配置你自己独一无二的 vim

作者：李兴中　　单位：中国科学技术大学地空学院　时间：2017/12

题记：古语有云：“工欲善其事必先利其器”，要想获得更高的工作效率，我们必须得有趁手的工具。又所谓“高端唯有定制”，要想拥有自己专属的高大上的 vim，我们必须得学会自己配置。

一. vim 简介

vim 是 linux 系统上最著名的 文本/代码 编辑器，它是早年的 vi 编辑器的加强版。它最大的特色是脱离鼠标操作完全使用键盘命令进行编辑，键盘流的各种巧妙组合操作使得编辑效率大幅提升。此外，vim 的可配置性非常强，各种插件、语法高亮配色方案等多不胜数，我们可以通过 vim 配置文件和 vim 插件将其打造为适合我们每个人自己口味与需求的神器。

1.如何安装 vim：

在 ubuntu 平台下，在终端输入“sudo apt-get vim”即可安装。

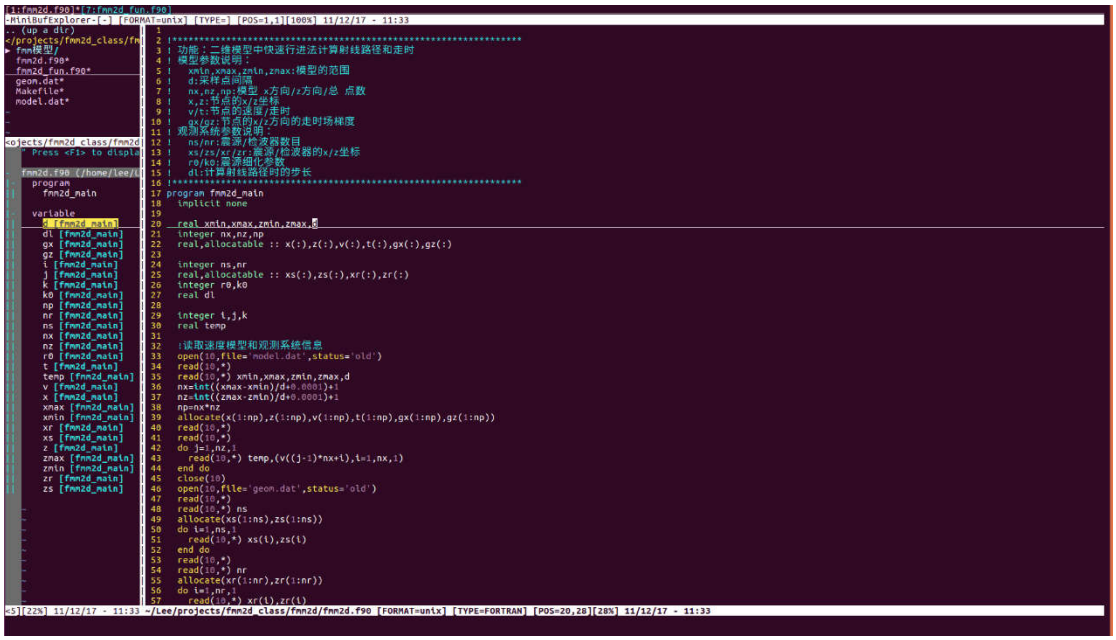
安装完成后可以在终端输入“vim”命令查看其版本号，现在最新的版本号是 7.4，为了保证配置成功，请确保你的 vim 编辑器是 7.4 或 7.4 以上版本。

2.vim 配置文件和 vim 插件：

Vim 强大的功能，其来源基本上就两个地方：vim 配置文件和 vim 插件。

Vim 本身的系统配置文件夹是在/usr/share/vim/和/etc/vim/两个文件夹下，一般情况下，我们不会去更改这两个文件夹下的配置文件，而是在用户文件夹/home/user(其中，user 为用户名)下建立自己的配置文件：“.vimrc”。然后开始编辑“.vimrc”文件中的内容。(注意：ubuntu 平台下以“.”开头的文件一般都是隐藏文件，按“ctrl”+“h”键便可以显示)

