

## 1. 关于 Vim

vim 是我最喜欢的编辑器，也是 **Linux** 下第二强大的编辑器。虽然 emacs 是公认的世界第一，我认为使用 emacs 并没有使用 vi 进行编辑来得高效。如果是初学 vi，运行一下 vimtutor 是个聪明的决定。（如果你的系统环境不是中文，而你想使用中文的 vimtutor，就运行 vimtutor zh）

## 1.1 Vim 的几种模式

- 正常模式：可以使用快捷键命令，或按:输入命令行。
- 插入模式：可以输入文本，在正常模式下，按 i、a、o 等都可以进入插入模式。
- 可视模式：正常模式下按 v 可以进入可视模式，在可视模式下，移动光标可以选择文本。按 V 进入可视行模式，总是整行整行的选中。ctrl+v 进入可视块模式。
- 替换模式：正常模式下，按 R 进入。

## 2. 启动 Vim

- vim -c cmd file: 在打开文件前，先执行指定的命令；
- vim -r file: 恢复上次异常退出的文件；
- vim -R file: 以只读的方式打开文件，但可以强制保存；
- vim -M file: 以只读的方式打开文件，不可以强制保存；
- vim -y num file: 将编辑窗口的大小设为 num 行；
- vim + file: 从文件的末尾开始；
- vim +num file: 从第 num 行开始；
- vim +/string file: 打开 file，并将光标停留在第一个找到的 string 上。
- vim --remote file: 用已有的 vim 进程打开指定的文件。如果你不想启用多个 vim 会话，这个很有用。但要注意，如果你用 vim，会寻找名叫 VIM 的服务器；如果你已经有一个 gvim 在运行了，你可以用 gvim --remote file 在已有的 gvim 中打开文件。

## 3. 文档操作

- :e file --关闭当前编辑的文件，并开启新的文件。如果对当前文件的修改未保存，vi 会警告。
- :e! file --放弃对当前文件的修改，编辑新的文件。
- :e+file -- 开始新的文件，并从文件尾开始编辑。
- :e+n file -- 开始新的文件，并从第 n 行开始编辑。
- :enew --编译一个未命名的新文档。(CTRL-W n)
- :e -- 重新加载当前文档。
- :e! -- 重新加载当前文档，并丢弃已做的改动。
- :e#或 ctrl+^ -- 回到刚才编辑的文件，很实用。
- :f 或 ctrl+g -- 显示文档名，是否修改，和光标位置。
- :f filename -- 改变编辑的文件名，这时再保存相当于另存为。
- gf -- 打开以光标所在字符串为文件名的文件。
- :w -- 保存修改。
- :n1,n2w filename -- 选择性保存从某 n1 行到另 n2 行的内容。
- :wq -- 保存并退出。
- ZZ -- 保存并退出。
- :x -- 保存并退出。
- :q[uit] ——退出当前窗口。(CTRL-W q 或 CTRL-W CTRL-Q)
- :saveas newfilename -- 另存为
- :browse e -- 会打开一个文件浏览器让你选择要编辑的文件。如果是终端中，则会打开 netrw 的文件浏览窗口；如果是 gvim，则会打开一个图形界面的浏览窗口。实际上:browse 后可以跟任何编辑文档的命令，如 sp 等。用 browse 打开的起始目录可以由 browsedir 来设置：
  - :set browsedir=last -- 用上次访问过的目录（默认）；
  - :set browsedir=buffer -- 用当前文件所在目录；

- :set browsedir=current -- 用当前工作目录；
- :Sex -- 水平分割一个窗口，浏览文件系统；
- :Vex -- 垂直分割一个窗口，浏览文件系统；

