17.vim编辑器

什么是 vim？

vim是从 vi 发展出来的一个文本编辑器。代码补完、编译及错误跳转等方便编程的功能特别丰富，在程序员中被广泛使用。

简单的来说， vi 是老式的字处理器，不过功能已经很齐全了，但是还是有可以进步的地方。

vim 则可以说是程序开发者的一项很好用的工具。**最显眼的区别，vi没有彩色，vim的代码有颜色区别**

连 vim 的官方网站 (http://www.vim.org) 自己也说 vim 是一个程序开发工具而不是文字处理软件。

CentOS7.6中最小化模式安装是没有默认安装vim的，所以，需要安装一下：yum install -y vim

vi/vim 的使用

基本上 vi/vim 共分为三种模式，分别是**命令模式（Command mode），输入模式（Insert mode）和底线命令模式（Last line mode）。**

这三种模式的作用分别是：

命令模式：

用户刚刚启动 vi/vim，便进入了**命令模式**。

此状态下敲击键盘动作会被Vim识别为命令，而非输入字符。比如我们此时按下i，并不会输入一个字符，i被当作了一个命令。

以下是常用的几个命令：

- i 切换到输入模式，以输入字符。
- x 删除当前光标所在处的字符。
- : 切换到底线命令模式，用以在最底下一行输入命令。

若想要编辑文本：启动Vim，进入了命令模式，按下i，切换到输入模式。

命令模式只有一些最基本的命令，因此仍要依靠底线命令模式输入更多命令。

输入模式

在命令模式下按下i就进入了输入模式。

在输入模式中，可以使用以下按键：

- **字符按键以及Shift组合**，输入字符
- ENTER，回车键，换行
- BACK SPACE，退格键，删除光标前一个字符

- DEL，删除键，删除光标后一个字符
- **方向键**，在文本中移动光标
- HOME/END，移动光标到行首/行尾
- Page Up/Page Down，上/下翻页
- Insert，切换光标为输入/替换模式，光标将变成竖线/下划线
- ESC，退出输入模式，切换到命令模式

