Linux下快捷键的使用

Linux 命令行(Bash)快捷键使用

```
Bash快捷键的规律：
    Ctrl开头的快捷键一般是针对字符的，而Alt开头的快捷键一般是针对词的。
参考文档：
    命令行艺术：https://linux.cn/article-5703-1.html
    Bash 下的快捷键：https://linux.cn/article-5660-1.html
    感叹号：bash 的历史扩展功能：https://linux.cn/article-5658-1.html
查看当前bash 的模式：默认的使用 emacs
    set -o
        emacs           on
        vi              off
    set -o vi       ## 可以设置为 vi 的模式
```

个人觉得最有用的

```
ctrl + r  (reverse) 输入单词搜索历史命令(多次按 ctrl + r 进行递归上翻， ctrl + j 确认搜索结果)
ctrl + u 删除 当前光标前面所有字符  相当于VIM里d^
ctrl + k 删除 当前光标后面所有字符  相当于VIM里d$
ctrl + w 删除 当前光标前一个单词(空格分割)

ctrl + y 恢复ctrl+u, ctrl + k,ctrl + w,上次执行时删除的字符，(粘贴)

alt + d (delete) 删除 光标所在位置的后单词 相当于VIM的 dw
alt + backspace 删除 光标所在位置前面的单词(非字母字符分割) 相当于VIM里db

ctrl + ? 撤消前一次输入
alt + r  (redo) 撤消前一次动作
alt + . 返回上一次执行命令或最后一个参数

ctrl + c (cancel) 撤销命令的输入，另起一行

工作目录切换：
    cd -
```

删除操作

```
ctrl + d (delete) 删除     光标所在位置上的字符  相当于VIM里x或者dl
ctrl + h (head) 删除  光标所在位置前的字符  相当于VIM里hx或者dh

ctrl + k 删除 光标后面所有字符     相当于VIM里d$
ctrl + u 删除 光标前面所有字符     相当于VIM里d^
ctrl + w 删除 光标前一个单词
ctrl + y 恢复ctrl+u上次执行时删除的字符
ctrl + ? 撤消前一次输入
alt + r 撤消前一次动作
alt + d 删除  光标所在位置的后单词 相当于VIM的 dw
```

移动操作

```
ctrl + a (ahead) 将光标移动到命令行开头 相当于VIM里 ^
ctrl + e (end) 将光标移动到命令行结尾处 相当于VIM里 $
ctrl + f (forward) 光标向后移动一个字符 相当于VIM里 l
ctrl + b (back) 光标向前移动一个字符 相当于VIM里 h
ctrl + 方向键左键 光标移动到前一个单词开头
ctrl + 方向键右键 光标移动到后一个单词结尾
ctrl + xx (exchange) 在上次光标所在字符和当前光标所在字符之间跳转
alt + f (forward) 跳到光标所在位置单词尾部
alt + b (back) 跳到光标所在位置单词开头
```

替换操作

```
ctrl + t (transfer) 将光标当前字符与前面一个字符替换
alt + t (transfer) 交换两个光标当前所处位置单词和光标前一个单词
alt + u (uppercase) 把光标当前位置单词变为大写（光标要在单词开头，才能完全转换）
alt + l (lowercase) 把光标当前位置单词变为小写（光标要在单词开头，才能完全转换）
alt + c (convert) 把光标当前位置单词头一个字母变为大写（光标要在所转换单词开头）
```

历史命令编辑

```
ctrl + p  (pre) 上翻历史命令
ctrl + n (next) 下翻历史命令
ctrl + r  (reverse) 输入单词搜索历史命令(多次按 ctrl + r 进行递归上翻， ctrl + j 确认搜索结果)
alt + p 输入字符查找与字符相接近的历史命令
alt + . 返回上一次执行命令或最后一个参数

alt + #  注释未输入完成的命令，历史命令可以进行查找
```

