



首页
最新文章
在线课程
业界
开发
IT技术
设计
创业
IT职场
投稿
更多 »

- 导航条 -

伯乐在线 > 首页 > 所有文章 > IT技术 > Vim自动补全神器：YouCompleteMe

Vim自动补全神器：YouCompleteMe

2014/08/10 · IT技术 · 11 评论 · Vim, YouCompleteMe

分享到：

45

前端开发工具技巧介绍—Sublime篇
瀑布流布局
SEO在网页制作中的应用
MySQL开发技巧（三）

原文出处：[marchtea 的博客](#) 欢迎分享原创到[伯乐头条](#)

第一次听说这个插件还是在偶然的情况下看到别人的博客，听说了这个插件的大名。本来打算在实训期间来完成安装的，无奈网实在不给力，也就拖到了回家的时候。在开始准备工作的时候就了解到这个插件不是很容易安装，安装的时候果然名不虚传。(关于这方面的内容，请查看另一篇文章)不过，有付出总有回报，安装之后用上这个插件，真心为这个插件的强大所折服。

那这个插件有何不同？

YouCompleteMe的特别之处

基于语义补全

总所周知，vim是一款文本编辑器。也就是说，其最基础的工作就是编辑文本，而不管该文本的内容是什么。在vim被程序员所使用后，其慢慢的被肩负了与IDE一样的工作，文本自动补全(ie.acp.omnicppcompleter)，代码检查(syntastic)等工作。

针对文本自动补全这个功能来说，主要有两种实现方式。

- 基于文本

我们常用的omnicppcompleter,acp,vim自带的c-x, c-n的实现方式就是基于文本。更通俗的说法，其实就是一个字：

猜

其通过文本进行一些正则表达式的匹配，再根据生成的tags(利用ctags生成)来实现自动补全的效果。

- 基于语义

顾名思义，其是通过分析源文件，经过语法分析以后进行补全。由于对源文件进行分析，基于语义的补全可以做到很精确。但是这显然是vim所不可能支持的。而且经过这么多年发展，由于语法分析有很高的难度，也一直没有合适的工具出现。直到，由apple支持的clang/llvm横空出世。YouCompleteMe也正是在clang/llvm的基础上进行构建的。

整合实现了多种插件

- clang_complete

- AutoComplPop
- Supertab
- neocomplcache
- [Syntastic](#)(类似功能,仅仅针对c/c++/obj-c代码)