底线命令模式

在命令模式下按下:（英文冒号）就进入了底线命令模式。

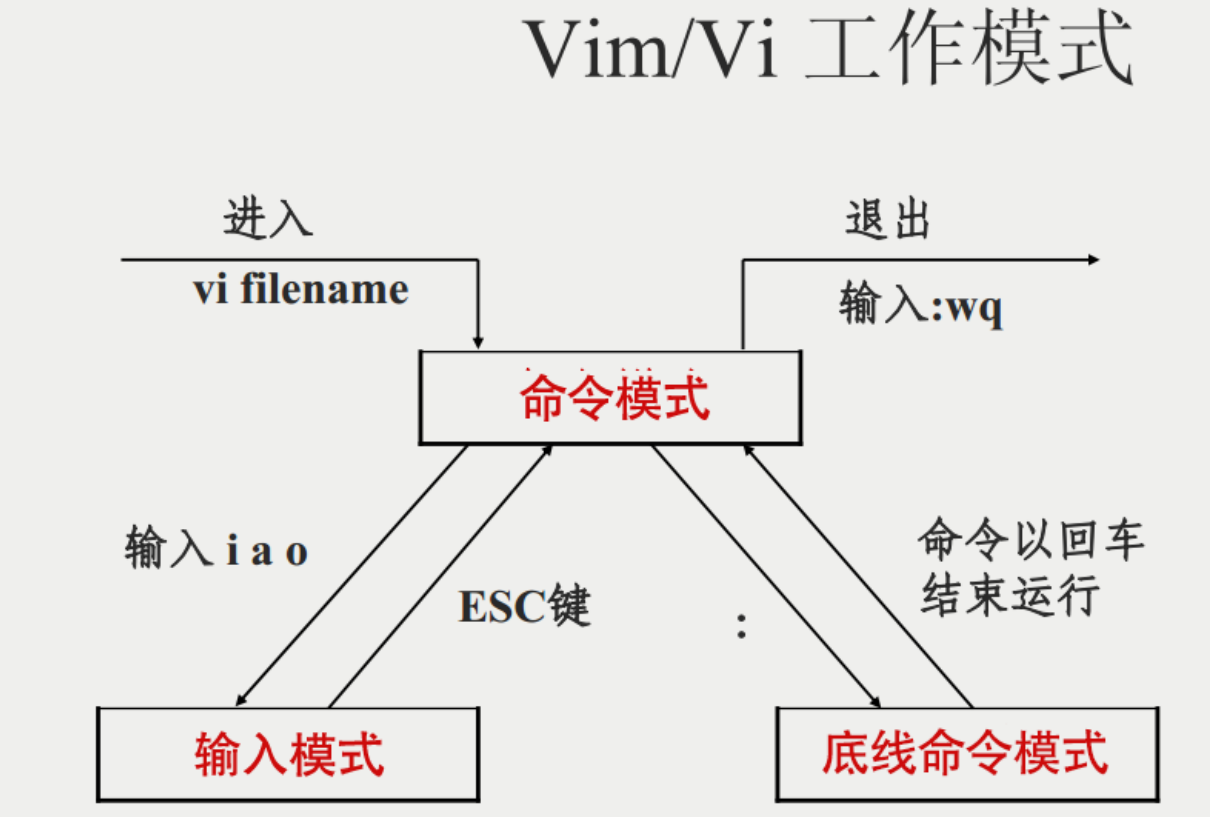
底线命令模式可以输入单个或多个字符的命令，可用的命令非常多。

在底线命令模式中，基本的命令有（已经省略了冒号）：

- q 退出程序
- w 保存文件

按ESC键可随时退出底线命令模式。

简单的说，我们可以将这三个模式想成下面的这个示意图：



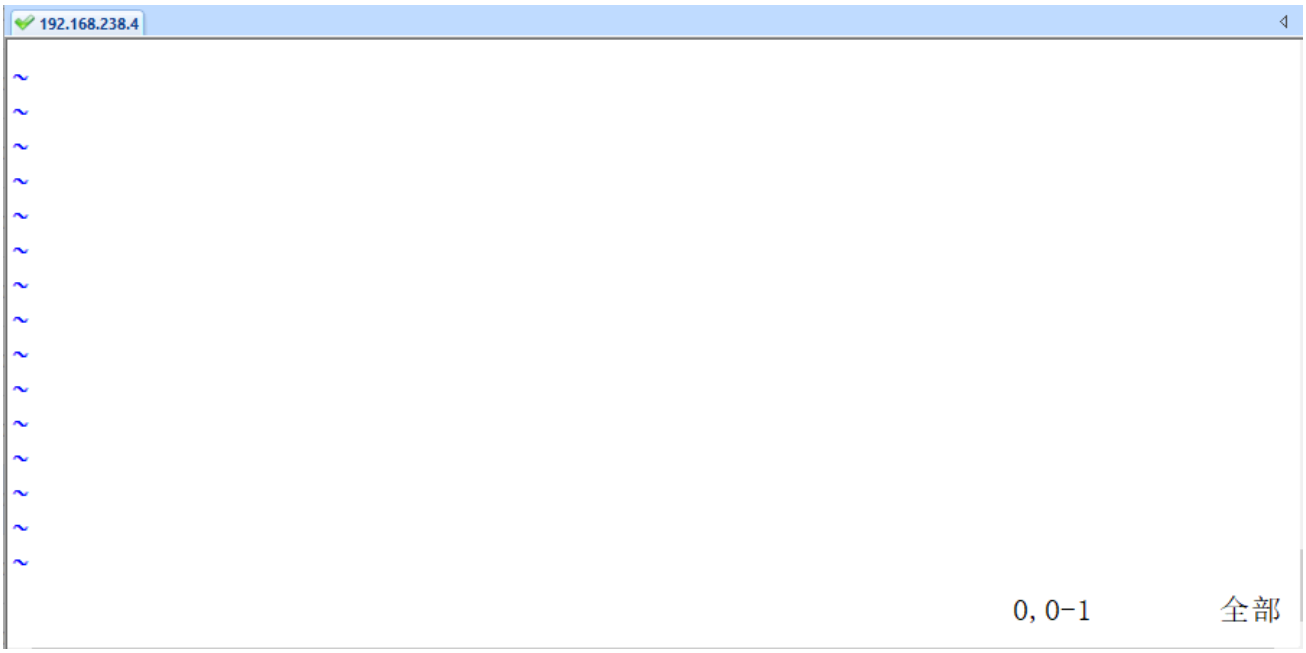
vi/vim 使用实例

使用 vi/vim 进入一般模式

如果你想要使用 vi 来建立一个名为 aa.txt 的文件时，你可以这样做：

```
$ vi aa.txt
```

直接输入 vi 文件名 就能够进入 vi 的命令模式了。**请注意，记得 vi 后面一定要加文件名，不管该文件存在与否！如果不存在，会自动新建文件！**

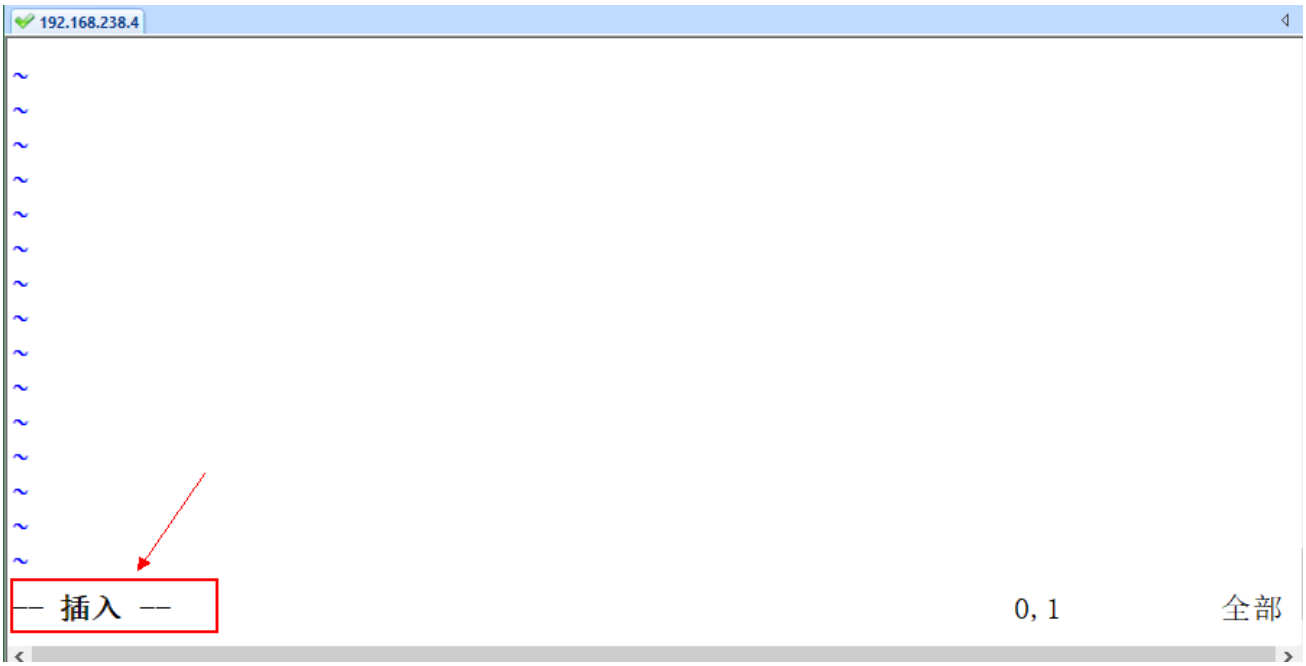


按下 i 进入输入模式(也称为编辑模式)，开始编辑文字

在一般模式之中，只要按下 i，o，a 等字符就可以进入输入模式了！

在编辑模式当中，你可以发现在左下角状态栏中会出现 –INSERT– 的字样，那就是可以输入任意字符的提示。

这个时候，键盘上除了 Esc 这个按键之外，其他的按键都可以视作为一般的输入按钮了，所以你可以进行任何的编辑。

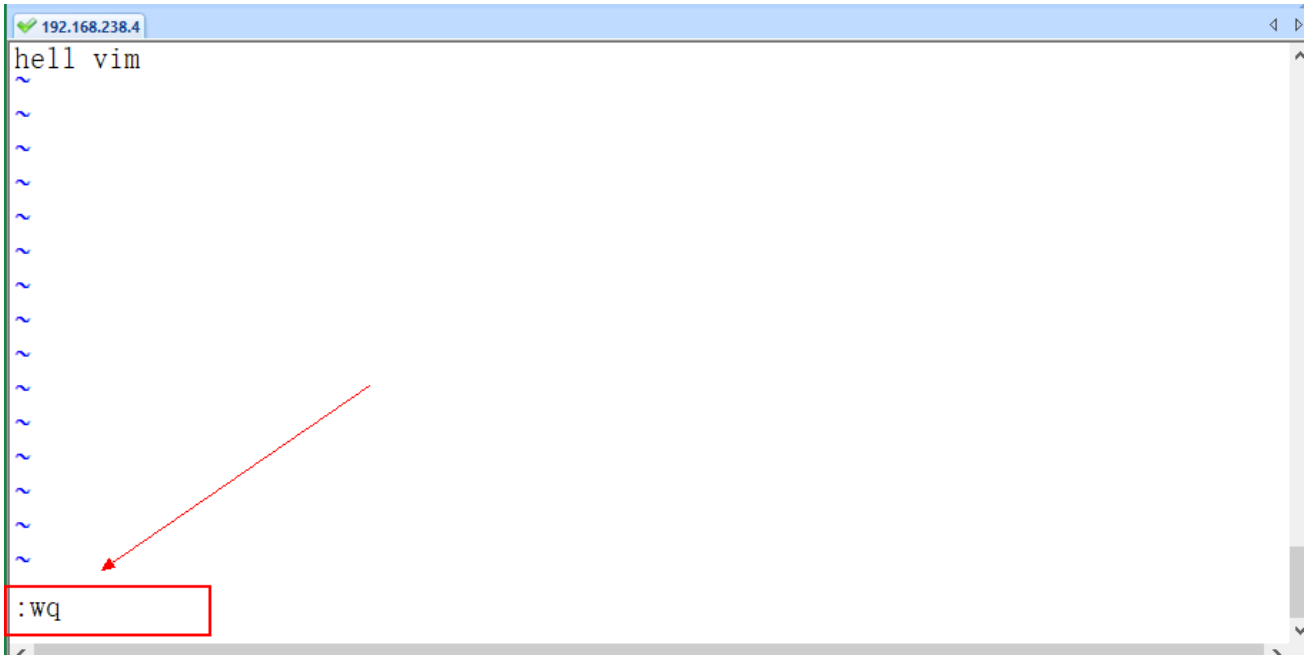


按下 ESC 按钮回到一般模式

好了，假设我已经按照上面的样式给他编辑完毕了，那么应该要如何退出呢？是的！没错！就是给他按下 Esc 这个按钮即可！马上你就会发现画面左下角的 – INSERT – 不见了！

在一般模式中按下 :wq 储存后离开 vi

