

(vim) 功能调节

手稿



封面

vim 在启动时自动加载配置文件中的设置，支持以插件模式调节自身功能。

概述

在 vim 的使用过程中，你可以根据需要调节它的功能，使之更加贴合自己的工作风格。

如果要调节 vim 的功能，最常用的方式是在命令模式下输入 `:set` 命令来设置各种参数开关。除此以外，你还可以将一些常规选项放在 vim 的配置文件，这样的话，每次 vim 启动时都会自动加载这些配置。

用 ..set 设置参数开关

vim 中包含大量内部变量和开关，通过修改这些设置，你可以获得一些特殊效果。这些选项可分为 3 大类：

- ✦ 开关 只能是开或关
- ✦ 数值 可以是数字值
- ✦ 字符串 可以是字符串值

以搜索为例，如果想要获得增量搜索的效果，你可以打开 `incsearch` 开关：

```
:set incsearch
```

当打开 `incsearch` 开关后，匹配项会随着用户输入实时变化。

```
102 linear() conforms to the folloing formula:
103     y = f(x) = a + bx
104 there are three parameters in function line
105     corresponding to the above formula.
106
107 Among these three parameters, a is the cons
108     b is the coefficient of the independent va
```

```
102 linear() conforms to the folloing formula:
103     y = f(x) = a + bx
104 there are three parameters in function line
105     corresponding to the above formula.
106
107 Among these three parameters, a is the cons
108     b is the coefficient of the independent va
/the
```

```
102 linear() conforms to the folloing formula:
103     y = f(x) = a + bx
104 there are three parameters in function line
105     corresponding to the above formula.
106
107 Among these three parameters, a is the cons
108     b is the coefficient of the independent va
/these
```

配置文件

对于一些常用的配置选项，它们可以放入配置文件。当 `vim` 启动时，这些配置被自动加载。`vim` 会从以下几个地方寻找配置文件（用 `:version` 命令查看）：

- ✦ 系统 `vimrc` 文件："`$VIM/vimrc`"
- ✦ 用户 `vimrc` 文件："`$HOME/.vimrc`"
- ✦ 第二用户 `vimrc` 文件："`~/vim/vimrc`"
- ✦ 用户 `exrc` 文件："`$HOME/.exrc`"
- ✦ `$VIM` 后备选项："`/usr/share/vim`"

你可以选择合适的位置来存放自己的 `vimrc` 配置文件。

配置文件示例

在前面介绍 `vim` 过程中，我们用到了许多参数选项，现在可以将它们放到配置文件中了，也免去了每次输入命令的麻烦。

假设我们选择 `~/.vimrc` 作为自己的配置文件，那么里面的内容看起来可能是这样（" 开头的是注释行）：

```
$ cat ~/.vimrc

" my favorite options
set nu
set autoindent
set cindent
set ts=4
set sw=4
set showmatch
set ruler
set hlsearch
```

插件

插件是调节 `vim` 功能的另一种重要方式。插件的本质是 `vim` 脚本文件，当被放到 `plugin` 目录（比如 `~/.vim/plugin`）时，它们在 `vim` 启动时被自动加载。

插件分为两种，分别是：

- ◆ 全局插件 适用于所有文件
- ◆ 特定文件插件 仅用于特定文件类型

注：在没有特别说明的情况下，本系列均在没有安装任何插件的条件下进行演示。

Tag List 插件

Tag List 是一款非常流行的源代码浏览插件，它为源代码文件提供结构总览，让你能够在不同语言的源代码文件中高效的浏览代码。

你可以从 <http://www.vim.org> 下载最新的 Tag List 压缩包，然后将里面的文件释放到 vim 目录，比如 `~/.vim/`。这步之后，vim 目录下会有以下 2 个文件：

```
plugin/taglist.vim - 插件文件
doc/taglist.txt    - 帮助文档
```

