

常见的编辑器

1. vim(编辑器之神)
2. emacs(神之编辑器)
3. `vi` (Unix 的编辑器，由它发展 `vim`)
4. `nano` (相比 `vim` 更加简单)

vim 的编辑模式

`vim` 的使用户在使用时脱离鼠标和键盘右侧部分，去编辑文件的文本（从最熟悉的文本文件，到各种语言写出的代码文件）。因此 `VIM` 两个大模式：

- **普通模式**（命令操作模式）：一切用鼠标和方向键操作文件的方式，在此模式下通过快捷键和命令解决，也是打开文件最先进入的模式。
- **编辑模式**(插入模式)：正常写文档那样对文件进行编辑

普通模式中又有：

- **命令模式**（输入 `:` 或 `/` 或 `?` 进入命令行模式）：例如：保存退出命令，查找替换命令，等。
- **可视化模式**（visual模式）：选择文本，进行其他操作。

按 `esc` 可推出各种模式

`shift` + 各种字母可写出大写

基础操作和技巧

使用 `VIM` 打开文件

`vim` 打开文件的目录

退出文件

退出文件需要用到命令模式，常见三种：

```
:w # 保存
:q # 推迟
! # 强制
:q! # 强制退出不保存文件
```

移动光标

选择在哪里插入字符，因省去方向键和鼠标，需要普通模式的快捷键，完成操作。

k # 向上移动一行
j # 向下移动一行
h # 向开头移动一个字符
l # 向末尾移动一个字符

b # 向开头移动 一个 单词或符号，光标位置在单词开头，如果光标在一个单词中第一下 **b** 会跳到该单词的开头
B # 向开头移动 多个 单词或符号，光标位置在单词开头

w # 向末尾移动 一个 单词或符号，光标位置在单词开头
W # 向末尾移动 多个 单词或符号，光标位置在单词开头

e # 向末尾移动 一个 单词或符号，光标位置在单词结尾，如果光标在一个单词中第一下 **e** 会跳到该单词的结尾
E # 向末尾移动 多个 单词或符号，光标位置在单词结尾

shift + 6 # 也就是 **^** 符号跳跃到本行到开头，忽略空格。
0 # 跳到开头，不忽略空格
shift + 4 # 也就是 **\$** 跳跃到本行末尾

{ # 光标以代码块为基础向开头跳跃
} # 光标以代码块为基础向末尾跳跃

翻页

查看时翻页

ctrl + f # 向下翻一页
ctrl + b # 向上翻一页
ctrl + e # 向下滚动一行
ctrl + y # 向上滚动一行
ctrl + G # 翻到整个文档末尾
ctrl + gg # 两下g 翻到整个文档的开头

编辑文本操作

```
# 进入文本编辑模式
i # 当前光标位置的前面插入
a # 当前光标位置的后面插入
o # 当前光标位置的下一行插入

# 普通模式下对文本操作
x # 删除光标所在字符
dd # 删除整个一行
u # 撤销
dw # 移除光标所在的单词往后的该单词字符（包括光标），注意仅限该光标所在的单词。因此如果光标在这个单词的开头这个单词就删掉了。
< > # 代码向左或向右缩进
```

“复制粘贴剪切”

`vim` 的复制与粘贴和传统不同，他有一个缓冲区，去存储你复制，甚至删除的字段。因此删除字段，在粘贴可以承担剪切的作用。

```
p # 粘贴
y # 复制
yy # 复制当前行
yw # 复制一个单词
y$ # 从当前开始往后复制到行末尾
```

Visual可视化模式

解决放弃鼠标后选择区域问题，该模式可配合：光标移动，编辑文本操作，以及复制粘贴。来使用。

```
v # 选择文本
V # 选择行
gg v G # 全选
ctrl+v # 矩阵选择
v o # 在使用 V 的基础上，切换文本首尾
vaw # 快速选择单词
vab # 包括括号
vaB # 包括大括号
va< # 包括尖括号
v u # v 选择后，u 都转化成小写
v U # v 选择后，U 都转化成大写
v ~ # v 选择后，~ 大写变小写，小写变大写。
```

查找文件并替换

命令模式操作

查找

/要查找的内容 #查找语法

n # 执行完查找命令的基础上，查下一个相似内容。

查找并替换

:s /查找内容/替换的内容 # 替换该行第一个查找到的内容（这个命令有局限性，光标必须在查找的内容那一行）

:s /查找内容/替换的内容/g # 替换该行查找到的所有内容（这个命令有局限性，光标必须在查找的内容那一行）

:%s /查找内容/替换的内容/g # % 整个文件替换文本中查到的所有内容

:%s /查找内容/替换的内容/gc # c 在替换时有一个确认的互动，其余同上

:set number # 显示行号

:行号, 行号s /查找内容/替换的内容/gc # 将多少行至多少行之间，替换文本中查到的所有内容，在替换时有一个确认的互动

vim的基础配置

在 home 目录下新建一个 .vimrc 文件，在这文件写入你要的配置内容。

简单的配置内容：

```
set tabstop=4          # 设置 tab键 空格个数量，只修改 tab 字符在 vim 中的显示宽度，不修改
插入模式下按 Tab 键
set softtabstop=4      # 修改插入模式下按 Tab 键空格数量。
set number             # 设置行号
set enc=utf-8          # 设置时区
```

（1）颜色设置

```
set syntax on          "开启代码高亮
set syntax off         "关闭代码高亮
set syntax enable      "开启代码高亮
```

（2）搜索设置

```
set hlsearch           "开启搜索高亮
set nohlsearch         "关闭搜索高亮
set incsearch          "输入搜索字符串的同时进行搜索
set ignorecase         "搜索时忽略大小写
```

（3）用户界面

```
set showmode           "开启模式显示
```

```
set ruler          "开启光标位置提示
set number         "显示行号
set nonu           "不显示行号
set cursorline     "强调光标所在行
set cmdheight=1    "命令部分高度为1
```

(4) 编辑辅助配置

```
set autoindent     "自动缩进
set noautoindent   "不自动缩进
set smartindent    "智能缩进
set autoread       "当文件在外部被改变时，vim自动更新载入
set showmatch      "显示匹配的括号
```

配置完保存退出输入

```
source .vimrc
```

source

在当前Shell环境中从指定文件读取和执行命令。

主要用途:执行文件并从文件中加载变量及函数到执行环境