## 4. 光标的移动

### 4.1 基本移动

以下移动都是在 normal 模式下。

- h 或退格: 左移一个字符；
- l 或空格: 右移一个字符；
- j: 下移一行；
- k: 上移一行；
- gj: 移动到一段内的下一行；
- gk: 移动到一段内的上一行；
- +或 Enter: 把光标移至下一行第一个非空白字符。
- -: 把光标移至上一行第一个非空白字符。
- w: 前移一个单词，光标停在下一个单词开头；
- W: 移动下一个单词开头，但忽略一些标点；
- e: 前移一个单词，光标停在下一个单词末尾；
- E: 移动到下一个单词末尾，如果词尾有标点，则移动到标点；
- b: 后移一个单词，光标停在上一个单词开头；
- B: 移动到上一个单词开头，忽略一些标点；
- ge: 后移一个单词，光标停在上一个单词末尾；
- gE: 同 ge，不过‘单词’包含单词相邻的标点。
- (: 前移 1 句。
- ): 后移 1 句。
- {: 前移 1 段。
- }: 后移 1 段。
- fc: 把光标移到同一行的下一个 c 字符处
- Fc: 把光标移到同一行的上一个 c 字符处
- tc: 把光标移到同一行的下一个 c 字符前
- Tc: 把光标移到同一行的上一个 c 字符后
- :: 配合 f & t 使用，重复一次
- .: 配合 f & t 使用，反向重复一次

上面的操作都可以配合 n 使用，比如在正常模式(下面会讲到)下输入 3h，则光标向左移动 3 个字符。

- O: 移动到行首。
- gO: 移到光标所在屏幕行行首。
- ^: 移动到本行第一个非空白字符。
- g^: 同 ^，但是移动到当前屏幕行第一个非空字符处。
- \$: 移动到行尾。
- g\$: 移动光标所在屏幕行行尾。
- n|: 把光标移到第 n 列上。
- nG: 到文件第 n 行。
- :n<cr> 移动到第 n 行。
- :\$<cr> 移动到最后一行。
- H: 把光标移到屏幕最顶端一行。
- M: 把光标移到屏幕中间一行。

- L: 把光标移到屏幕最底端一行。
- gg: 到文件头部。
- G: 到文件尾部。

## 4.2 翻屏

- ctrl+f: 下翻一屏。
- ctrl+b: 上翻一屏。
- ctrl+d: 下翻半屏。
- ctrl+u: 上翻半屏。
- ctrl+e: 向下滚动一行。
- ctrl+y: 向上滚动一行。
- n%: 到文件 n%的位置。
- zz: 将当前行移动到屏幕中央。
- zt: 将当前行移动到屏幕顶端。
- zb: 将当前行移动到屏幕底端。

## 4.3 标记

使用标记可以快速移动。到达标记后，可以用 Ctrl+o 返回原来的位置。Ctrl+o 和 Ctrl+i 很像浏览器上的 *后退* 和 *前进*。

- m{a-z}: 标记光标所在位置，局部标记，只用于当前文件。
- m{A-Z}: 标记光标所在位置，全局标记。标记之后，退出 Vim，重新启动，标记仍然有效。
- `{a-z}: 移动到标记位置。
- '{a-z}: 移动到标记行的行首。
- `{0-9}: 回到上[2-10]次关闭 vim 时最后离开的位置。
- ``: 移动到上次编辑的位置。"也可以，不过``精确到列，而"精确到行。如果想跳转到更老的位置，可以按 C-o，跳转到更新的位置用 C-i。
- `": 移动到上次离开的地方。
- `.: 移动到最后改动的地方。
- :marks 显示所有标记。
- :delmarks a b -- 删除标记 a 和 b。
- :delmarks a-c -- 删除标记 a、b 和 c。
- :delmarks a c-f -- 删除标记 a、c、d、e、f。
- :delmarks! -- 删除当前缓冲区的所有标记。
- :help mark-motions 查看更多关于 mark 的知识。

## 5. 插入文本

### 5.1 基本插入

- i: 在光标前插入；一个小技巧：按 8，再按 i，进入插入模式，输入=，按 esc 进入命令模式，就会出现 8 个=。这在插入分割线时非常有用，如 30i+<esc>就插入了 36 个+组成的分割线。
- I: 在当前行第一个非空字符前插入；
- gI: 在当前行第一列插入；
- a: 在光标后插入；
- A: 在当前行最后插入；
- o: 在下面新建一行插入；
- O: 在上面新建一行插入；
- :r filename 在当前位置插入另一个文件的内容。
- :[n]r filename 在第 n 行插入另一个文件的内容。
- :r !date 在光标处插入当前日期与时间。同理，:r !command 可以将其它 shell 命令的输出插入当前文档。

## 5.2 改写插入

- c[n]w: 改写光标后 1(n)个词。
- c[n]l: 改写光标后 n 个字母。
- c[n]h: 改写光标前 n 个字母。
- [n]cc: 修改当前[n]行。
- [n]s: 以输入的文本替代光标之后 1(n)个字符，相当于 c[n]l。
- [n]S: 删除指定数目的行，并以所输入文本代替之。