3.我的 vim 视图：



作者简介：中国科学技术大学固体地球物理专业硕士研究生在读，本科毕业于长安大学地球物理系。

二．基本设置

1.关闭兼容模式：

```
set nocompatible            关闭兼容模式
```

Vim 是 vi 的加强版，它在 vi 的基础上增加了很多功能，但就不与 vi 完全兼容了。“set compatible”就是让 vim 关闭所有扩展的功能，尽量模拟 vi 的行为，但这样就不应用 vim 的很多强大功能，所以一般如果没有什么特殊需要的话(比如执行很老的 vi 脚本)，都要在 vim 配置的开始，写上“set nocompatible”来关闭兼容模式。由于这个选项是最最基础的选项，会连带很多其它选项发生变动，所以它必须是第一个设定的选项。

2.鼠标设置：

```
set mouse=a                 启动对鼠标的支持
set selection=exclusive     允许区域选择
set selection=mouse,key     设置选择模式为 mouse 和 key
```

set mouse 选项说明：

虽然 vim 最大的特色是脱离鼠标操作完全使用键盘命令进行编辑，但是，不可否认的是，在许多情况下，使用鼠标会使得操作更为快捷和直观，所以，我们有必要在 vim 中启用对于鼠标的支持。

‘mouse’选项的字符决定 vim 会在什么模式下会使用鼠标：

n	普通模式(normal mode)
v	可视化模式(visual mode)
i	插入模式(insert mode)
c	命令行模式(command-line mode)
h	在帮助文件里，以上所有模式 (all previous modes when editing a help file)
a	以上所有模式(all previous modes)

通常情况下，我们会用“set mouse=a”来启动对鼠标的支持。

这样会带来一个新的问题：在 vim 中鼠标右键不能对选中的文本内容进行复制操作。其原因如下：鼠标事件有两种处理方式，程序处理和 X 处理。如果让 X 负责处理，则是左键选择，右键复制；要让 vim 中由 X 负责处理，按住 shift 键，然后选择，copy 选项就 enable 了，如果放掉 shift 键，则由 vim 处理该选择。

3.窗口参数设置：

```
winpos 5 5                  设置窗口位置
set lines=60 columns=200    设置窗口大小
```

winpos 选项说明：

vim 界面左上角的位置，原则上与界面左上角重合，更方便编辑。每个人的显示器尺寸不同，具体可以根据自己的显示器大小自行调整以达到满意的效果。

窗口大小说明:

设定的原则是让桌面全部显式为 vim 界面，更方便编辑。每个人的显示器尺寸不同，具体可以根据自己的显示器大小自行调整以达到满意的效果。

4.光标所在行设置：

```
set number                  显示行号
set ruler                   标尺功能，显式光标位置状态行
set statusline=.....        设置状态行显式的内容
```

set laststatus=2	设置总是显式状态行：(1：不显式；2：显式)
set cursorline	高亮显示光标所在行
set nowrap	设置不自动换行
set novisualbell	去掉输入错误时光标闪烁

5.命令行模式设置：

set cmdheight=2	设置命令行高度为 2(默认为 1)
set showcmd	显示输入的命令

6. tab/backspace 相关设置：

set noexpandtab	不将 tab 转化为 space
set shiftwidth=2	设置 tab 宽度为 2 个空格
set tabstop=2	将 tab 解释为 2 个空格
set softtabstop=2	2 个空白会被当成一个 tab 删除
set backspace=indent,eol,start	

tab 键设置说明：

shiftwidth/tabstop/softtabstop 一般设置为相同的值。

backspace 键设置说明：

indent:如果用了“set indent”等缩进，想用退格键将字段缩进的删掉，必须折这这个选项，否则不响应；eol:即“end of line”,如果插入模式下光标在下一行开头，想通过退格键合并两行，需要设置 eol；start:要想删除此次插入前的输入，需要设置 start。

7.自动缩进设置：

set autoindent	设置自动缩进(继承前一行的缩进方式)