Bang(!) 命令

```
#! 的名字为什么叫Sha-Bang?
    Sha-Bang是Sharp和Bang的组合词。Sharp for #，Bang for ! 类似的情况是，C#通常被称为C Sharp
Sha-Bang(#!)所在行的作用是告知该脚本使用的是哪种命令解释器，并不是可有可无的。
```

```
!!  执行上一条命令
!blah   执行最近的以 blah 开头的命令，如 !ls
!blah:p 仅打印输出，而不执行
!?blan?:P 仅打印包含 blan 的历史命令
!$  上一条命令的最后一个参数，与 Alt + . 相同
!$:p    打印输出 !$ 的内容
!*  上一条命令的所有参数
!*:p    打印输出 !* 的内容
^blah   删除上一条命令中的 blah
^blah^foo   将上一条命令中的 blah 替换为 foo
^blah^foo^  将上一条命令中所有的 blah 都替换为 foo
^blah^foo^:p    将上一条命令中所有的 blah 都替换为 foo，打印出来，不执行

!n  重复历史中编号为n的命令—历史编号可以参看history命令
!-n 执行倒推的第 n 条命令 (!! 等同于 !-1)
```

其它操作

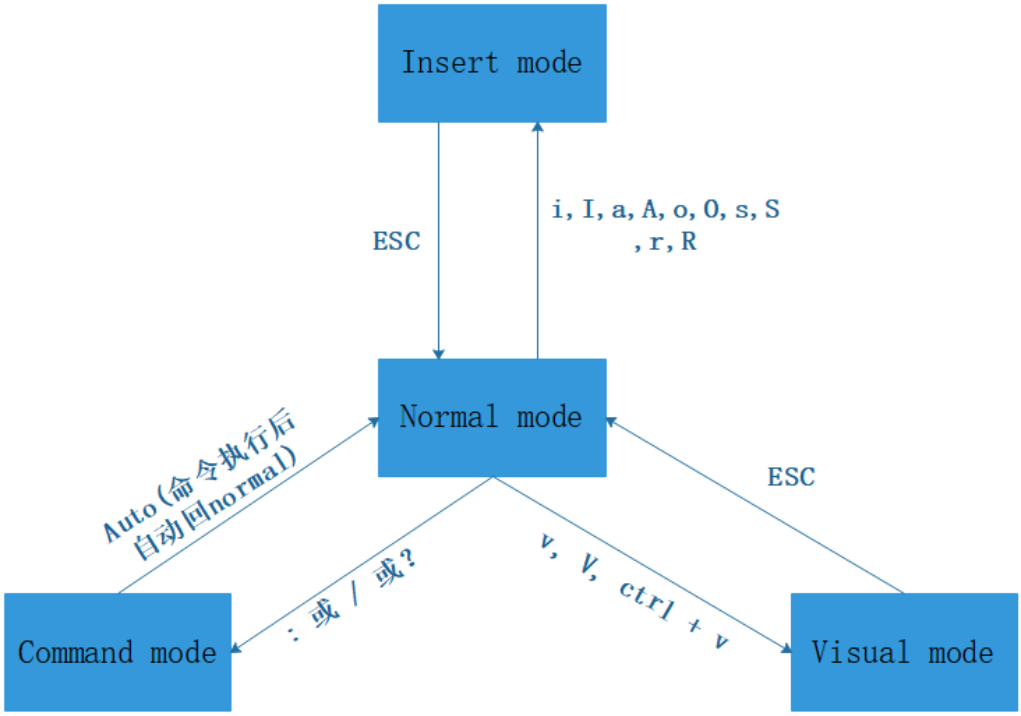
```
ctrl + s (stop, suspend) 停止终端输出，锁住终端
ctrl + q (quit) 解锁终端
ctrl + l 清屏相当于命令clear
ctrl + c 另起一行
ctrl + z （zombie ？？）放入后台
ctrl + d 相当于"EOF" (文件结尾: end of file)。它用于表示标准输入 (stdin) 的结束，退出终端
ctrl + i (interactive) 类似TAB健补全功能
ctrl + o 重复执行命令
alt + 数字键 操作的次数
```