OK，我们要存档了，存盘并离开的指令很简单，输入 :wq 即可保存离开！



OK！这样我们就成功创建了一个 aa.txt 的文件。

除了上面简易范例的 i，Esc，:wq 之外，其实 vim 还有非常多的按键可以使用。

第一部份：一般模式可用的光标移动、复制粘贴、搜索替换等

移动光标的方法	
h 或 向左箭头键(←)	光标向左移动一个字符
j 或 向下箭头键(↓)	光标向下移动一个字符
k 或 向上箭头键(↑)	光标向上移动一个字符
l 或 向右箭头键(→)	光标向右移动一个字符
如果你将右手放在键盘上的话，你会发现 hjkl 是排列在一起的，因此可以使用这四个按钮来移动光标。 如果想要进行多次移动的话，例如向下移动 30 行，可以使用 "30j" 或 "30↓" 的组合按键， 亦即加上想要进行的次数(数字)后，按下动作即可！	
[Ctrl] + [f]	屏幕『向下』移动一页，相当于 [Page Down]按键 (常用)
[Ctrl] + [b]	屏幕『向上』移动一页，相当于 [Page Up] 按键 (常用)
[Ctrl] + [d]	屏幕『向下』移动半页
[Ctrl] + [u]	屏幕『向上』移动半页
+	光标移动到非空格符的下一行
-	光标移动到非空格符的上一行
n<space>	那个 n 表示『数字』，例如 20 。按下数字后再按空格键，光标会向右移动这一行的 n 个字符。例如 20<space> 则光标会向后面移动 20 个字符距离。
0 或功能键[Home]	这是数字『 0 』：移动到这一行的最前面字符处 (常用)
$ 或功能键[End]	移动到这一行的最后面字符处(常用)
H	光标移动到这个屏幕的最上方那一行的第一个字符

H	光标移动到这个屏幕的最上方那一行的第一个字符
M	光标移动到这个屏幕的中央那一行的第一个字符
L	光标移动到这个屏幕的最下方那一行的第一个字符
G	移动到这个档案的最后一行(常用)