注意，类似 cnw,dnw,ynw 的形式同样可以写为 ncw,ndw,nyw。

## 6. 剪切复制和寄存器

### 6.1 剪切和复制、粘贴

- [n]x: 剪切光标右边 n 个字符，相当于 d[n]l。
- [n]X: 剪切光标左边 n 个字符，相当于 d[n]h。
- y: 复制在可视模式下选中的文本。
- yy or Y: 复制整行文本。
- y[n]w: 复制一(n)个词。
- y[n]l: 复制光标右边 1(n)个字符。
- y[n]h: 复制光标左边 1(n)个字符。
- y\$: 从光标当前位置复制到行尾。
- y0: 从光标当前位置复制到行首。
- :m,ny<cr> 复制 m 行到 n 行的内容。
- y1G 或 ygg: 复制光标以上的所有行。
- yG: 复制光标以下的行。
- yaw 和 yas: 复制一个词和复制一个句子，即使光标不在词首和句首也没关系。
- d: 删除（剪切）在可视模式下选中的文本。
- d\$ or D: 删除（剪切）当前位置到行尾的内容。
- d[n]w: 删除（剪切）1(n)个单词
- d[n]l: 删除（剪切）光标右边 1(n)个字符。
- d[n]h: 删除（剪切）光标左边 1(n)个字符。
- d0: 删除（剪切）当前位置到行首的内容
- [n] dd: 删除（剪切）1(n)行。
- :m,nd<cr> 剪切 m 行到 n 行的内容。
- d1G 或 dgg: 剪切光标以上的所有行。
- dG: 剪切光标以下的行。
- daw 和 das: 剪切一个词和剪切一个句子，即使光标不在词首和句首也没关系。
- d/f<cr>: 这是一个比较高级的组合命令，它将删除当前位置 到下一个 f 之间的内容。
- p: 在光标之后粘贴。
- P: 在光标之前粘贴。

### 6.2 文本对象

- aw: 一个词
- as: 一句。
- ap: 一段。
- ab: 一块（包含在圆括号中的）。

y, d, c, v 都可以跟文本对象。

### 6.3 寄存器

- a-z: 都可以用作寄存器名。"ayy 把当前行的内容放入 a 寄存器。

- A-Z：用大写字母索引寄存器，可以在寄存器中追加内容。如"Ayy 把当前行的内容追加到 a 寄存器中。
- :reg 显示所有寄存器的内容。
- ""：不加寄存器索引时，默认使用的寄存器。
- "\*"：当前选择缓冲区，"\*yy 把当前行的内容放入当前选择缓冲区。
- "+"：系统剪贴板。"+yy 把当前行的内容放入系统剪贴板。

## 7. 查找与替换

### 7.1 查找

- /something: 在后面的文本中查找 something。
- ?something: 在前面的文本中查找 something。
- /pattern/+number: 将光标停在包含 pattern 的行后面第 number 行上。
- /pattern/-number: 将光标停在包含 pattern 的行前面第 number 行上。
- n: 向后查找下一个。
- N: 向前查找下一个。

可以用 grep 或 vimgrep 查找一个模式都在哪些地方出现过，其中:grep 是调用外部的 grep 程序，而:vimgrep 是 vim 自己的查找算法。

用法为：:vim[grep]/pattern/[g] [j] files

g 的含义是如果一个模式在一行中多次出现，则这一行也在结果中多次出现。

j 的含义是 grep 结束后，结果停在第 j 项，默认是停在第一项。

vimgrep 前面可以加数字限定搜索结果的上限，如

:1vim/pattern/ % 只查找那个模式在本文件中的第一个出现。

其实 vimgrep 在读纯文本电子书时特别有用，可以生成导航的目录。

比如电子书中每一节的标题形式为：n. xxxx。你就可以这样：

:vim/^d{1,}/ %

然后用:cw 或:copen 查看结果，可以用 C-w H 把 quickfix 窗口移到左侧，就更像个目录了。

### 7.2 替换

- :s/old/new - 用 new 替换当前行第一个 old。
- :s/old/new/g - 用 new 替换当前行所有的 old。
- :n1,n2s/old/new/g - 用 new 替换文件 n1 行到 n2 行所有的 old。
- :%s/old/new/g - 用 new 替换文件中所有的 old。
- :%s/^/xxx/g - 在每一行的行首插入 xxx，^表示行首。
- :%s/\$/xxx/g - 在每一行的行尾插入 xxx，\$表示行尾。
- 所有替换命令末尾加上 c，每个替换都将需要用户确认。如：%s/old/new/gc，加上 i 则忽略大小写(ignore)。

