AndrewRadev/id3.vim', { 'for': 'audio.flac' }
Plug 'ConradIrwin/vim-bracketed-paste'
Plug 'SirVer/ultisnips' | Plug 'honza/vim-snippets'
Plug 'SolaWing/vim-objc-syntax', { 'for': 'objc' }
Plug 'Yggdroot/indentLine', { 'on': 'IndentLinesEnable' }
Plug 'ajh17/VimCompletesMe'
Plug 'blindFS/vim-taskwarrior', { 'on': 'TW' }
Plug 'brookhong/cscope.vim'
Plug 'christianrondeau/vim-base64'
Plug 'd0c-s4vage/pct-vim'
Plug 'fidian/hexmode', { 'on': 'Hexmode' }
Plug 'goldfeld/vim-seek'
Plug 'gorkunov/smartpairs.vim'
Plug 'guns/xterm-color-table.vim', { 'on': 'XtermColorTable' }
Plug 'jamessan/vim-gnupg'
Plug 'jiangmiao/auto-pairs'
Plug 'jremmen/vim-ripgrep', { 'on': 'Rg' }
Plug 'junegunn/fzf', { 'dir': '~/.fzf', 'do': './install --all' }
Plug 'junegunn/fzf.vim'
Plug 'junegunn/gv.vim', { 'on': ['GV', 'GV!'] }
Plug 'junegunn/vim-fnr'
Plug 'junegunn/vim-pseudocl'
Plug 'junegunn/vim-slash'
```



```
Plug 'kergoth/vim-hilinks'
Plug 'lervag/vimtex', { 'for': 'latex' }
Plug 'majutsushi/tagbar', { 'on': 'TagbarToggle' }
Plug 'millermedeiros/vim-statline'
Plug 'ntpeters/vim-better-whitespace'
Plug 'rhyds/clever-f.vim'
Plug 'solarnz/thrift.vim', { 'for': 'thrift' }
Plug 'tomtom/quickfixsigns_vim'
Plug 'tpope/vim-afterimage'
Plug 'tpope/vim-characterize'
Plug 'tpope/vim-eunuch'
Plug 'tpope/vim-fugitive'
Plug 'tpope/vim-repeat'
Plug 'tpope/vim-vinegar'
Plug 'vim-utils/vim-man'
Plug 'will133/vim-dirdiff'
call plug#end()
```

```
" Don't load plugins that have unmet dependencies
```

```
if !executable('task')
    let g:loaded_taskwarrior = 1
endif
```

```
if !has('python3')
```

```

    let g:loaded_pct = 1
endif

" netrw {{{
let g:netrw_liststyle=0
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']
let g:quickfixsigns_echo_balloon = 1
" 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('abcdefghijklmnopqrstuvxyz', '\\zs')
let g:quickfixsign_use_dummy = 0
let g:quickfixsigns#vcsdiff#highlight = {'DEL': 'QuickFixSignsDiffDeleteLx', 'ADD': 'QuickFixSignsDiffAddLx',
" }}}

" buftabline {{{
let g:buftabline_show=1

```

```

let g:buftabline_separators=1
" }}}

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

" ultisnips {{{
let g:UltiSnipsExpandTrigger = "<C-l>"
" }}}

" cscope {{{
let g:cscope_interested_files = '\.java$\|\.php$\|\.h$\|\.hpp$\|\.cpp$\|\.c$\|\.m$\|\.swift$\|\.py$\|\.hs$'
let g:cscope_split_threshold = 99999
let g:cscope_auto_update = 0
nnoremap <leader>fa :call CscopeFindInteractive(expand('<cword>'))<CR>
nnoremap <leader>l :call ToggleLocationList()<CR>
nnoremap <leader>fs :call CscopeFind('s', expand('<cword>'))<CR>
nnoremap <leader>fg :call CscopeFind('g', expand('<cword>'))<CR>
nnoremap <leader>fd :call CscopeFind('d', expand('<cword>'))<CR>
nnoremap <leader>fc :call CscopeFind('c', expand('<cword>'))<CR>
nnoremap <leader>ft :call CscopeFind('t', expand('<cword>'))<CR>
nnoremap <leader>fe :call CscopeFind('e', expand('<cword>'))<CR>

```

```

nnoremap <leader>ff :call CscopeFind('f', expand('<cword>'))<CR>
nnoremap <leader>fi :call CscopeFind('i', expand('<cword>'))<CR>
" }}}

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

" ripgrep {{{
let g:rg_highlight = 1
" "}}}

" FZF {{{
set rtp+=~/ .fzf
set rtp+=/usr/local/opt/fzf
nmap <C-e> :Files<CR>
nmap <C-g> :GFiles<CR>
nmap <leader>m :History<CR>
nmap <leader>e :Files<CR>
nmap <Leader>t :Tags<CR>
nmap <Leader>b :BTags<CR>
nmap <C-j> :call fzf#vim#tags(expand('<cword>'), {'options': '--exact --select-1 --exit-0'})<CR>

```