nG	n 为数字。移动到这个档案的第 n 行。例如 20G 则会移动到这个档案的第 20 行(可配合 :set nu)
gg	移动到这个档案的第一行，相当于 1G 啊！ (常用)
n<Enter>	n 为数字。光标向下移动 n 行(常用)
搜索替换	
/word	向光标之下寻找一个名称为 word 的字符串。例如要在档案内搜寻 vbird 这个字符串，就输入 /vbird 即可！ (常用)
?word	向光标之上寻找一个字符串名称为 word 的字符串。
n	这个 n 是英文按键。代表重复前一个搜寻的动作。举例来说， 如果刚刚我们执行 /vbird 去向下搜寻 vbird 这个字符串，则按下 n 后，会向下继续搜寻下一个名称为 vbird 的字符串。如果是执行 ?vbird 的话，那么按下 n 则会向上继续搜寻名称为 vbird 的字符串！
N	这个 N 是英文按键。与 n 刚好相反，为『反向』进行前一个搜寻动作。 例如 /vbird 后，按下 N 则表示『向上』搜寻 vbird 。

:n1,n2s/word1/word2/g	n1 与 n2 为数字。在第 n1 与 n2 行之间寻找 word1 这个字符串，并将该字符串取代为 word2！举例来说，在 100 到 200 行之间搜寻 vbird 并取代为 VBIRD 则：『:100,200s/vbird/VBIRD/g』。(常用)
`:1,$s/word1/word2/g` 或 `:%s/word1/word2/g`	从第一行到最后一行寻找 word1 字符串，并将该字符串取代为 word2！ (常用)
`:1,$s/word1/word2/gc` 或 `:%s/word1/word2/gc`	从第一行到最后一行寻找 word1 字符串，并将该字符串取代为 word2！且在取代前显示提示字符给用户确认 (confirm) 是否需要取代！(常用)