接下来，进入 `~/.vim/doc`，执行命令 `“:helptags .”`；如果没有这一步，你无法跳转到 taglist 帮助主题。

Tag List 依赖于 ctags -- 为源代码文件找到的语言对象生成索引（或标签），从而让这些条目能够被文本编辑器或工具快速找到并轻松定位。

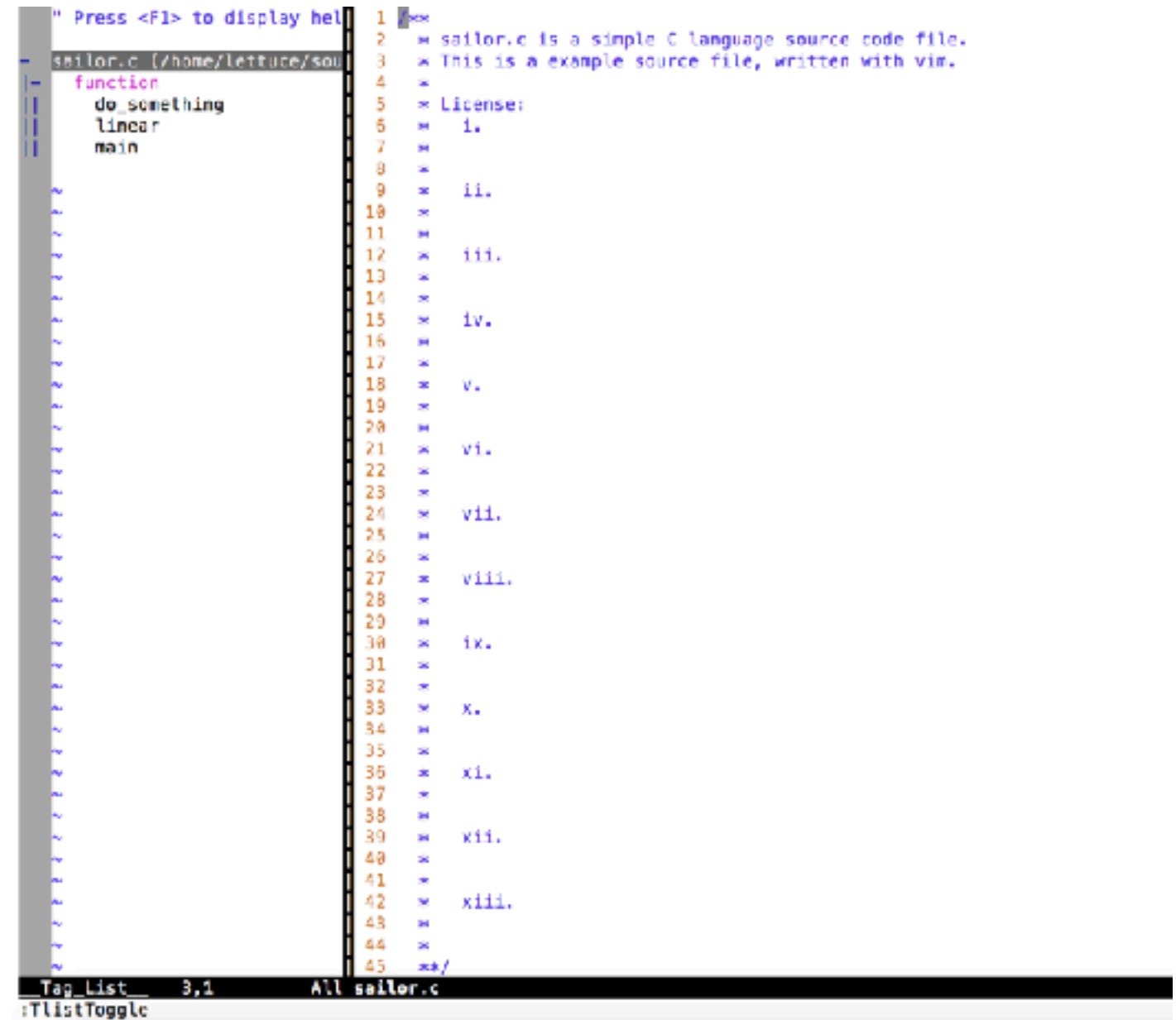
如果你使用的是 Unix 或类似平台，可以选择从源代码编译得到 ctags 二进制文件并安装。首先，从 <http://ctags.sourceforge.net/> 下载最新的源代码压缩包，然后进入源代码目录，依次执行：

```
$ ./configure
$ make
$ make check (可选)
$ sudo make install
```

taglist 窗口

在 Tag List 插件安装完成后，用 vim 重新打开源代码文件，在命令模式下输入 `:TlistToggle`，这时 vim 左侧会显示 taglist 窗口。