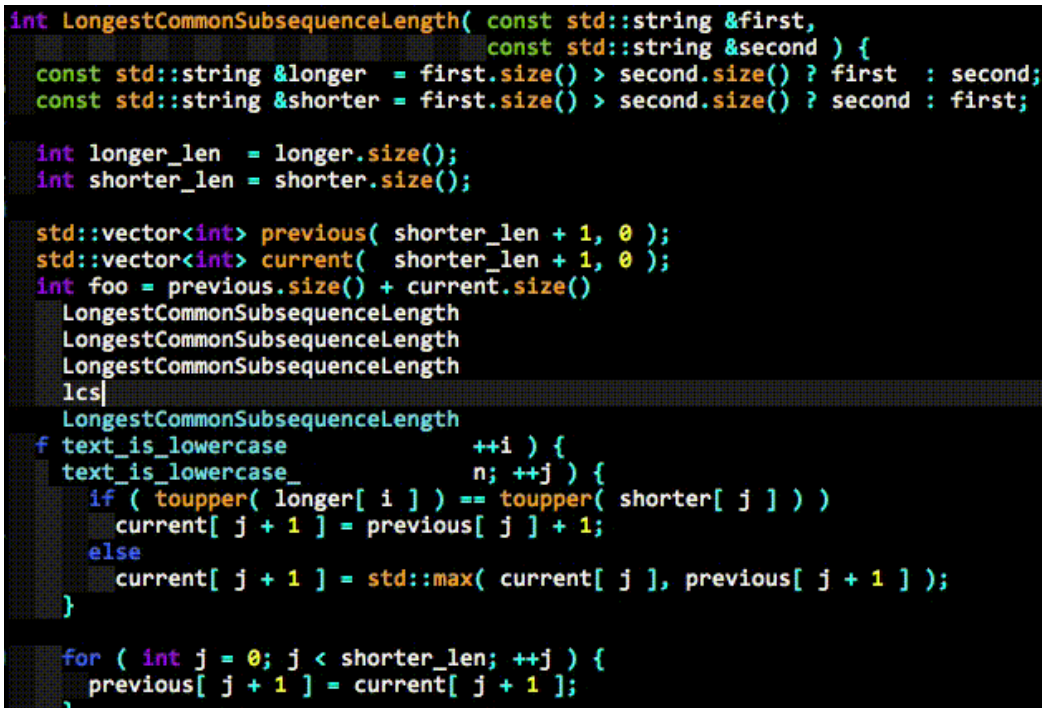
支持语言

- C
- C++
- obj-c
- C#
- python

对于其他的语言, 会调用vim设置的omnifunc来匹配, 因此同样支持php,ruby等语言。

已知的有 * javascript — [tern_for_vim](#) * ruby/java — [eclim](#)

使用效果图



```
int LongestCommonSubsequenceLength( const std::string &first,
                                    const std::string &second ) {
    const std::string &longer  = first.size() > second.size() ? first  : second;
    const std::string &shorter = first.size() > second.size() ? second : first;

    int longer_len  = longer.size();
    int shorter_len = shorter.size();

    std::vector<int> previous( shorter_len + 1, 0 );
    std::vector<int> current(  shorter_len + 1, 0 );
    int foo = previous.size() + current.size()
    LongestCommonSubsequenceLength
    LongestCommonSubsequenceLength
    LongestCommonSubsequenceLength
    lcs|
    LongestCommonSubsequenceLength
    f text_is_lowercase      ++i ) {
    text_is_lowercase_      n; ++j ) {
        if ( toupper( longer[ i ] ) == toupper( shorter[ j ] ) )
            current[ j + 1 ] = previous[ j ] + 1;
        else
            current[ j + 1 ] = std::max( current[ j ], previous[ j + 1 ] );
    }

    for ( int j = 0; j < shorter_len; ++j ) {
        previous[ j + 1 ] = current[ j + 1 ];
    }
}
```

使用感受

- 和IDE一样, 自动补全,
- 根据include的文件进行补全
- 不用再蹩脚的生成tags
- 补全非常精准, 而且速度很快, 不会有延迟(以前在大项目上, acp用起来实在是很卡)
- 支持类似tags的跳转, 跳到定义处以及使用处
- 出错提示很智能, 并且用起来真的是如丝般柔滑, 不用输入:w进行强制检测

安装

说完了那么多好处, 就要说到安装了。不同于以往其他vim插件, YCM是一款编译型的插件。在下载完后, 需要手动编译后才能使用。对应其他的插件来说, 仅仅就是把.vim的文件丢到相应文件夹下就可以。而这也加大了使用YCM的难度。

安装准备

- 最新版的Vim(7.3.584+), 编译时添加+python标志(已经安装的可以通过vim --version查看)
- cmake(mac可以通过[homebrew](#)安装, brew install cmake,ubuntu可以通过sudo apt-get install cmake)
- 安装[vundle](#)插件, 用于安装管理vim的插件

mac下快速安装

在.vimrc中添加下列代码

```
1 | Bundle 'Valloric/YouCompleteMe'
```

保存退出后打开vim, 在正常模式下输入

```
1 | :BundleInstall
```

等待vundle将YouCompleteMe安装完成

而后进行编译安装:

```
1 | cd ~/.vim/bundle/YouCompleteMe
2 | ./install --clang-completer
```

如果不需要c-family的补全, 可以去掉--clang-completer。如果需要c#的补全, 请加上--omnisharp-completer。

正常来说,YCM会去下载clang的包, 如果已经有, 也可以用系统--system-libclang。

就这样, 安装结束。打开vim, 如果没有提示YCM未编译, 则说明安装已经成功了。

手动编译安装

安装的脚本并不是什么时候都好用, 至少对我来说是这样的。安装完之后出现了问题, 参考[issue#809](#)。

在用:BundleInstall安装完成或者使用

```
1 | git clone --recursive https://github.com/Valloric/YouCompleteMe.git
```

