vim效率指南

文件管理器

从shell打开

```
# 从shell打开
vim PATH
```

从vim中打开

```
:Explore (打开当前文件目录)
:Lexplore (在左侧打开)
:Texplore (在新标签页打开)
```

增强插件

`NERD tree`

https://github.com/scrooloose/nerdtree

标签页

新建标签页

```
:tabnew [文件名] (新建标签页)
:tab split (新建标签页为当前buffer内容)
:tab ball (为当前打开的所有buffer新建标签页)
:tabfind (匹配通配符打开标签页)
```

列出所有标签页

```
:tabs
```

关闭标签页

```
:tabclose
:tabonly 只保留当前页
```

或者在normal模式下`Ctrl-W`+`c`

切换标签页

```
:tabn 切换到下一标签页
:tabp 切换到上一标签页
```

或者在noraml模式下`gt`切换到下一标签页/`gT`切换到上一标签页

`Ngt` 切换到第N个标签页。

窗口操作

```
:split 分屏
:vsplit 竖直分屏
```

`ctrl`+`w` 为窗口操作触发键

触发键+`h/i/j/k` 移动到对应位置窗口

触发键+`+/-` 调整窗口高度

触发键+`=` 将所有窗口高度平均分摊

十六进制编辑器

```
:%!xxd（调用外部xxd命令，以全文为输入获取16进制结果）
:%!xxd -r（以16进制结果为输入获得还原的结果）
```

文件比较

```
vimdiff FILE1 FILE2
# 或者
vim -d FILE1 FILE2
```

使用`dp`将当前不同的块应用到另一文件

使用`do`将另一文件不同的块应用到当前文件

行编辑和块编辑

在normal模式下使用`v`进入行编辑模式

在normal模式下使用`ctrl`+`v`进入列编辑模式，批量注释常用

快速移动

基本操作

`f`移动到下一字符，`F`反向移动，`;`重复，`,`反向重复

`w`移动掉下一单词开头，`W`将所有相连符号当成一个单词

`b`移动到上一单词开头，`B`将所有相连符号当成一个单词

`e`移动到当前单词结尾，`E`将所有相连符号当成一个单词

`{/}` 移动到段落首，段落尾

`0`移动到行开头

`^`移动到首个非空字符

`$` 移动到行尾

`gg` 移动到首行，`G` 移动到尾行

`Ng` / `:N` 移动到第N行

`%` 移动到匹配的括号引号

`ctrl+u` 向上滚半屏 `ctrl+d` 向下滚半屏

`ctrl+f` 向下滚一屏 `ctrl+b` 向上滚一屏

复合操作

可配合数量执行多次。e.g

`3fa` 移动到往后3个a处。

编辑操作

基本操作

`y` 复制，`yy` / `Y` 复制一整行

`p` 在光标后粘贴，`P` 在光标前粘贴

`c` 修改，`C` 删除光标之后内容并切为插入模式, `cc` 删除一整行并切为插入模式

`d` 删除，`D` 删除光标之后内容，`dd` 删除一整行

`r` 替换单个字符 ，`R` 进入替换模式

`s` 删除一个字符并切为插入模式，`S` 同 `cc`

`J` 与下一行合并

`i` 在光标前插入，`I` 在首个非空字符前插入，`a` 在光标后插入，`A` 在行尾插入

`o` 在下一行插入一行，`O` 在上一行插入一行

`u` 撤销上一操作 ，`ctrl` + `r` 重复下一操作, `U` 取消当前行中所有的改动

复合操作

`d2w` 删除2个词

`ci(` 删除括号内的内容并切为插入模式

`ya"` 复制包括引号的引号内内容

`dfa` 删除到下个a为止的内容，包括a

`yta` 复制到下个a之前的内容

as：一句。

ap：一段。

ab：一块（包含在圆括号中的）。

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。

宏

录制宏

1. `q`+寄存器
2. 进行要录制的操作
3. `q`

使用宏

`@`+寄存器

可以+N重复执行N次

使用`.`可重复上次操作，也可和N配合执行多次，可看作特殊的宏。

查找和替换

使用系统的grep工具

在vim中用`:grep`可以直接调用系统的grep工具进行搜索。

搜索结果可以用`:copen`展示，并且在对应行按回车可以直接跳转。

使用vimgrep

```
:vimgrep/pattern/g files
```

vim内文本查找

使用 `/`/`?`+要查找的内容 进行查找，可在查找内容前加入`\v`进行正则匹配

使用`#`/`*`可向上/向下查找当前单词，查找变量定义位置时常用

替换

使用

```
:s/old/new/g
```

进行文本替换。s前可加入范围限定，如`:%s/old/new`可全文替换

命令末尾加入`c`可每条替换都手动确认

范围限定

> m,n: 从m行到n行。
>
> $: 最后一行
>
> .: 当前行
>
> %: 所有行

匹配后执行命令

```
:g/pattern/command
```

例如，删除匹配到aaa开头的行

```
:%g/^aaa/normal dd
```

缩进

`>>`或`<<`当前行向后/向前缩进

`==`当前行自动缩进，选择范围内`=`选中范围自动缩进

ctags

使用ctags生产的tag可在vim中使用。

生成tags

```
ctags -R
```

使用tag

vim中`ctrl`+`w` `}`可以打开当前tag的预览窗口

也可以用

```
:ptag 关键字
```

打开预览窗口，关键字可用tab补全

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。

make

vim中可以直接使用

```
:make
```

来调用make进行编译，编译产生的错误同样可以`:copen`打开并直接定位。

对于`:make`执行的内容可以更改makeprg来自定义编译命令。

例如

```
set makeprg=cd\ build\;cmake\ ..\;make
```

对于编译的输出可以指定efm来自定义输出的解析

例如

```
: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，具体参考 :help errorformat。

其他技巧

插入外部命令的输出

写配置或处理数据时常用

`:r!ls`将ls的结果插入文本，配合列选择进行进一步处理

插入模式下直接计算

使用`ctrl`+`r` `=`接上要计算的表达式即可插入计算结果

数字自增自减

`ctrl`+`a`当前位置数字自增，`ctrl`+`x`当前数字自减

常配合宏一起使用

直接执行python脚本

选中要执行的python脚本后输入

```
:!python
```

代码段会被替换成执行后的结果。其他脚本解释器一样，原理同利用xxd做16进制编辑

加快选择速度

光标在选择位置中间时先向后选择到末尾，再按`o`将光标移到原位向前选择可以节省光标移动的次数

命令行模式下快速插入单词

在需要插入的地方，例如替换中的匹配关键字，按`ctrl`+`r`再按`ctrl`+`w`就能插入当前光标所在的单词

在`ctrl`+`r`后按`"0`可以插入上次y的内容，或者接其他寄存器插入寄存器的值

普通用户保存root权限的文件

一般发生于系统配置文件，修改后发现不是root无法保存。

输入

```
:w !sudo tee %
```

应用

给文件中每行前面加上4位数序号

```
:%s/^/\=printf("%03d", line('.'))
```