如果你使用的不是 GUI 窗口，比如 xterm，那么可以用命令 `:TlistOpen` 打开 taglist 窗口并获得光标，用命令 `:TlistClose` 关闭 taglist 窗口；或者使用 `C-w` `C-w`（C 表示 Control 键）在源代码窗口和 taglist 窗口间切换。



The screenshot shows a vim editor interface. On the left, the taglist window is open, displaying a list of tags from the file 'sailor.c'. The tags listed are 'function', 'do_something', 'linear', and 'main'. On the right, the main editor window shows the source code of 'sailor.c'. The code includes a header comment, a license notice, and a list of Roman numerals from i to xiii. The status bar at the bottom indicates the current file is 'sailor.c' and the command ':TlistToggle' is active.

```
" Press <F1> to display help
1  *
2  * sailor.c is a simple C language source code file.
3  * This is a example source file, written with vim.
4  *
5  * License:
6  * i.
7  *
8  * ii.
9  *
10 *
11 *
12 * iii.
13 *
14 *
15 * iv.
16 *
17 *
18 * v.
19 *
20 *
21 * vi.
22 *
23 *
24 * vii.
25 *
26 *
27 * viii.
28 *
29 *
30 * ix.
31 *
32 *
33 * x.
34 *
35 *
36 * xi.
37 *
38 *
39 * xii.
40 *
41 *
42 * xiii.
43 *
44 *
45 **/

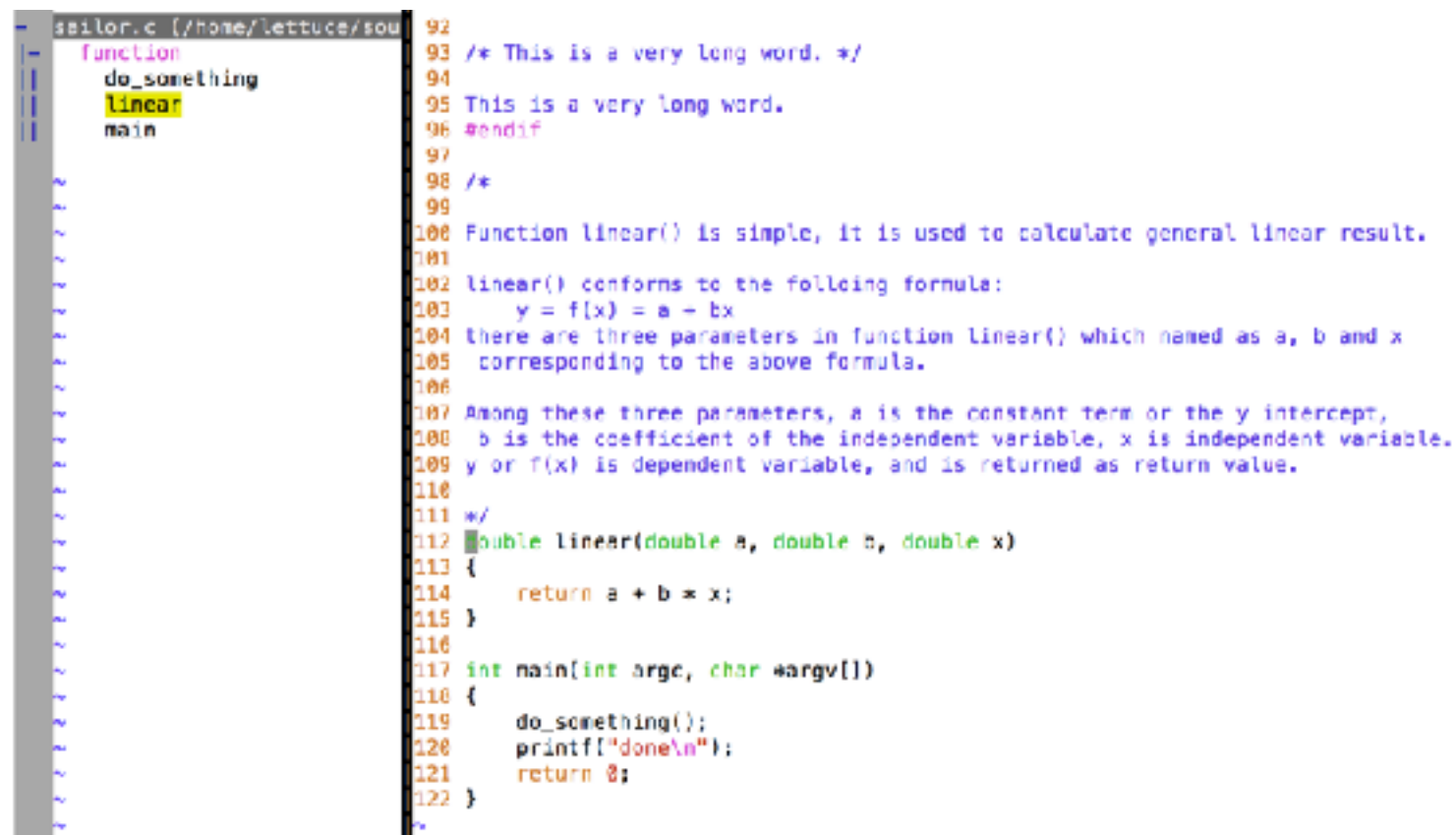
function
do_something
linear
main

Tag_List_ 3,1 All sailor.c
:TlistToggle
```