获取最新的仓库, 而后使用git submodule update --init --recursive确认仓库的完整性后, 开始安装流程。

1. 下载最新的clang二进制文件 YCM要求clang版本 > 3.2, 一般来说都是[下载最新的](#)。
2. 安装python-dev.(ubuntu下使用sudo apt-get install python-dev,mac下默认提供, 否则请安装[command line tools](#))
3. 编译

```
1 | cd ~
2 | mkdir ycm_build
3 | cd ycm_build
4 | cmake -G "Unix Makefiles" -DPATH_TO_LLVM_ROOT=~/.vim/bundle/YouCompleteMe/cpp make ycm_sup
```

这里需要注意的是, ~/.vim/bundle/YouCompleteMe/cpp/llvm_root_dir中包含的是根据第一步下载的压缩包解压出来的内容(包括include, bin等等文件)。

这样就完成了,开始感受YCM提供的完全不逊色于大型IDE所提供的自动补全功能吧。

配置

不同于很多vim插件, YCM首先需要编译, 另外还需要有配置。在vim启动后, YCM会找寻当前路径以及上层路径的.ycm_extra_conf.py.在~/.vim/bundle/YouCompleteMe/cpp/ycm/.ycm_extra_conf.py中提供了默认的模板。也可以参考我的(就在模板上改改而已)。不过这个解决了标准库提示找不到的问题。