8.文件相关设置：

set autowrite	设置自动保存
set autoread	文件被改动时自动载入
set nobackup	不备份文件
set noswapfile	不生成 swap 文件

9.折叠相关设置：

set foldenable	允许折叠
set foldmethod=syntax	用语法高亮来定义折叠
let fortran_fold=1	允许 fortran 代码折叠

foldmethod 选项说明：

vim 代码折叠方式可以用“foldmethod”选项来设置，共六种：

set foldmethod=manual	手工定义折叠
set foldmethod=indent	用缩进定义折叠
set foldmethod=expr	用表达式来定义折叠
set foldmethod=syntax	用语法高亮来定义折叠
set foldmethod=diff	对没有更改的文本进行折叠
set foldmethod=marker	用标志定义折叠

常用命令介绍：

za	打开/关闭在光标下的折叠
zA	循环地打开/关闭光标下地折叠
zo	打开在光标下地折叠
zO	循环地打开光标下的折叠

zc	关闭光标下的折叠
zC	循环地关闭光标下的折叠
zM	关闭所有折叠
zR	打开所有折叠

为了使用的方便，我们将 zo/zc 映射为快捷键：

```
noremap <F5> zo
noremap <F6> zc
```

10. 搜索相关设置：

set incsearch	增量式搜索(搜索时，输入语句逐字符高亮)
set hlsearch	高亮显示被搜索的内容
set ignorecase	搜索时忽略大小写

11. 匹配相关设置：

set showmatch	高亮显示匹配的括号
set matchtime=10	设置匹配括号高亮的时间为 10s

12. 编码/解码相关设置：

set enc=utf-8	转化为当前系统编码(utf-8)来显示
set fenc=utf-8	设置当前文件的编码为 utf-8
set fens=utf-8,gbk……	打开文件时进行解码的猜测列表

vim 中编码设置说明：

vim 里面的编码主要和三个参数有关:enc,fenc,fencs.

enc(encoding)：其作用基本只是显示，不管打开的文件是什么编码的，vim 都会将其转化为当前系统编码来处理，这样才能在当前系统里面正确地显示出来。

fenc(fileencoding)：指当前文件的编码，即当前文件要以什么样的编码形式存储。

fencs(fileencodings)：打开文件时进行解码的猜测列表。文件编码没有百分百正确的判断办法，所以 vim 只能猜测文件编码，不断尝试去进行解码。

13. 显示相关设置：

colorscheme ron	设置配色方案(不设置即为默认配色方案)
set scrolloff=3	设置光标移动到 buffer 顶部和底部时保持三行的距离

colorscheme 选项说明：

vim 的配色方案在/usr/share/vim/vim74/colors 文件夹里面，一系列以“.vim”结尾的文件，当然，你也可以自己编写或者去网上下载合适的自己喜欢的配色方案。

14. 语法高亮设置：

syntax on	打开语法高亮

syntax 选项说明：

vim 中打开语法高亮有两种方法：syntax enable 和 syntax on。后者会覆盖当前你对语法高亮的更改，由于更改高亮不常见，所以这两个命令的区别很小，建议使用后者。vim 一般都可以正确识别文本类型，并做出相应的高亮。

15. 文件类型检测设置：

filetype plugin indent on	开启文件类型检测/加载插件/缩进

filetype 选项说明：

filetype 的默认属性：detection:on plugin:off indent:off

detection:默认情况下 vim 会对文件自动检测文件类型。还有一种方式就是在文件内容中指定，vim 会从文件的头几行自动扫描文件是否有声明文件类型的代码，如在文件的首行加入//vim:filetype=html,当作注释写入，以至于不影响文件的编译，这样 vim 不通过文件名也能检测出文件是什么类型。

plugin:当 plugin 状态为 on 时，那么就会在 vim 的运行时环境目录下加载该类型相关的插件。

indent:不同类型文件有不同的缩进方式，比如 python 采用 4 个空格作为缩进，而 html 采用 2 个空格作为缩进，那么 indent 就可以为不同文件类型选择合适的缩进方式。