标签跳转

当位于 `taglist` 窗口时，将光标移至某个标签，比如 `linear`，按下回车，则光标切换至源代码窗口 `linear` 函数的第一行。

更多信息请参阅 <http://vim-taglist.sourceforge.net/>。



```
ssilor.c (/home/lettuce/sou 92
| 93 /* This is a very long word. */
| 94
| 95 This is a very long word.
| 96 #endif
| 97
| 98 /*
| 99
| 100 Function linear() is simple, it is used to calculate general linear result.
| 101
| 102 linear() conforms to the follling formula:
| 103     y = f(x) = a + bx
| 104 there are three parameters in function linear() which named as a, b and x
| 105     corresponding to the above formula.
| 106
| 107 Among these three parameters, a is the constant term or the y intercept,
| 108 b is the coefficient of the independent variable, x is independent variable.
| 109 y or f(x) is dependent variable, and is returned as return value.
| 110
| 111 */
| 112 double linear(double a, double b, double x)
| 113 {
| 114     return a + b * x;
| 115 }
| 116
| 117 int main(int argc, char *argv[])
| 118 {
| 119     do_something();
| 120     printf("done\n");
| 121     return 0;
| 122 }
| 123
| 124 */
```

简单映射

最后，我们介绍一下如何将多个命令绑定到一个按键，即映射。定义映射的一般格式为：

```
:map <key> commands
```

举例来说，如果想要用按键 F8 对一行使用 C 样式注释（/* */），可以这样定义：

```
:map <F8> I/* <Esc>A */<Esc>
```

其中：

F8 代表键盘上的 F8 键，这里是逐字写下 F 和 8，而不是按下 F8

I 表示在行首插入字符

<Esc> 表示 esc 键，同样是逐字写下 E、s 和 c

A 表示在行尾添加字符

注意 1：

绑定的按键不能与已有命令使用的按键重复。

注意 2：

命令模式下定义的映射在 vim 退出后失效，也可以用 :unmap 命令（这里是 :unmap <F8>）显式取消；为避免每次都手动输入，可以将 :map 命令放在配置文件中。

```
Comment this line.
```

F8



```
/* Comment this line. */
```