删除、复制与贴上	
x, X	在一行字当中，x 为向后删除一个字符 (相当于 [del] 按键)， X 为向前删除一个字符(相当于 [backspace] 亦即是退格键) (常用)
nx	n 为数字，连续向后删除 n 个字符。举例来说，我要连续删除 10 个字符， 『10x』。
dd	删除游标所在的那一整行(常用)
ndd	n 为数字。删除光标所在的向下 n 行，例如 20dd 则是删除 20 行 (常用)
d1G	删除光标所在到第一行的所有数据
dG	删除光标所在到最后一行的所有数据
d$	删除游标所在处，到该行的最后一个字符
d0	那个是数字的 0 ，删除游标所在处，到该行的最前面一个字符
yy	复制游标所在的那一行(常用)
nyy	n 为数字。复制光标所在的向下 n 行，例如 20yy 则是复制 20 行(常用)
y1G	复制游标所在行到第一行的所有数据
yG	复制游标所在行到最后一行的所有数据
y0	复制光标所在的那个字符到该行行首的所有数据
y$	复制光标所在的那个字符到该行行尾的所有数据
p, P	p 为将已复制的数据在光标下一行贴上，P 则为贴在游标上一行！ 举例来说，我目前光标在第 20 行，且已经复制了 10 行数据。则按下 p 后， 那 10 行数据会贴在原本的 20 行之后，亦即由 21 行开始贴。但如果是按下 P 呢？ 那么原本的第 20 行会被推到变成 30 行。 (常用)
J	将光标所在行与下一行的数据结合成同一行
c	重复删除多个数据，例如向下删除 10 行，[10cj]
u	复原前一个动作。(常用)
[Ctrl]+r	重做上一个动作。(常用)
这个 u 与 [Ctrl]+r 是很常用的指令！一个是复原，另一个则是重做一次～ 利用这两个功能按键，你的编辑，嘿嘿！很快乐的啦！	
.	不要怀疑！这就是小数点！意思是重复前一个动作的意思。 如果你想要重复删除、重复贴上等等动作，按下小数点『.』就好了！ (常用)

第二部份：一般模式切换到编辑模式的可用的按钮说明

进入输入或取代的编辑模式	
i, I	进入输入模式(Insert mode)： i 为『从目前光标所在处输入』， I 为『在目前所在行的第一个非空格符处开始输入』。 (常用)
a, A	进入输入模式(Insert mode)： a 为『从目前光标所在的下一个字符处开始输入』， A 为『从光标所在行的最后一个字符处开始输入』。(常用)
o, O	进入输入模式(Insert mode)： 这是英文字母 o 的大小写。o 为『在目前光标所在的下一行处输入新的一行』； O 为在目前光标所在处的上一行输入新的一行！(常用)
r, R	进入取代模式(Replace mode)： r 只会取代光标所在的那一个字符一次；R会一直取代光标所在的文字，直到按下 ESC 为止；(常用)
上面这些按键中，在 vi 画面的左下角处会出现『--INSERT--』或『--REPLACE--』的字样。 由名称就知道该动作了吧！！特别注意的是，我们上面也提过了，你想要在档案里面输入字符时， 一定要在左下角处看到 INSERT 或 REPLACE 才能输入喔!	
[Esc]	退出编辑模式，回到一般模式中(常用)

第三部份：一般模式切换到指令行模式的可用的按钮说明

指令行的储存、离开等指令	
:w	将编辑的数据写入硬盘档案中(常用)
:w!	若文件属性为『只读』时，强制写入该档案。不过，到底能不能写入， 还是跟你对该档案的档案权限有关啊!
:q	离开 vi (常用)
:q!	若曾修改过档案，又不想储存，使用 ! 为强制离开不储存档案。
注意一下啊，那个惊叹号 (!) 在 vi 当中，常常具有『强制』的意思～	
:wq	储存后离开，若为 :wq! 则为强制储存后离开 (常用)
ZZ	这是大写的 Z 喔! 若档案没有更动，则不储存离开，若档案已经被更动过，则储存后离开!
:w [filename]	将编辑的数据储存成另一个档案 (类似另存新档)
:r [filename]	在编辑的数据中，读入另一个档案的数据。亦即将 『filename』 这个档案内容加到游标所在行后面
:n1,n2 w [filename]	将 n1 到 n2 的内容储存成 filename 这个档案。
:! command	暂时离开 vi 到指令行模式下执行 command 的显示结果! 例如 『:! ls /home』即可在 vi 当中察看 /home 底下以 ls 输出的档案信息!
vim 环境的变更	
:set nu	显示行号，设定之后，会在每一行的前缀显示该行的行号
:set nonu	与 set nu 相反，为取消行号!