一般用 vim 都会打开这三者，因为这会极大地提高 vim 对不同类型文件编辑时的适应性和灵活性。

三. 自动补全设置

1.自动补全括号/引号：

在编写程序时，经常会用到成对的括号，引号等，不免会出现漏写右半部分从而导致错误，致使编译报错不通过的情况，所以我们可以通过映射来实现括号/引号的自动补全：当你键入左半部分“(”、“{”、“[”、““”、“‘”时，vim 会为你自动补全右半部分，并且将光标移到括号内，进入 insert 模式，等待用户输入。代码如下：

```
"自动补全括号，引号等
:inoremap ( ()<ESC>i
:inoremap { {}<ESC>i
:inoremap [ []<ESC>i
:inoremap " ""<ESC>i
:inoremap ' ''<ESC>i
```

2.supertab 插件：

(1)Overview:

supertab is a vim plugin which allow you to use <tab> for all your insert completion needs. (from developers)

(2)下载与安装：

下载地址 1：www.vim.org/scripts/script.php?script_id=1643

下载地址 2：https://github.com/ervandew/supertab

这两个网站上有资源与详细的安装步骤,其核心步骤是将“supertab.vim”文件放到“.vim/plugin”目录，此处不再赘述。

(3)在“.vimrc”中添加以下配置：

```
let g:SuperTabDefaultCompletionType=“<C-N>”
```

选项说明：设置按下 tab 键后默认的补全方式为<Ctrl+N>，更多补全方式用 :help ins-completion 查看；

```
let g:SuperTabRetainCompppletionType=2
```

选项说明：

0　　不记录上次的补全方式

1　　记住上次的补全方式，直到使用其他的补全命令改变它

2　　记住上次的补全方式，直到按 ESC 推出插入模式

```
set completeopt=longest,menu
```

选项说明：

只在下拉菜单中显示匹配项目。

该选项缺省时，vim 会使用下拉菜单和一个 preview 窗口来显示匹配项目，下拉菜单列出所有匹配的项目，preview 窗口显示对应匹配项目的详细信息。

(4)基本功能使用方法：

Supertab 使用起来很简单，只要在输入变量名或者路径名等符号中途按 tab 键，就能得到以前输入过的并且和当前字符匹配的符号下拉列表，并通过 tab 键循环选择，高亮的项表示当前选中项，按 ENTER 键确认选择；按 ESC 键退出下拉列表。

四. ctags

1.ctags 简介：

ctags(generate tag files for source code)是 vim 下方便代码阅读的工具，ctags 可以建立源码树的标签索引(标签就是一个标识符被定义的地方，如变量，函数等)，使程序员在编程时能迅速定位函数、变量、宏定义等。

2.ctags 安装：

ubuntu 下在终端中输入：sudo apt-get install ctags，即可安装。

3.如何使用：

(1)生成 tags 文件：

```
ctags -R
```

递归地为当前目录及其子目录下的所有代码文件生成 tags 文件

(2)指定 tags 文件：

```
set tags=./tags
```

指定 tags 文件为当前路径下的 tags 文件(在 vim 中打开源码时，必须先指定 tags 文件，然后才可以正常使用)

```
set autochdir
```

自动切换当前目录为当前文件所在目录

(3)如何跳转：

```
Ctrl+]
```

使光标跳转到函数、变量等被定义的地方

```
Ctrl+T
```

使光标跳转到函数、变量等被调用的地方

为了使用的方便，我们将(1)(3)中的命令映射为快捷键：

```
noremap <F2> :!ctags -R<CR>
noremap <F3> <C-]>
noremap <F4> <C-T>
```

五.taglist

1. taglist 简介：

taglist(a source code browser)是一款基于 ctags，在 vim 代码窗口旁以分割窗口形式显示当前的代码结构概览，增加代码浏览的便利程度的 vim 插件。Taglist 提供了源码的结构化浏览功能，可以将源码中定义的类、函数、变量等以树结构显示，使得层次结构一目了然，便于快速定位和查看。