还有一种比替换更灵活的方式，它是匹配到某个模式后执行某种命令，

语法为 :[range]g/pattern/command

例如 :%g/^ xyz/normal dd。

表示对于以一个空格和 xyz 开头的行执行 normal 模式下的 dd 命令。

关于 range 的规定为：

- 如果不指定 range，则表示当前行。
- m,n: 从 m 行到 n 行。
- 0: 最开始一行（可能是这样）。
- \$: 最后一行
- .: 当前行

- %: 所有行

## 7.3 正则表达式

高级的查找替换就要用到正则表达式。

- \d: 表示十进制数（我猜的）
- \s: 表示空格
- \S: 非空字符
- \a: 英文字母
- \|: 表示 或
- \.: 表示.
- {m,n}: 表示 m 到 n 个字符。这要和 \s 与 \a 等连用，如 \a{m,n} 表示 m 到 n 个英文字母。
- {m,}: 表示 m 到无限多个字符。
- \*\*: 当前目录下的所有子目录。

:help pattern 得到更多帮助。

---

## 8. 排版

### 8.1 基本排版

- << 向左缩进一个 shiftwidth
- >> 向右缩进一个 shiftwidth
- :ce(n)ter 本行文字居中
- :le(ft) 本行文字靠左
- :ri(ght) 本行文字靠右
- gq 对选中的文字重排，即对过长的文字进行断行
- gqq 重排当前行
- gqnq 重排 n 行
- gqap 重排当前段
- gqnap 重排 n 段
- gqnj 重排当前行和下面 n 行
- gqQ 重排当前段对文章末尾
- J 拼接当前行和下一行
- gj 同 J，不过合并后不留空格。

### 8.2 拼写检查

- :set spell - 开启拼写检查功能
- :set nospell - 关闭拼写检查功能
- ]s - 移到下一个拼写错误的单词
- [s - 作用与上一命令类似，但它是从相反方向进行搜索
- z= - 显示一个有关拼写错误单词的列表，可从中选择
- zg - 告诉拼写检查器该单词是拼写正确的
- zw - 与上一命令相反，告诉拼写检查器该单词是拼写错误的

### 8.3 统计字数

g ^g 可以统计文档字符数，行数。将光标放在最后一个字符上，用字符数减去行数可以粗略统计中文文档的字数。以上对 Mac 或 Unix 的文件格式适用。如果是 Windows 文件格式（即换行符有两个字节），字数的统计方法为：字符数 - 行数 \* 2。

---

## 9. 编辑多个文件

### 9.1 一次编辑多个文件

我们可以一次打开多个文件，如

```
vi a.txt b.txt c.txt
```

- 使用:next(:n)编辑下一个文件。
- :2n 编辑下 2 个文件。
- 使用:previous 或:N 编辑上一个文件。
- 使用:wnext，保存当前文件，并编辑下一个文件。
- 使用:wprevious，保存当前文件，并编辑上一个文件。
- 使用:args 显示文件列表。
- :n filenames 或:args filenames 指定新的文件列表。
- vi -o filenames 在水平分割的多个窗口中编辑多个文件。
- vi -O filenames 在垂直分割的多个窗口中编辑多个文件。

### 9.2 多标签编辑

- vim -p files: 打开多个文件，每个文件占用一个标签页。
- :tabe, tabnew -- 如果加文件名，就新的标签中打开这个文件，否则打开一个空缓冲区。
- ^w gf -- 在新的标签页里打开光标下路径指定的文件。
- :tabn -- 切换到下一个标签。Control + PageDown，也可以。
- :tabp -- 切换到上一个标签。Control + PageUp，也可以。
- [n] gt -- 切换到下一个标签。如果前面加了 *n*，就切换到第 *n* 个标签。第一个标签的序号就是 1。
- :tab split -- 将当前缓冲区的内容在新页签中打开。
- :tabc[lose] -- 关闭当前的标签页。
- :tabo[nly] -- 关闭其它的标签页。
- :tabs -- 列出所有的标签页和它们包含的窗口。
- :tabm[ove] [N] -- 移动标签页，移动到第 *N* 个标签页之后。如 *tabm 0* 当前标签页，就会变成第一个标签页。