特别注意，在 vi/vim 中，数字是很有意义的! 数字通常代表重复做几次的意思! 也有可能是代表去到第几个什么什么的意思。

举例来说，要删除 50 行，则是用 『50dd』 对吧! 数字加在动作之前，如我要向下移动 20 行呢? 那就是『20j』或者是『20↓』即可。

环境变量很多，直接设置只是临时生效，关闭重新打开就不起作用了，要永远起作用可以修改文件：~/.vimrc

vim工具开头的时候就说了，它是为编程用的工具并不是一个文字处理工具，所以远远不止上面的基本功能，这个给大家提供一个网上的配置文件，大家可以用用：

```
map <F9> :call SaveInputData()<CR>
func! SaveInputData()
 exec "tabnew"
 exec 'normal "+gP'
 exec "w! /tmp/input_data"
endfunc

"colorscheme torte
"colorscheme murphy
"colorscheme desert
"colorscheme desert
"colorscheme elflord
```

```
colorscheme ron




"set fencs=utf-8,ucs-bom,shift-jis,gb18030,gbk,gb2312,cp936
"set termencoding=utf-8
"set encoding=utf-8
"set fileencodings=ucs-bom,utf-8,cp936
"set fileencoding=utf-8

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
" 显示相关
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
"set shortmess=atI   " 启动的时候不显示那个援助乌干达儿童的提示
"winpos 5 5          " 设定窗口位置
"set lines=40 columns=155    " 设定窗口大小
set go=             " 不要图形按钮
"color asmanian2     " 设置背景主题
"set guifont=Courier_New:h10:cANSI   " 设置字体
"syntax on           " 语法高亮
autocmd InsertLeave * se nocul  " 用浅色高亮当前行
autocmd InsertEnter * se cul    " 用浅色高亮当前行
"set ruler           " 显示标尺
set showcmd         " 输入的命令显示出来，看的清楚些
"set cmdheight=1     " 命令行（在状态行下）的高度，设置为1
"set whichwrap+=<,>,h,l   " 允许backspace和光标键跨越行边界(不建议)
"set scrolloff=3     " 光标移动到buffer的顶部和底部时保持3行距离
set novisualbell    " 不要闪烁(不明白)
set statusline=%F%m%r%h%w\ [FORMAT=%{&ff}]\ [TYPE=%Y]\ [POS=%l,%v][%p%%]\ %{strftime(\"%d/%m/%y\ -\
%H:%M\")}   "状态行显示的内容
set laststatus=1    " 启动显示状态行(1),总是显示状态行(2)
set foldenable      " 允许折叠
set foldmethod=manual   " 手动折叠
"set background=dark "背景使用黑色
set nocompatible  "去掉讨厌的有关vi一致性模式，避免以前版本的一些bug和局限
" 显示中文帮助
if version >= 603
 set helplang=cn
 set encoding=utf-8
endif
" 设置配色方案
"colorscheme murphy
"字体
"if (has("gui_running"))
"   set guifont=Bitstream\ Vera\ Sans\ Mono\ 10
"endif
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
""""新文件标题
```

```
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
"新建.c,.h,.sh,.java文件，自动插入文件头
autocmd BufNewFile *.cpp,*.[ch],*.sh,*.java exec ":call SetTitle()"
""定义函数SetTitle，自动插入文件头
func SetTitle()
 "如果文件类型为.sh文件
 if &filetype == 'sh'
  call setline(1,"\#########################################################################")
  call append(line("."), "\# File Name: ".expand("%"))
  call append(line(".")+1, "\# Author: zhonghongfa")
  call append(line(".")+2, "\# mail: zhonghongfa@163.com")
  call append(line(".")+3, "\# Created Time: ".strftime("%c"))
  call append(line(".")+4, "\#########################################################################")
  call append(line(".")+5, "\#!/bin/bash")
  call append(line(".")+6, "")
 else
  call setline(1, "/*************************************************************************")
  call append(line("."), " > File Name: ".expand("%"))
  call append(line(".")+1, " > Author: zhonghongfa")
  call append(line(".")+2, " > Mail: zhonghongfa@163.com ")
  call append(line(".")+3, " > Created Time: ".strftime("%c"))
  call append(line(".")+4, " ************************************************************************/")
  call append(line(".")+5, "")
 endif
 if &filetype == 'cpp'
  call append(line(".")+6, "#include<iostream>")
  call append(line(".")+7, "using namespace std;")
  call append(line(".")+8, "")
 endif
 if &filetype == 'c'
  call append(line(".")+6, "#include<stdio.h>")
  call append(line(".")+7, "")
 endif
 " if &filetype == 'java'
 "  call append(line(".")+6,"public class ".expand("%"))
 "  call append(line(".")+7,"")
 " endif
 "新建文件后，自动定位到文件末尾
 autocmd BufNewFile * normal G
endfunc
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
"键盘命令
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""


nmap <leader>w :w!<cr>
nmap <leader>f :find<cr>

" 映射全选+复制 ctrl+a
```

```
map <C-A> ggVGY
map! <C-A> <Esc>ggVGY
map <F12> gg=G
" 选中状态下 Ctrl+c 复制