```
let g:fzf_tags_command = '/usr/local/bin/ctags -R'
```

```
function! s:build_quickfix_list(lines)
  call setqflist(map(copy(a:lines), '{ "filename": v:val }'))
  copen
  cc
endfunction
```

```
let g:fzf_action = {
  \ 'ctrl-q': function('s:build_quickfix_list'),
  \ 'ctrl-t': 'tab split',
  \ 'ctrl-x': 'split',
  \ 'ctrl-v': 'vsplit' }
```

```
command! -bang -nargs=* F
  \ call fzf#vim#grep(
  \   'rg --column --line-number --no-heading --fixed-strings --ignore-case --no-ignore --hidden --follow --glob'
  \   <bang>0 ? fzf#vim#with_preview('up:60%')
  \   : fzf#vim#with_preview('right:50%:hidden', '?'),
  \   <bang>0)
```

```
command! -bang -nargs=? -complete=dir Files
  \ call fzf#vim#files(<q-args>, fzf#vim#with_preview(), <bang>0)
" }}}}
```

```

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

" tagbar {{{
let g:tagbar_iconchars = [ ' ', ' ' ]
" }}}

" augroups {{{
augroup cjava
    au!
    au BufWinEnter *.mchly set 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,*.hs set number
    au BufWinLeave *.mchly mkview
    au BufWinEnter *.mchly silent loadview
    au BufWinLeave *.cpp,*.java,*.hs mkview
    au BufWinEnter *.cpp,*.java,*.hs silent loadview

```

```
augroup end
```

```
augroup html
```

```
  au!
  au FileType html set spell wrapmargin=5 wrapscan number
  au FileType html set wrapscan&
  au BufWinLeave *.htm* mkview
  au BufWinEnter *.htm* silent loadview
```

```
augroup end
```

```
augroup pythonphp
```

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

```
augroup end
```

```
augroup markdown
```

```
  " Don't highlight underscores
  syn match markdownError "\w\@<=\w\@="
  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, normal zR
au BufWinEnter *.md,*.notes,*mutt*, imap >> <C-t>
au BufWinEnter *.md,*.notes,*mutt*, imap << <C-d>
au FileType markdown set spell 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, set number
  au FileType qf, noremap ' <CR><C-W><C-P>j
  au FileType qf, noremap <silent> <buffer> <right> :cnew<CR>
  au FileType qf, noremap <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,*.odp,*.otp,*.odg,*.otg call zip#Browse(expand("<amatch>"))
augroup end

```



```
augroup misc
```

```

au FileType git set foldlevel=99
au BufWinEnter *.applescript set filetype=applescript
au BufWinEnter *.nmap, set syntax=nmap
au BufWinEnter *.scala, set filetype=scala
au BufWinEnter *.proto, set filetype=proto
au BufWinEnter *.dtrace, set filetype=D
au BufWinEnter *.less, set filetype=css
au BufWinEnter *.fugitiveblame,*.diff, set number
au BufWinLeave *.txt,*.conf,.vimrc,*.notes mkview
au BufWinEnter *.txt,*.conf,.vimrc,*.notes silent loadview
au BufWinEnter .vimrc set foldmethod=marker
au FileType json set conceallevel=0
au FileType make set diffopt=-iwhite
au FileType mail set spell complete+=k nonu comments+=b:-,b:+,b:*,b:+,n:>
au FileType mail if executable("par") | set formatprg=par | endif
au FileType mail map <F8> :%g/^> >/d<CR>gg10j
au FileType mail StripWhitespace
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
" 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
au GUIEnter * set visualbell t_vb=
augroup end

" }}}

" Custom functions {{{
" Quickfix toggle

let g:quickfix_is_open = 0

function! ToggleQuickfix()
    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()
        copen
        let g:quickfix_is_open = 1
    endif
endfunction

```

```

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

" 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

" I use this to highlight the match from grep, but keep quickfix syntax
" highlighting intact. Detects Linux due to the different escape sequences of
" GNU grep.
command -bar GrepColors call GrepColors()
function GrepColors()
    set conceallevel=3
    set cocu=nv

    if system('uname')==~'Linux'

```

```

    syn region ansiRed start="\e\[01;31m"me=e-2 end="\e\[m"me=e-3 contains=ansiConceal
    syn match ansiConceal contained conceal "\e\[\\(\\d*;\\)*\\d*m"
    syn match ansiStop conceal "\e\[m"
elseif system('uname')==~'FreeBSD'
    syn region ansiRed start="\e\[01;31m"me=e-2 end="\e\[00m"me=e-5 contains=ansiConceal
    syn match ansiConceal contained conceal "\e\[\\(\\d*;\\)*\\d*m"
    syn match ansiStop conceal "\e\[00m\e\[K"
else
    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"
    syn match ansiStop conceal "\e\[m\e\[K"
endif

hi ansiRed ctermfg=197 guifg=#FF005F cterm=none gui=none
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/git/graudit/graudit -B -x 'cscope.*' -c0 -d " . a:db . " . | awk 'length($0) < 200' > /t
    copen
    cf /tmp/graudit.out
endfunction
" }}}

if filereadable("~/vimrc-local")
    source ~/.vimrc-local
endif

```

---