2.taglist 安装：

下载地址：www.vim.org/scripts/script.php?script_id=273

你会得到一个压缩包：taglist_46.zip，解压后会得到两个文件夹：plugin/doc，分别存放了 taglist 脚本:taglist.vim 和帮助文档：taglist.txt，将它们分别 copy 到“./vim/plugin”，“./vim/doc”文档下。

需要注意的是：taglist 基于 ctags 才能发挥作用，因此在使用 taglist 之前请确保已经安装了 ctags。

3.taglist 相关配置：

let Tlist_Show_One_File=1

不同时显示多个文件的 tag，只显示当前文件的 tag

let Tlist_File_Fold_Auto_Close=1

当同时显示多个文件的 tag 时，使 taglist 只显示当前文件的 tag 其他文件的 tag 被折叠起来

let Tlist_Enable_Fold_Column=1

显示折叠树

let Tlist_Exit_OnlyWindow=0

taglist 为最后一个窗口时，退出 vim

let Tlist_Sort_Type=“name”

taglist 以 tag 名称进行排序，缺省时按 tag 在文件中出现的顺序排序

let Tlist_WinHeight=50

设置 taglist 窗口高度为 50

let Tlist_WinWidth=30

设置 taglist 窗口宽度为 30

六. quickfix

1.quickfix 简介：

quickfix 是 vim 默认的插件，无需安装即可使用。

2.为何要用 quickfix:

通常，我们在开发代码地过程中，经常要写代码、编译、修改编译错误，这个过程通常会重复很多遍。如果你仅仅根据编译器输出地错误信息，打开出错地文件，找到出错的行，然后再开始修改，那样效率未免也太低下了，所以人们开发出了 quickfix 作为 vim 的标准插件，来帮助我们加速这一过程。

3.quickfix 功能：

将编译过程中产生的错误信息保存到文件中，然后 vim 利用这些信息跳转到源文件的对应位置，我们就可以进行错误的修正，完成以后跳到下一个

错误的地方继续修正，从而极大地提高了编译错误地修正效率。

4.两个常用的切换命令：

```
:cn     change to next(切换到下一个结果)
:cp     change to previous(切换到上一个结果)
```

如果你经常使用这两个命令，你还可以给它们设定快捷键。

七. cscope

1. cscope 简介：

cscope 是一个类似于 ctags 的工具，不过其功能比 ctags 强大很多，用 cscope 自己的话说：“你可以把它当作超过频的 ctags”,其功能和强大程度可见一斑！

2.cscope 安装：

```
sudo apt-get install cscope
```

3.cscope 配置:

我主要用 cscope 来向 quickfix 窗口输出编译错误的位置信息，在./vimrc 文件中加上：“`set cscopequickfix=s-,c-,d-,i-,t-,e-`”。cscope 的其他功能可以通过帮助手册自己慢慢学习。

八. NERDtree

1. NERDtree 简介：

NERDtree : a tree explorer plugin for navigating the filesystem.

The NERDtree allows you to explore your filesystem and to open files and directories. It presents the filesystem to you in the form of a tree which you manipulate with the keyboard and/or mouse. It also allows you to perform simple filesystem operations.(from developers)

NERDtree 的作用就是列出当前路径的目录树，让开发者能方便地浏览项目的总体目录结构，还可以创建删除重命名文件等。

2. NERDtree 安装：

下载地址：www.vim.org/scripts/script.php?script_id=1658

将下载得到的压缩包(NERD_tree.zip)解压到~/.vim 文件夹里面,并且将plugin/NERD_tree.vim和doc/NERD_tree.txt分别copy到~/.vim/plugin和~/.vim/doc 目录下。

九. minibufexpl

1.minibufexpl 简介：

minibufexpl: elegant buffer explorer - takes very little screen space.(from develpoers)