vmap <C-c> "+y
"去空行
nnoremap <F2> :g/^\s*$/d<CR>
"比较文件
nnoremap <C-F2> :vert diffsplit
"新建标签
map <M-F2> :tabnew<CR>
"列出当前目录文件
map <F3> :tabnew .<CR>
"打开树状文件目录
map <C-F3> \be
"C，C++ 按F5编译运行
map <F5> :call CompileRunGcc()<CR>
func! CompileRunGcc()
 exec "w"
 if &filetype == 'c'
  exec "!g++ % -o %<"
  exec "! ./%<"
 elseif &filetype == 'cpp'
  exec "!g++ % -o %<"
  exec "! ./%<"
 elseif &filetype == 'java'
  exec "!javac %"
  exec "!java %<"
 elseif &filetype == 'sh'
  :!./%
 elseif &filetype == 'py'
  exec "!python %"
  exec "!python %<"
 endif
endfunc
"C,C++的调试
map <F8> :call Rungdb()<CR>
func! Rungdb()
 exec "w"
 exec "!g++ % -g -o %<"
 exec "!gdb ./%<"
endfunc


""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
""实用设置
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
```

```
" 设置当文件被改动时自动载入
set autoread
" quickfix模式
autocmd FileType c,cpp map <buffer> <leader><space> :w<cr>:make<cr>
"代码补全
set completeopt=preview,menu
"允许插件
filetype plugin on
"共享剪贴板
set clipboard+=unnamed
"从不备份
set nobackup
"make 运行
:set makeprg=g++\ -Wall\ \ %
"自动保存
set autowrite
set ruler                   " 打开状态栏标尺
set cursorline              " 突出显示当前行
set magic                   " 设置魔术
set guioptions-=T           " 隐藏工具栏
set guioptions-=m           " 隐藏菜单栏
"set statusline=\ %<%F[%1*%M%*%n%R%H]%=\ %y\ %0(%{&fileformat}\ %{&encoding}\ %c:%l/%L%)\
" 设置在状态行显示的信息
set foldcolumn=0
set foldmethod=indent
set foldlevel=3
set foldenable              " 开始折叠
" 不要使用vi的键盘模式，而是vim自己的
set nocompatible
" 语法高亮
set syntax=on
" 去掉输入错误的提示声音
set noeb
" 在处理未保存或只读文件的时候，弹出确认
set confirm
" 自动缩进
set autoindent
set cindent
" Tab键的宽度
set tabstop=4
" 统一缩进为4
set softtabstop=4
set shiftwidth=4
" 不要用空格代替制表符
set noexpandtab
" 在行和段开始处使用制表符
set smarttab
" 显示行号
```

```
set number
" 历史记录数
set history=1000
"禁止生成临时文件
set nobackup
set noswapfile
"搜索忽略大小写
set ignorecase
"搜索逐字符高亮
set hlsearch
set incsearch
"行内替换
set gdefault
"编码设置
set enc=utf-8
set fencs=utf-8,ucs-bom,shift-jis,gb18030,gbk,gb2312,cp936
"语言设置
set langmenu=zh_CN.UTF-8
set helplang=cn
" 我的状态行显示的内容（包括文件类型和解码）
"set statusline=%F%m%r%h%w\ [FORMAT=%{&ff}]\ [TYPE=%Y]\ [POS=%l,%v][%p%%]\ %{strftime(\"%d/%m/%y\ -\
%H:%M\")}
"set statusline=[%F]%y%r%m%*%=[Line:%l/%L,Column:%c][%p%%]
" 总是显示状态行
set laststatus=2
" 命令行（在状态行下）的高度，默认为1，这里是2
set cmdheight=2
" 侦测文件类型
filetype on
" 载入文件类型插件
filetype plugin on
" 为特定文件类型载入相关缩进文件
filetype indent on
" 保存全局变量
set viminfo+=!
" 带有如下符号的单词不要被换行分割
set iskeyword+=_,$,@,%,#,-
" 字符间插入的像素行数目
set linespace=0
" 增强模式中的命令行自动完成操作
set wildmenu
" 使回格键（backspace）正常处理indent, eol, start等
set backspace=2
" 允许backspace和光标键跨越行边界
set whichwrap+=<,>,h,l
" 可以在buffer的任何地方使用鼠标（类似office中在工作区双击鼠标定位）
set mouse=a
set selection=exclusive
```

```
set selectmode=mouse,key
" 通过使用: commands命令，告诉我们文件的哪一行被改变过
set report=0
" 在被分割的窗口间显示空白，便于阅读
set fillchars=vert:\ ,stl:\ ,stlnc:\
" 高亮显示匹配的括号
set showmatch
" 匹配括号高亮的时间（单位是十分之一秒）
set matchtime=1
" 光标移动到buffer的顶部和底部时保持3行距离
set scrolloff=3
" 为C程序提供自动缩进
set smartindent
" 高亮显示普通txt文件（需要txt.vim脚本）
au BufRead,BufNewFile *  setfiletype txt
"自动补全
:inoremap ( ()<ESC>i
:inoremap ) <c-r>=ClosePair(')')<CR>
":inoremap { {<CR>}<ESC>O
":inoremap } <c-r>=ClosePair('}')<CR>
:inoremap [ []<ESC>i
:inoremap ] <c-r>=ClosePair(']')<CR>
:inoremap " ""<ESC>i
:inoremap ' ''<ESC>i
function! ClosePair(char)
 if getline('.')[col('.') - 1] == a:char
  return "\<Right>"
 else
  return a:char
 endif
endfunction
filetype plugin indent on
"打开文件类型检测, 加了这句才可以用智能补全
set completeopt=longest,menu
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
"NERDtee设定
let NERDChristmasTree=1
let NERDTreeAutoCenter=1
let NERDTreeBookmarksFile=$VIM.'\Data\NerdBookmarks.txt'
let NERDTreeMouseMode=2
let NERDTreeShowBookmarks=1
let NERDTreeShowFiles=1
let NERDTreeShowHidden=1
let NERDTreeShowLineNumbers=1
let NERDTreeWinPos='left'
let NERDTreeWinSize=31
nnoremap f :NERDTreeToggle
map <F7> :NERDTree<CR>
```