VIM 快捷键的使用

VIM 的工作模式

```
Vim具有6种基本模式和5种派生模式:
    基本模式:
        普通模式(Normal mode):
            移动光标(行间跳转，行内跳转)，执行重复的命令，粘贴复制
        插入模式(Insert mode):
            大多数按键都会向文本缓冲中插入文本，编辑文本。
        命令行模式(Command line mode)或末行模式:
            可以输入会被解释成并执行的命令，普通模式通常按 **执行命令 (:键) ，搜索 (/和?键) ** 进入
        可视模式(Visual mode):
            与普通模式比较相似。但是移动命令会扩大高亮的文本区域。
        选择模式(Select mode):
            普通模式下，可以按 gh 进入。使用上下左右箭头进行选择，高亮文本，和可视模式不同的是，在这个模
式下，选择完了高亮区域后，敲任何按键就直接输入并替换选择的文本了。
        Ex模式(Ex mode):
            和命令行模式比较相似，使用 Q 进入，在使用:visual命令离开Ex模式前，可以一次执行多条命令。
```

模式之间的切换



进入VIM

```
# vim filename
$ vim practice_1.txt
```

```
$ vim               # 打开空文件
:e  filename         # 进入命令模式打开文件
```

```
$ vim +n filename   # 打开文件，将光标置于第n行行首
$ vim /pattern filename  # 打开文件，将光标置于第一个 pattern 匹配处
$ vim + filename    # 打开文件，将光标置于最后一行行首
$ vim filename1 filename2 ...   # 打开多个文件依次编辑
$ vim -r filename        # 上次vim 异常退出，对filename 进行recover
```

游标移动

普通模式下

命令	描述	说明
h (2h)	左移一个(两个)字符	
j (2j)	下移一行（两行）	
k (2k)	上移一行（两行）	
l (2l)	下移一行（两行）	
space	光标右移一个字符	
backspace	光标左移一个字符	
w/W	向右移动一个单词词首	小写以 非数字字母下划线 作为一个单词分割，大写以 空格作为一个单词的分割
b/B	向左移动一个单词词首	小写以 非数字字母下划线 作为一个单词分割，大写以 空格作为一个单词的分割
e/E	向右移动一个单词词尾	小写以 非数字字母下划线 作为一个单词分割，大写以 空格作为一个单词的分割
)	光标移至句尾	以 . (句点) 进行判断
(	光标移至句首	以 . (句点) 进行判断
}	光标移至段尾	以 空行 进行判断
{	光标移至段首	以 空行 进行判断
n$	光标移至第n行行尾	$ 移至当前行行尾
0	光标移至当前行行首	包含空字符
^	光标移至当前行行首	不包含空字符
g_	移动 到本行最后一个不是blank字符上	
gg	移动到文本第一行	
G	移动到文本最后一行	18G 移动到 第18行， dG 删除当前行至文本最后一行，ggdG 清空文本
ctrl + o	跳回到上一次的光标所在的位置	
f=	移动到当前行向右搜索的第一个字符 = 上去	
f+ ;	查找 + ， ；（分号）进行下次查询 的 + 处， ，（逗号）进行上次的查询的 + 处	
F=	移动到当前行向左搜索的第一个字符 = 上去	
t=	移动当前光标向右搜索第一个字符 = 之前	
T=	移动当前光标向左搜索第一个字符 = 之后	

命令	描述	说明
%	移动到当前光标所在字符与之匹配的括号或引号上	(), [] , {}, “”, ”
H	光标移动至屏幕顶行	操作相对于当前屏幕显示的文本内容，文本不滚动
M	光标移动至屏幕中间	
L	光标移动至屏幕最后行	
10|	把光标移动到第10列的位置	