一般来说,我会在~目录下放一个默认的模板, 而后再根据不同的项目在当前目录下再拷贝个.ycm_extra_conf.py。

```
1 | # This file is NOT licensed under the GPLv3, which is the license for the rest
2 | # of YouCompleteMe.
3 | #
4 | # Here's the license text for this file:
5 | #
6 | # This is free and unencumbered software released into the public domain.
7 | #
8 | # Anyone is free to copy, modify, publish, use, compile, sell, or
9 | # distribute this software, either in source code form or as a compiled
10 | # binary, for any purpose, commercial or non-commercial, and by any
11 | # means.
12 | #
13 | # In jurisdictions that recognize copyright laws, the author or authors
14 | # of this software dedicate any and all copyright interest in the
15 | # software to the public domain. We make this dedication for the benefit
16 | # of the public at large and to the detriment of our heirs and
17 | # successors. We intend this dedication to be an overt act of
18 | # relinquishment in perpetuity of all present and future rights to this
19 | # software under copyright law.
20 | #
21 | # THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
22 | # EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
23 | # MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
24 | # IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR
25 | # OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
26 | # ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
27 | # OTHER DEALINGS IN THE SOFTWARE.
28 | #
29 | # For more information, please refer to <http://unlicense.org/>
30 |
31 | import os
32 | import ycm_core
33 |
34 | # These are the compilation flags that will be used in case there's no
35 | # compilation database set (by default, one is not set).
```

```

36 # CHANGE THIS LIST OF FLAGS. YES, THIS IS THE DROID YOU HAVE BEEN LOOKING FOR.
37 flags = [
38     '-Wall',
39     '-Wextra',
40     #'-Werror',
41     #'-Wc++98-compat',
42     '-Wno-long-long',
43     '-Wno-variadic-macros',
44     '-fexceptions',
45     '-stdlib=libc++',
46     # THIS IS IMPORTANT! Without a "-std=<something>" flag, clang won't know which
47     # language to use when compiling headers. So it will guess. Badly. So C++
48     # headers will be compiled as C headers. You don't want that so ALWAYS specify
49     # a "-std=<something>".
50     # For a C project, you would set this to something like 'c99' instead of
51     # 'c++11'.
52     '-std=c++11',
53     # ...and the same thing goes for the magic -x option which specifies the
54     # language that the files to be compiled are written in. This is mostly
55     # relevant for c++ headers.
56     # For a C project, you would set this to 'c' instead of 'c++'.
57     '-x',
58     'c++',
59     '-I',
60     '.',
61     '-isystem',
62     '/usr/include',
63     '-isystem',
64     '/usr/local/include',
65     '-isystem',
66     '/Library/Developer/CommandLineTools/usr/include',
67     '-isystem',
68     '/Library/Developer/CommandLineTools/usr/bin/./lib/c++/v1',
69 ]
70
71 # Set this to the absolute path to the folder (NOT the file!) containing the
72 # compile_commands.json file to use that instead of 'flags'. See here for
73 # more details: http://clang.llvm.org/docs/JSONCompilationDatabase.html
74 #
75 # Most projects will NOT need to set this to anything; you can just change the
76 # 'flags' list of compilation flags. Notice that YCM itself uses that approach.
77 compilation_database_folder = ''
78
79 if os.path.exists( compilation_database_folder ):
80     database = ycm_core.CompilationDatabase( compilation_database_folder )
81 else:
82     database = None
83
84 SOURCE_EXTENSIONS = [ '.cpp', '.cxx', '.cc', '.c', '.m', '.mm' ]
85
86 def DirectoryOfThisScript():
87     return os.path.dirname( os.path.abspath( __file__ ) )
88
89 def MakeRelativePathsInFlagsAbsolute( flags, working_directory ):
90     if not working_directory:
91         return list( flags )
92     new_flags = []
93     make_next_absolute = False
94     path_flags = [ '-isystem', '-I', '-isystem', '--sysroot=' ]
95     for flag in flags:
96         new_flag = flag
97
98         if make_next_absolute:
99             make_next_absolute = False
100             if not flag.startswith( '/' ):
101                 new_flag = os.path.join( working_directory, flag )
102
103         for path_flag in path_flags:
104             if flag == path_flag:
105                 make_next_absolute = True
106                 break
107
108         if flag.startswith( path_flag ):
109             path = flag[ len( path_flag ): ]
110             new_flag = path_flag + os.path.join( working_directory, path )
111             break
112
113         if new_flag:
114             new_flags.append( new_flag )
115     return new_flags
116
117 def IsHeaderFile( filename ):
118     extension = os.path.splitext( filename )[ 1 ]
119     return extension in [ '.h', '.hxx', '.hpp', '.hh' ]
120
121 def GetCompilationInfoForFile( filename ):
122
123     # The compilation_commands.json file generated by CMake does not have entries
124     # for header files. So we do our best by asking the db for flags for a
125     # corresponding source file, if any. If one exists, the flags for that file
126     # should be good enough.
127     if IsHeaderFile( filename ):
128         basename = os.path.splitext( filename )[ 0 ]
129         for extension in SOURCE_EXTENSIONS:
130             for extension in SOURCE_EXTENSIONS:

```

```

133     replacement_file = basename + extension
134     if os.path.exists( replacement_file ):
135         compilation_info = database.GetCompilationInfoForFile(
136             replacement_file )
137         if compilation_info.compiler_flags_:
138             return compilation_info
139     return None
140     return database.GetCompilationInfoForFile( filename )
141
142 def FlagsForFile( filename, **kwargs ):
143     if database:
144
145         # Bear in mind that compilation_info.compiler_flags_ does NOT return a
146
147         # python list, but a "list-like" StringVec object
148         compilation_info = GetCompilationInfoForFile( filename )
149         if not compilation_info:
150             return None
151
152         final_flags = MakeRelativePathsInFlagsAbsolute(
153             compilation_info.compiler_flags_,
154             compilation_info.compiler_working_dir_ )
155
156
157         # NOTE: This is just for YouCompleteMe; it's highly likely that your project
158
159         # does NOT need to remove the stdlib flag. DO NOT USE THIS IN YOUR
160
161         # ycm_extra_conf IF YOU'RE NOT 100% SURE YOU NEED IT.
162
163         #try:
164
165         # final_flags.remove( '-stdlib=libc++' )
166
167         #except ValueError:
168
169         # pass
170         else:
171             relative_to = DirectoryOfThisScript()
172             final_flags = MakeRelativePathsInFlagsAbsolute( flags, relative_to )
173
174         return {
175             'flags': final_flags,
176             'do_cache': True
177         }