这个配置文件主要能帮我们实现：

1.按F5可以直接编译并执行C、C++、java代码以及执行shell脚本，按“F8”可进行C、C++代码的调试

2.自动插入文件头 ，新建C、C++源文件时自动插入表头：包括文件名、作者、联系方式、建立时间等，可根据需求自行更改

3.映射“Ctrl + A”为全选并复制快捷键，方便复制代码映射“Ctrl + C”

4.按“F2”可以直接消除代码中的空行

5.“F3”可列出当前目录文件，打开树状文件目录

6. 支持鼠标选择、方向键移动

7. 代码高亮，自动缩进，显示行号，显示状态行

8.按“Ctrl + P”可自动补全

9.[]、{}、()、″″、’ ’等都自动补全

10.其他功能.....略，看配置上的注释

vi/vim 的使用技巧

1.设定快捷键

:map 快捷键 快捷键执行的命令 自定义快捷键

vim 允许自定义快捷键，常用的自定义快捷键如下：

:map ^V I#<ESC> 按“ctrl+p”时，在行首加入注释

:map ^B ^x 按“ctrl+b”时，删除行首第一个字母（删除注释）

注意：^V 快捷键不能手工输入，需要执行 ctrl+V+P 来定义，或 ctrl+V ，然后 ctrl+P。^B 快捷键也是一样

2.多文件打开

在 vim 中可以同时打开两个文件，只要执行如下命令：

[root@localhost ~]# vim -o abc bcd

[root@localhost ~]# vim -O abc bcd

#-o 小写 o 会上下分屏打开两个文件

#-O 大写 O 会左右分屏打开两个文件

这样可以同时打开两个文件， 方便操作。

如果是 “-o” 上下打开两个文件， 可以通过先按 “ctrl+w” ，再按“上下箭头”的方式在两个文件之间切换。

如果是“-O”左右打开两个文件，可以通过先按“ctrl+w”，再按“左右箭头”的方式在两个文件之间切换。

最后，还要把前面遇到的一些坑，给大家填一下：

解决windos下写的脚本，传到linux里中文乱码

1、iconv命令转码

参数：

-f, --from-code=原始文本编码格式

-t, --to-code=输出编码格式

-o, --output=输出文件名

```
# iconv -f gb2312  -t utf8 aaa.txt  -o bbbb.txt
```

2、在win上另存为，换编码格式utf8

windows下编写的脚本和linux下编写的脚本，格式是不一样的，最明显回车行号符号，不是一回事，不通用！

把win下写的脚本转linux的方法：

1、在vi里：set ff=unix

2、安装dos2unix工具，用命令dos2unix xxxx文件名

反过来，把linux下写的脚本，在win上打开，就没的换行的，解决办法：

1、安装dos2unix工具，在传回win前，用命令 unix2dos xxxx文件名