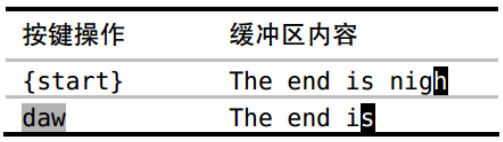
屏幕滚动

命令	描述	说明
ctrl + u （up）	向文件首翻半屏	文本会进行滚动
ctrl + d （down）	向文件尾翻半屏	
ctrl + f (forward)	向文件首翻一屏	
ctrl + b (back)	向文件尾翻一屏	
zz	光标当前行至屏幕中间	

插入文本

命令	描述	说明
i/I	在光标前插入，在行首插入	
a/A	在光标后插入，在行尾插入	
o/O	在当前行下一行，在当前行上一行插入一行	
s (3s)	当前光标处，以输入文本替换	s(删除当前光标所在的字符)，并进入插入模式；3s(删除当前光标与之后的2个字符)，并进入插入模式
S (3S)	删除光标所在行，并进入插入模式	
r/R	替换光标所在的字符，替换光标所在之后的字符连续替换	
cw/CW (2cw/2CW)	删除光标所在处之后的一个(n个)单词，进入插入模式	
cb/CB	删除光标所在处之前的一个(n个)单词，进入插入模式	

删除命令

命令	描述	说明
x/X	x删除光标出的字符，X 删除光标前的一个字符	
dw/d2w	删除一个（2个）单词，至下一个单词的开始位置	
de	删除当前的光标所在单词的末尾	
d0/d^	删除当前光标所在单词的行首（0与 ^ 是非 空字符的区别）	
d$ (D)	删除光标所在的位置到行尾	
dd (2dd)	输出一行，2行	
:n1, n2 d	删除n1 行到n2 行内容	
d2h (d2l)	向左、向右删除n 个字符	
d2j (d2k)	向下、向上删除n 行	
c$（C)	替换本行到末尾	
cc	替换整行	
J，v + J	合并行，将当前光标所在行与下一行合并为一行(可视模式下，选中多行进行合并)	
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	
dt", df"	删除本行当前光标到第一次找到 " 之间的字符	
da{ , da(, da[	光标移动到 括号的前半部分，删除多行 {} 所包含的内容	

Insert mode 下的删除

命令	描述	说明
ctrl + h	删除前一个字符（同退格键）	与bash 命令行中的快捷方式同
ctrl + w	删除前一个单词	
ctrl + u	删除至行首	

复制粘贴

命令	描述	说明
p/P	向当前光标后粘贴字符或向下一行粘贴，向当前光标前粘贴字符或向上一行粘贴	x, d, 删除的字符，或 y 复制的字符，或 v 选择的
yw (y2w)	向后复制一个或2个字符 至缓冲区	
yy (2yy)	复制光标所在行货光标所在行及下一行 至 缓冲区	
y2h (y2l)	左右复制字符	
y2k (y2j)	上下复制行	
y$	复制光标所在位置到行的末尾的字符	
:m,ny	m 行的内容复制到n 行	
yaw和yas	复制一个词和一个句子，即使光标不在词首或句首也我有关系	

插入模式下进行粘贴

命令	描述	说明
ctrl + r 0	普通模式下进行的赋值操作 如：yw ,的赋值文本默认的会放入 0 这个最近的寄存器中，使用 ctrl + r 会在当前的光标处显示 “ 按 0 ， 进行调用	按键操作 缓冲区内容 yt, Practical Vim, by Drew Neil Read Drew Neil's jA␣ Practical Vim, by Drew Neil Read Drew Neil's <C-r>0 Practical Vim, by Drew Neil Read Drew Neil's Practical Vim .<Esc> Practical Vim, by Drew Neil Read Drew Neil's Practical Vim.