### 9.3 缓冲区

- :buffers 或:ls 或:files 显示缓冲区列表。
- ctrl+^：在最近两个缓冲区间切换。
- :bn -- 下一个缓冲区。
- :bp -- 上一个缓冲区。
- :bl -- 最后一个缓冲区。
- :b[n]或:[n]b -- 切换到第 *n* 个缓冲区。
- :nbw(ipeout) -- 彻底删除第 *n* 个缓冲区。
- :nbd(elete) -- 删除第 *n* 个缓冲区，并未真正删除，还在 unlisted 列表中。
- :ba[ll] -- 把所有的缓冲区在当前页中打开，每个缓冲区占一个窗口。

## 10. 分屏编辑

- vim -o file1 file2:水平分割窗口，同时打开 file1 和 file2
- vim -O file1 file2:垂直分割窗口，同时打开 file1 和 file2

### 10.1 水平分割

- :split(:sp) -- 把当前窗水平分割成两个窗口。(CTRL-W s 或 CTRL-W CTRL-S) 注意如果在终端下，CTRL-S 可能会冻结终端，请按 CTRL-Q 继续。
- :split filename -- 水平分割窗口，并在新窗口中显示另一个文件。
- :nsplit(:nsp) -- 水平分割出一个 *n* 行高的窗口。



- :[N]new -- 水平分割出一个 N 行高的窗口，并编辑一个新文件。(CTRL-W n 或 CTRL-W CTRL-N)
- ctrl+w f -- 水平分割出一个窗口，并在新窗口打开名称为光标所在词的文件。
- C-w C-^ -- 水平分割一个窗口，打开刚才编辑的文件。

## 10.2 垂直分割

- :vsplit(:vsp) -- 把当前窗口分割成水平分布的两个窗口。(CTRL-W v 或 CTRL CTRL-V)
- :[N]vne[w] -- 垂直分割出一个新窗口。
- :vertical 水平分割的命令：相应的垂直分割。

## 10.3 关闭子窗口

- :qall -- 关闭所有窗口，退出 vim。
- :wall -- 保存所有修改过的窗口。
- :only -- 只保留当前窗口，关闭其它窗口。(CTRL-W o)
- :close -- 关闭当前窗口，CTRL-W c 能实现同样的功能。(象 :q :x 同样工作)

## 10.4 调整窗口大小

- ctrl+w + -- 当前窗口增高一行。也可以用 n 增高 n 行。
- ctrl+w - -- 当前窗口减小一行。也可以用 n 减小 n 行。
- ctrl+w \_ -- 当前窗口扩展到尽可能的大。也可以用 n 设定行数。
- :resize n -- 当前窗口 n 行高。
- ctrl+w = -- 所有窗口同样高度。
- n ctrl+w \_ -- 当前窗口的高度设定为 n 行。
- ctrl+w < -- 当前窗口减少一列。也可以用 n 减少 n 列。
- ctrl+w > -- 当前窗口增宽一列。也可以用 n 增宽 n 列。
- ctrl+w | -- 当前窗口尽可能的宽。也可以用 n 设定列数。

## 10.5 切换和移动窗口

如果支持鼠标，切换和调整子窗口的大小就简单了。

- ctrl+w ctrl+w: 切换到下一个窗口。或者是 ctrl+w w。
- ctrl+w p: 切换到前一个窗口。
- ctrl+w h(l,j,k): 切换到左（右，下，上）的窗口。
- ctrl+w t(b): 切换到最上（下）面的窗口。 <BR>
- ctrl+w H(L,K,J): 将当前窗口移动到最左（右、上、下）面。
- ctrl+w r: 旋转窗口的位置。
- ctrl+w T: 将当前的窗口移动到新的标签页上。