```

YouCompleteMe提供的其他功能

YCM除了提供了基本的补全功能,自动提示错误的功能外,还提供了类似tags的功能:

- 跳转到定义GoToDefinition
- 跳转到声明GoToDeclaration
- 以及两者的合体GoToDefinitionElseDeclaration

可以在.vimrc中配置相应的快捷键。

```

1 | noremap <leader>gl :YcmCompleter GoToDeclaration<CR>
2 | noremap <leader>gf :YcmCompleter GoToDefinition<CR>
3 | noremap <leader>gg :YcmCompleter GoToDefinitionElseDeclaration<CR>

```

另外, YCM也提供了丰富的配置选项, 同样在.vimrc中配置。具体请[参考](#)

```

1 | let g:ycm_error_symbol = '>>'
2 | let g:ycm_warning_symbol = '>*'

```

同时, YCM可以打开location-list来显示警告和错误的信息:YcmDiags。个人关于ycm的配置如下:

```

1 | " for ycm
2 | let g:ycm_error_symbol = '>>'
3 | let g:ycm_warning_symbol = '>*'
4 | noremap <leader>gl :YcmCompleter GoToDeclaration<CR>
5 | noremap <leader>gf :YcmCompleter GoToDefinition<CR>
6 | noremap <leader>gg :YcmCompleter GoToDefinitionElseDeclaration<CR>
7 | nmap <F4> :YcmDiags<CR>

```

YCM提供的跳跃功能采用了vim的jumplist, 往前跳和往后跳的快捷键为Ctrl+O以及Ctrl+I。

总结

YouCompleteMe是我用过的最爽的一个自动补全的插件了。之前使用acp时, 遇到大文件基本上就卡死了, 以至于都不怎么敢使用。由于YCM使用的时C/S结构, 部分使用vim脚本编写, 部分认为原生代码, 使得跑起来速度飞快。

抛弃Vim自带的坑爹的补全吧, 抛弃ctags吧, 抛弃cscope吧, YCM才是终极的补全神器。

在安装过程中,我也遇到了不少的坑。一会会发一篇解决这些坑的文章。

最后祝大家码年顺利,一码平川,码到功成。

👍 1 赞

🔖 6 收藏

💬 11 评论



相关文章

- [Vim 实用技术, 第 2 部分: 常用插件](#)
- [为什么 Vim 使用 HJKL 键作为方向键](#)
- [我的 Vim 常用插件和键位映射配置](#)
- [Vim 实用技术, 第 1 部分: 实用技巧](#)
- [七个高效的文本编辑习惯 \(以Vim为例\)](#)
- [Vim 实用技术, 第 3 部分: 定制 Vim](#)
- [Sublime Text使用体验](#)
- [25个Vim教程、视频和资源](#)
- [如何将Vim打造成一个成熟的IDE](#)
- [文本三巨头: zsh、tmux 和 vim](#)

可能感兴趣的话题

- [前端有多少细化的发展方向? · 23](#)
- [怎么样才算是一名程序员, 这位程序媛说我不是, 你说我是不是? · 38](#)
- [迅雷的口碑貌似越来越差了 · 9](#)
- [对未来的路感到迷茫、焦虑、希望能得到经历过的前人指点 · 2](#)
- [就想知道现在单身的程序员到底多不多? · 7](#)
- [消息队列一般用什么技术实现比较好? · 4](#)
- [驻留在磁盘上的类标记元组训练集D不能装进内存怎么解决?](#)
- [当你的工作好搭档要离职是一种怎样的体验? · 14](#)
- [2013 Google面试题 · 16](#)
- [怎么克服智齿带来的疼痛? · 6](#)

« [如果看了此文你还不理解傅里叶变换, 那就过来掐死我吧【完整版】](#)
[递归是如何进入编程的? »](#)

登录后评论

新用户注册

直接登录



最新评论



太漠

2014/02/11

不支持我大Java!!!!

👍 赞 回复 ↩



ic

2014/02/11

java 的要装eclim吧。。

👍 赞 回复 ↩



giraffe

04/24

Java直接用IDE好么?

👍 赞 回复 ↩

影行

2014/02/12