插入模式插入特殊编码字符

命令	描述	说明
ctrl +v u1234	插入utf-8 字符 ก	u1234 为字符编码
ga	光标移动到字符上查看字符的编码	
ctrl + k 12	二合字母的输入， ½	

大小写转换

命令	描述	说明
~ （3~）	光标所在处的字母进行大小写转换	
g~	当前光标所在的行进行大小写的转换	
gul （gUl）	当前光标所在字符转换为小写	
guh (gUh)	当前光标所在字符前的字符转换为小写	
guj (gUj)	当前光标所在行及下一行字符转换为小写	
guk (.gUk)	当前光标所在行及上一行字符转换为小写	
gUU	当前行所有的字符转换为大写	
guu	当前行所有的字符转换为小写	
ggguG	整篇文章小写	
gggUG	整篇文章大写	
guw gue(gUw/gUe) gu5w	转换单个单词的大小写	
gU0 (gU$)	从光标所在位置到行首，都变为大写；从光标所在位置到行尾，都变为大写	
gUG（gU1G)	从光标所在位置到文章最后一个字符，都变为大写；从光标所在位置到文章第一个字符，都变为大写	

撤销与反撤销

命令	描述	说明
u	撤销前一次的输入	
U	撤销该行的所有的操作	
ctrl + r	反撤销，将上一次撤销的内容恢复	
:earlier 4m	回到 4min 中之前的文档状态	
:later 20s	向前推进20s 的文档状态	

搜索与替换

命令	描述	说明
/pattern	从光标开始处，向文本末尾开始搜索 pattern, n 搜索下一个，N 搜索上一个	/haha, cw hehe, n, .
?pattern	从光标开始处，向文本开头开始搜索 pattern, n 搜索上一个，N 搜索下一个	
* 与 #	匹配光标当前所在的单词	*是下一个，#是上一个
:s/p1/p2/	替换当前行中的第一个搜索的 p1 替换为 p2	
:s/p1/p2/g	当前行中所有的p1 替换为p2	
:20,200s/p1/p2/g	将20 到200 行的 所有p1 替换为p2	
:%s/p1/p2/g :1,$s/p1/p2/g	将文件中所有的p1替换为p2	
y2/foo	普通模式下，复制当前行当前光标处到第二个foo 之前的字符	

VIM 的正则表达式

使用正在表达式的命令:

1. 最常见的就是 /（搜索）命令，如:
   /正则表达式
2. 另一个是 :s （替换）命令，如:
   :s/正则表达式/替换字符串/选项

元字符	说明
.	匹配任意一个字符
[abc]	匹配方括号中的任意一个字符。可以使用-表示字符范围， 如[a-z0-9]匹 配小写字母和阿拉伯数字
[^abc]	在方括号内开头使用^符号，表示匹配除方括号中字符之外的任意字符
\d	匹配阿拉伯数字，等同于[0-9]
\D	匹配阿拉伯数字之外的任意字符，等同于[^0-9]
\x	匹配十六进制数字，等同于[0-9A-Fa-f]
\X	匹配十六进制数字之外的任意字符，等同于[^0-9A-Fa-f]
\w	匹配单词字母，等同于[0-9A-Za-z_]
\W	匹配单词字母之外的任意字符，等同于[^0-9A-Za-z_]
\t	匹配<TAB>字符
\s	匹配空白字符，等同于[\t]
\S	匹配非空白字符，等同于[^ \t]

表示数量的元字符

元字符	说明
*	匹配0-任意个
+	匹配1-任意个
\?	匹配0-1个
{n,m}	匹配n-m个
{n}	匹配n个
{n,}	匹配n-任意个
{,m}	匹配0-m个

表示位置的元字符

元字符	说明
$	匹配行尾
^	匹配行首
<	匹配单词词首
>	匹配单词词尾

需要转义的元字符

转义元字符	真实匹配字符
*	匹配 * 字符。
\.	匹配 . 字符
\/	匹配 / 字符
\\	匹配 \ 字符
\[	匹配 [字符

vim 正则中的函数式操作

```
:s/替换字符串/\=函数式
在函数式中可以使用 submatch(1)、submatch(2) 等来引用 \1、\2 等的内容，而submatch(0)可以引用匹配的整个内容。
```