## 11. 快速编辑

### 11.1 改变大小写

- ~: 反转光标所在字符的大小写。
- 可视模式下的 U 或 u: 把选中的文本变为大写或小写。
- gu(U)接范围（如 \$, 或 G），可以把从光标当前位置到指定位置之间字母全部 转换成小写或大写。如 ggguG, 就是把开头到最后一行之间的字母全部变为小写。再如 gu5j, 把当前行和下面四行全部变成小写。

### 11.2 替换（normal 模式）

- r: 替换光标处的字符，同样支持汉字。
- R: 进入替换模式，按 esc 回到正常模式。

### 11.3 撤消与重做（normal 模式）

- [n] u: 取消一(n)个改动。
- :undo 5 -- 撤销 5 个改变。



- :undolist -- 你的撤销历史。
- ctrl + r: 重做最后的改动。
- U: 取消当前行中所有的改动。
- :earlier 4m -- 回到 4 分钟前
- :later 55s -- 前进 55 秒

#### 11.4 宏

- . --重复上一个编辑动作
- qa : 开始录制宏 a ( 键盘操作记录 )
- q : 停止录制
- @a : 播放宏 a

### 12. 编辑特殊文件

#### 12.1 文件加解密

- vim -x file: 开始编辑一个加密的文件。
- :X -- 为当前文件设置密码。
- :set key= -- 去除文件的密码。

这里是 滇狐总结的比較高级的 vi 技巧。

#### 12.2 文件的编码

- :e ++enc=utf8 filename, 让 vim 用 utf-8 的编码打开这个文件。
- :w ++enc=gbk , 不管当前文件什么编码, 把它转存成 gbk 编码。
- :set fenc 或:set fileencoding , 查看当前文件的编码。
- 在 vimrc 中添加 set fileencoding=ucs-bom,utf-8,cp936 , vim 会根据要打开的文件选择合适的编码。注意: 编码之间不要留空格。 cp936 对应于 gbk 编码。 ucs-bom 对应于 windows 下的文件格式。

让 vim 正确处理文件格式和文件编码, 有赖于 ~/.vimrc 的正确配置

#### 12.3 文件格式

大致有三种文件格式: unix, dos, mac. 三种格式的区别主要在于回车键的编码: dos 下是回车加换行, unix 下只有换行符, mac 下只有回车符。

- :e ++ff=dos filename, 让 vim 用 dos 格式打开这个文件。
- :w ++ff=mac filename, 以 mac 格式存储这个文件。
- :set ff , 显示当前文件的格式。
- 在 vimrc 中添加 set fileformats=unix,dos,mac , 让 vim 自动识别文件格式。

### 13. 编程辅助

#### 13.1 一些按键

- gd: 跳转到局部变量的定义处;
- gD: 跳转到全局变量的定义处, 从当前文件开头开始搜索;
- g;: 上一个修改过的地方;
- g,: 下一个修改过的地方;
- [[: 跳转到上一个函数块开始, 需要有单独一行的{。
- ]]: 跳转到下一个函数块开始, 需要有单独一行的{。
- []: 跳转到上一个函数块结束, 需要有单独一行的}。
- ]]: 跳转到下一个函数块结束, 需要有单独一行的}。
- [{: 跳转到当前块开始处;
- ]}: 跳转到当前块结束处;

- [/: 跳转到当前注释块开始处；
- ]/: 跳转到当前注释块结束处；
- %: 不仅能移动到匹配的(),{}或[]上，而且能在#if，#else，#endif之间跳跃。

下面的括号匹配对编程很实用的。

- ci', di', yi' : 修改、剪切或复制'之间的内容。
- ca', da', ya' : 修改、剪切或复制'之间的内容，包含'。
- ci", di", yi" : 修改、剪切或复制"之间的内容。
- ca", da", ya" : 修改、剪切或复制"之间的内容，包含"。
- ci(, di(, yi( : 修改、剪切或复制()之间的内容。
- ca(, da(, ya( : 修改、剪切或复制()之间的内容，包含()。
- ci[, di[, yi[ : 修改、剪切或复制[]之间的内容。
- ca[, da[, ya[ : 修改、剪切或复制[]之间的内容，包含[]。
- ci{, di{, yi{ : 修改、剪切或复制{}之间的内容。
- ca{, da{, ya{ : 修改、剪切或复制{}之间的内容，包含{}。
- ci<, di<, yi< : 修改、剪切或复制<>之间的内容。
- ca<, da<, ya< : 修改、剪切或复制<>之间的内容，包含<>。