在编程的时候不可能永远只编辑一个文件，你肯定会打开很多源文件进行编辑，如果每个文件都打开一个 vim 进行编辑的话那操作起来会很麻烦，所以 vim 有了缓冲区(buffer)的概念，vim 自带的 buffer 管理工具只

有:ls,:bnext,:bdelet 等命令，既不好用，又不直观，所以有大牛们就开发出了 minibufexpl 这款 vim 插件。

2.minibufexpl 安装：

下载地址：www.vim.org/scripts/script.php?script_id=159

将下载得到的 minibufexpl.vim 放在~/.vim/plugin 文件夹下即可。

3.minibufexpl 配置：

let g: miniBufExplMapWindowNavVim=1

可以用<Ctrl>+hjkl 切换到上下左右的不同窗口

let g: miniBufExplMapWindowNavArrows=1

可以用<Ctrl>+箭头切换到上下左右的不同窗口

let g: miniBufExplMapCTabswitchBufs=1

<Ctrl>+<Tab>向前循环切换到每个 buffer 名上

<Ctrl>+<Shift>+<Tab>向后循环切换到每个 buffer 名上

let g: miniBufExplModSelTarget=1

设置只在源码窗口打开选中的 buffer(默认是关闭的)

十. winmanager

1.winmanager 简介：

wimanager : a windows style IDE for vim(from developers)

winmanager 是一款 vim 插件，可以实现将 vim 整合成传统的 IDE 窗口界面，winmanger 主要用来整合 NERDtree 和 taglist，所以，首先确保成功安装了 NERDtree 和 taglist.

2.winmanager 安装：

下载地址：www.vim.org/scripts/script.php?script_id=95

下载压缩包 winmanager.zip,解压后将/plugin 和/doc 目录下的文件分别 copy 到~/.vim/plugin 和~/.vim/doc 文件夹中去即可。

3.winmanager 配置：

let g:NERDtree_title=“[NERDtree]”

let g:winManagerWindowLayout=“NERDtree|Taglist”

设置界面分割

let g:WinManagerWidth=30

设置 winmanager 的宽度为 30

noremap <F10> :WMToggle<CR>

设置<F10>为打开/关闭 winmanager 的快捷键

还需要配置两个额外的函数：

```
function! NERDTree_Start()
  exec 'NERDTree'
endfunction

function! NERDTree_IsValid()
  return 1
endfunction
```

注意：这个版本的 winmanager 有个小 bug，你在打开 winmanager 界面时，会同时打开一个空的文件，这会影响我们的使用，所以我我们要在打开

winmanager 时关闭这个文件，解决方法如下：

在~/.vim/plugin 目录下的 winmanager.vim 文件中找到以下函数定义，并且在第 5 行下添加第六行的内容：

```
function! <SID>ToggleWindowsManager()
    if IsWinManagerVisible()
        call s:CloseWindowsManager()
    else
        call s:StartWindowsManager()
        exe 'q'
    end
endfunction
```

十一.make/quit 快捷键设置

noremap <F9>:make<CR>

设置<F9>为 make 快捷键(需要配合 Makefile 文件才能使用)

noremap <F12>:xa<CR>

设置<F12>为保存并退出 vim 快捷键

十二.Map 简介

文中我们经常将一些命令组合映射为快捷键以方便我们操作，那么映射(map)究竟是怎么工作的呢？同 vim 中的其他命令一样，命令的名字往往有好几段组成，前缀作为命令本身的修饰符，起到微调命令的效果，对于 map 而言，主要有这么几种前缀：

nore(no recursive)	表示非递归
n(normal)	表示在普通模式下生效
v(visual)	表示在可视化模式下生效
i(insert)	表示在插入模式下生效
c(command-line)	表示在命令行模式下生效

后记：

为了方便大家使用，本人已经将自己的.vimrc 文件分享在 github 上(https://github.com/xzli8/vim-configuration).

本文参考了网络上许多博主的博文，感谢这些乐于分享的人们！总结经验并分享，方便大家，共同进步！如发现本文内容的谬误或者需要补充的地方，请联系作者(email:xzli8@mail.ustc.edu.cn)。