示例	描述	说明
:%s/<id>/\=line(".")	将各行的 id 字符串替换为行号	line() 函数取得行号，“.” 当前行
:%s/(^\w+\)/\=(line(".")-3).".".submatch(1)	将文本的每行开头的单词替换为 (行号-3).单词 的，如第11行的 word 替换成 1.word	(line(".")-3) 为 当前行号 -3 . 为字符字符连接操作 “.” 单纯的英文句点 submatch(1) 为前匹配项的第一个括号中的内容
:%s/(^\w+)/\="hahaha"."+++"."hehehe".submatch(1)	将文本的每一行首单词(如：word)，进行替换为 hahaha+++heheheword	验证 . （句点）表示 字符的连接

示例

示例	说明
/char\s+\w*;	查找所有以char开头，之后是一个以上的空白， 最后是一个标识符是分号
/\d\d:\d\d:\d\d	查找如 17:37:01 格式的时间字符串
:g/^$/d	删除文本中的空行
:s/<four>/4/g	将所有的four替换成4，但是如fourteen中的four不替换

正则分组

示例	说明
/(a+)[^a]+\1	查找开头和结尾处a的个数相同的字符串，中间字符为非a，如：如 aabbbaa，aaacccaaa，但是不匹配 abbbaa
:s/(http:\/\/[-a-z._~+%\/]+)/<a href="\1">\1<\/a>/	将URLhttp://url 替换为http://url
:s/(\w+)\s+(\w+)/\2\t\1	将 data1 data2 修改为 data2 data1

重复执行命令

命令	描述	说明
. (英文句点)	重复执行上次的命令	
j.	下一行重复执行上一次的命令	按键操作 缓冲区内容 {start} var foo = "method("+argument1+","+argument2+")"; f+ var foo = "method("+argument1+","+argument2+")"; 按键操作 缓冲区内容 s␣+␣<Esc> var foo = "method(" +argument1+","+argument2+")"; ; var foo = "method(" + argument1+","+argument2+")"; . var foo = "method(" + argument1 +","+argument2+")"; ;. var foo = "method(" + argument1 + "," +argument2+")"; ;. var foo = "method(" + argument1 + "," + argument2 +")";
&	:s/target/replacement 重复进行匹配替换	
100<command>	重复执行后面的命令100次	

文件操作

命令	描述	说明
:q	退出vim	
:q!	不保存修改强制退出	
:e filename	打开并切换至其他的文件，进行编辑	
:e#	使用:e filename 打开新的文件后，切换到上个编辑的文件	
:ls	打开正在编辑的文件	
:b 2.txt	（或者编号）可以直接进入文件2.txt编辑	
:bd 2.txt	或者编号）可以删除以前编辑过的列表中的文件项目	
:f	显示正在编辑的文件名	
:f new.txt	改变正在编辑的文件名字为new.txt（fork 了一份）	
:w	保存当前的文件	
:w newfile :saveas newfile	另存当前的文件为 newfile	
:n1, n2 w temp.txt	将n1与n2 的内容写入到临时文件temp.txt 中	
:bdelete N1 N2 N3 :N,M bdelete	每次打开一个文件时，Vim 就会创建一个新的缓冲区，删除一个缓冲区并不会影响缓冲区所关联的文件，而只是简单地把该文件	
:wq，:x 或 ZZ	保存并退出	:wq 每次保存会更新文件的mtime,及时文件没有任何修改；而:x 与 ZZ 不会
:!command	在编辑命令的时候，临时执行shell 命令	:1，12!sort -r -n -k4.1,5 :1,$!sort -k 4 -t"." -n :r !date +%s
:r filename	读取filename 的内容，写入当前文件的当前行下	
:r!command	将命令的执行结果写入此文件的当前行下	
:bn，:bp，2 + ctrl + ^	使用 vim filename1 filename2 打开多个文件时，来进行文件之间的切换;	现在在1文件进行编辑，使用:w保存后，按 2 执行 ctrl + ^ 切换到第二个文件