### 13.2 ctags

- ctags -R: 生成 tag 文件，-R 表示也为子目录中的文件生成 tags
- :set tags=path/tags -- 告诉 ctags 使用哪个 tag 文件
- :tag xyz -- 跳到 xyz 的定义处，或者将光标放在 xyz 上按 C-]，返回用 C-t
- :stag xyz -- 用分割的窗口显示 xyz 的定义，或者 C-w ]，如果用 C-w n ]，就会打开一个 n 行高的窗口
- :ptag xyz -- 在预览窗口中打开 xyz 的定义，热键是 C-w }。
- :pclose -- 关闭预览窗口。热键是 C-w z。
- :pedit abc.h -- 在预览窗口中编辑 abc.h
- :psearch abc -- 搜索当前文件和当前文件 include 的文件，显示包含 abc 的行。

有时一个 tag 可能有多个匹配，如函数重载，一个函数名就会有多个匹配。这种情况会先跳转到第一个匹配处。

- :[n]tnext -- 下一[n]个匹配。
- :[n]tprev -- 上一[n]个匹配。
- :tfirst -- 第一个匹配
- :tlast -- 最后一个匹配
- :tselect tagname -- 打开选择列表

tab 键补齐

- :tag xyz<tab> -- 补齐以 xyz 开头的 tag 名，继续按 tab 键，会显示其他的。
- :tag /xyz<tab> -- 会用名字中含有 xyz 的 tag 名补全。

### 13.3 cscope

- cscope -Rbq: 生成 cscope.out 文件
- :cs add /path/to/cscope.out /your/work/dir
- :cs find c func -- 查找 func 在哪些地方被调用
- :cw -- 打开 quickfix 窗口查看结果

### 13.4 gtags

Gtags 综合了 ctags 和 cscope 的功能。使用 Gtags 之前，你需要安装 GNU Gtags。然后在工程目录

运行 *gtags*。

- `:Gtags funcname` 定位到 `funcname` 的定义处。
- `:Gtags -r funcname` 查询 `funcname` 被引用的地方。
- `:Gtags -s symbol` 定位 `symbol` 出现的地方。
- `:Gtags -g string` Goto `string` 出现的地方。 `:Gtags -gi string` 忽略大小写。
- `:Gtags -f filename` 显示 `filename` 中的函数列表。 你可以用 `:Gtags -f %` 显示当前文件。
- `:Gtags -P pattern` 显示路径中包含特定模式的文件。 如 `:Gtags -P .h$` 显示所有头文件， `:Gtags -P /vm/` 显示 `vm` 目录下的文件。

### 13.5 编译

`vim` 提供了 `:make` 来编译程序，默认调用的是 `make`，如果你当前目录下有 `makefile`，简单地 `:make` 即可。

如果你没有 `make` 程序，你可以通过配置 `makeprg` 选项来更改 `make` 调用的程序。如果你只有一个 `abc.java` 文件，你可以这样设置：

```
set makeprg=javac\ abc.java
```

然后 `:make` 即可。如果程序有错，可以通过 `quickfix` 窗口查看错误。不过如果要正确定位错误，需要设置好 `errorformat`，让 `vim` 识别错误信息。如：

```
:setl efm=%A%f:%l:\ %m,%-Z%p^,%-C%.%#
```

`%f` 表示文件名，`%l` 表示行号，`%m` 表示错误信息，其它的还不能理解。请参考 `:help errorformat`。

### 13.6 快速修改窗口

其实是 `quickfix` 插件提供的功能，对编译调试程序非常有用：)

- `:copen` -- 打开快速修改窗口。
- `:cclose` -- 关闭快速修改窗口。

快速修改窗口在 `make` 程序时非常有用，当 `make` 之后：

- `:cl` -- 在快速修改窗口中列出错误。
- `:cn` -- 定位到下一个错误。
- `:cp` -- 定位到上一个错误。
- `:cr` -- 定位到第一个错误。

### 13.7 自动补全

- `C-x C-s` -- 拼写建议。
- `C-x C-v` -- 补全 `vim` 选项和命令。
- `C-x C-l` -- 整行补全。
- `C-x C-f` -- 自动补全文件路径。弹出菜单后，按 `C-f` 循环选择，当然也可以按 `C-n` 和 `C-p`。
- `C-x C-p` 和 `C-x C-n` -- 用文档中出现过的单词补全当前的词。直接按 `C-p` 和 `C-n` 也可以。
- `C-x C-o` -- 编程时可以补全关键字和函数名啊。
- `C-x C-i` -- 根据头文件内关键字补全。
- `C-x C-d` -- 补全宏定义。
- `C-x C-n` -- 按缓冲区中出现过的关键字补全。直接按 `C-n` 或 `C-p` 即可。

当弹出补全菜单后：

- `C-p` 向前切换成员；
- `C-n` 向后切换成员；

- C-e 退出下拉菜单，并退回到原来录入的文字；
- C-y 退出下拉菜单，并接受当前选项。

### 13.8 多行缩进缩出

- 正常模式下，按两下>;光标所在行会缩进。
- 如果先按了 n，再按两下>;，光标以下的 n 行会缩进。
- 对应的，按两下<;，光标所在行会缩出。
- 如果在编辑代码文件，可以用=进行调整。
- 在可视模式下，选择要调整的代码块，按=，代码会按书写规则缩排好。
- 或者 n =，调整 n 行代码的缩排。

### 13.9 折叠

- zf -- 创建折叠的命令，可以在一个可视区域上使用命令；
- zd -- 删除当前行的折叠；
- zD -- 删除当前行的折叠；
- zfaf -- 折叠光标所在的段；
- zo -- 打开折叠的文本；
- zc -- 收起折叠；
- za -- 打开/关闭当前折叠；
- zr -- 打开嵌套的折行；
- zm -- 收起嵌套的折行；
- zR (zO) -- 打开所有折行；
- zM (zC) -- 收起所有折行；
- zj -- 跳到下一个折叠处；
- zk -- 跳到上一个折叠处；
- zi -- enable/disable fold;

## 14. 命令行

normal 模式下按:进入命令行模式

### 14.1 命令行模式下的快捷键：

- 上下方向键：上一条或者下一条命令。如果已经输入了部分命令，则找上一条或者下一条匹配的命令。
- 左右方向键：左/右移一个字符。
- C-w：向前删除一个单词。
- C-h：向前删除一个字符，等同于 Backspace。
- C-u：从当前位置移动到命令行开头。
- C-b：移动到命令行开头。
- C-e：移动到命令行末尾。
- Shift-Left：左移一个单词。
- Shift-Right：右移一个单词。
- @：重复上一次的冒号命令。
- q：正常模式下，q 然后按'!'，打开命令行历史缓冲区，可以像编辑文件一样编辑命令。
- q/和 q? 可以打开查找历史记录。

### 14.2 执行外部命令

- :! cmd 执行外部命令。
- :!! 执行上一次的外部命令。
- :sh 调用 shell，用 exit 返回 vim。
- :r !cmd 将命令的返回结果插入文件当前位置。
- :m,nw !cmd 将文件的 m 行到 n 行之间的内容做为命令输入执行命令。

## 15. 其它

### 15.1 工作目录

- `:pwd` 显示 vim 的工作目录。
- `:cd path` 改变 vim 的工作目录。
- `:set autochdir` 可以让 vim 根据编辑的文件自动切换工作目录。

### 15.2 一些快捷键（收集中）

- `K`: 打开光标所在词的 manpage。
- `*`: 向下搜索光标所在词。
- `g*`: 同上，但部分符合即可。
- `#`: 向上搜索光标所在词。
- `g#`: 同上，但部分符合即可。
- `g C-g`: 统计全文或统计部分的字数。

### 15.3 在线帮助

- `:h(elp)`或 `F1` 打开总的帮助。
- `:help user-manual` 打开用户手册。
- 命令帮助的格式为：第一行指明怎么使用那个命令；然后是缩进的一段解释这个命令的作用，然后是进一步的信息。
- `:helptags somepath` 为 somepath 中的文档生成索引。
- `:helpgrep` 可以搜索整个帮助文档，匹配的列表显示在 quickfix 窗口中。
- `Ctrl+j` 跳转到 tag 主题，`Ctrl+t` 跳回。
- `:ver` 显示版本信息。

### 15.4 一些小功能

- 简单计算器: 在插入模式下，输入 `C-r =`，然后输入表达式，就能在 光标处得到计算结果。