多文件tab 标签页

使用 vim -p filename1 filename2

命令	描述	说明
: tabe[dit] filename	在新的标签页打开文件编辑	
:tabnew	新打开一个空的文件，在新的标签页	
:tabs[how]	显示标签已打开的所有的文件	
:tabc[lose] :tabc 3	关闭当前的标签 关闭打开的第三个标签	
:tabo[nly]	关闭除当前标签外的其他标签	
:tabdo cmd	所有的标签页执行命令	
:tabn[ext] 或 gt 或 ctrl + w T :tabn 3	切换标签页到下一个 切换标签页到第三个标签页	
:tabp[revious] 或 gT :tabp 2	切换标签页到上一个 切换标签页到第三个标签页	
:tabfirst	移动到标签页首	
:tablast	移动到最后的一个标签页	
:tabm[ove] 2 :tabm -2	移动当前的标签到相对于此标签的向后2个标签页 移动当前的标签页到相对于此标签页向前的 2个标签页	

两个文件比较 （窗口操作）

使用vimdiff 命令亦可进入

命令	描述	说明
:diffthis	vim -O filename1 filename2 打开两个文件后，分别输入:diffthis 后进入比较	
:vert[ical] diffsplit filename2	已经打开了文件file1，再打开另一个文件file2进行比较	
[c	跳到上一个不同点	
]c	跳到上一个不同点	
:diffput 或 :dp	把当前光标所在文本的不同当 put 给另一个文档	
:diffget	把当前光标所在文本的不同当 get 到当前文档	

书签

命令	描述	说明
m[a-z]	在文本中做标记，以a-z 的任意一个字符	
`a	反引号 + 标签名，跳转到a 定义的标签处	
ctrl + t	标签退栈	
:marks	查看所有的标签的定义	

可视模式

可以模式与选择模式之间的来回切换 ctrl +g

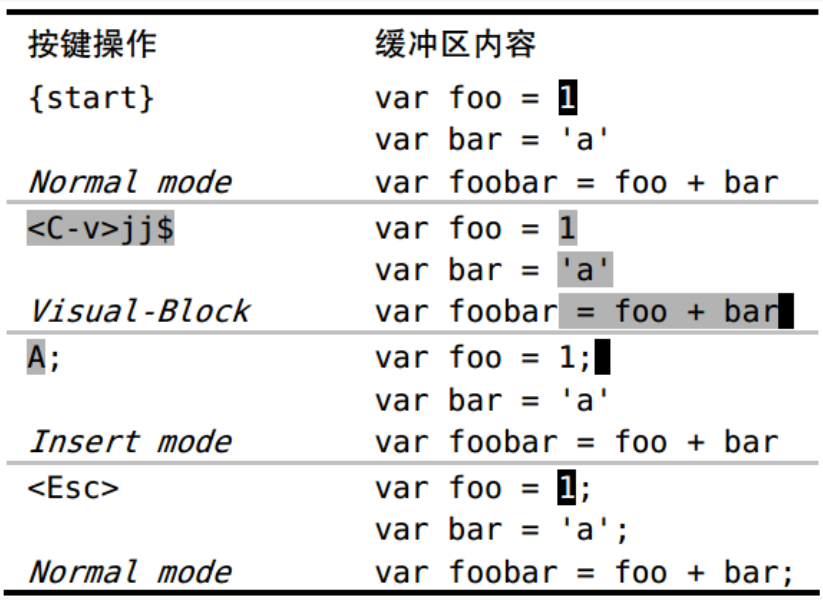
命令	描述	说明
v	进入字符的visual，类似鼠标拖选字符	
V	进入行的visual，一行行进行选取	
ctrl + v	进入列的visual，以多行同一列的字符为选取单位	o与O可以改变选取的边界，o更变选取高度，O改变选取宽度 按键操作 缓冲区内容 {start} Select from here to here. vbb Select from here to here. o Select from here to here. e Select from here to here.
gv	重选上次的选择可视区	
VG :'<,'>p	对选中的区域执行操作	

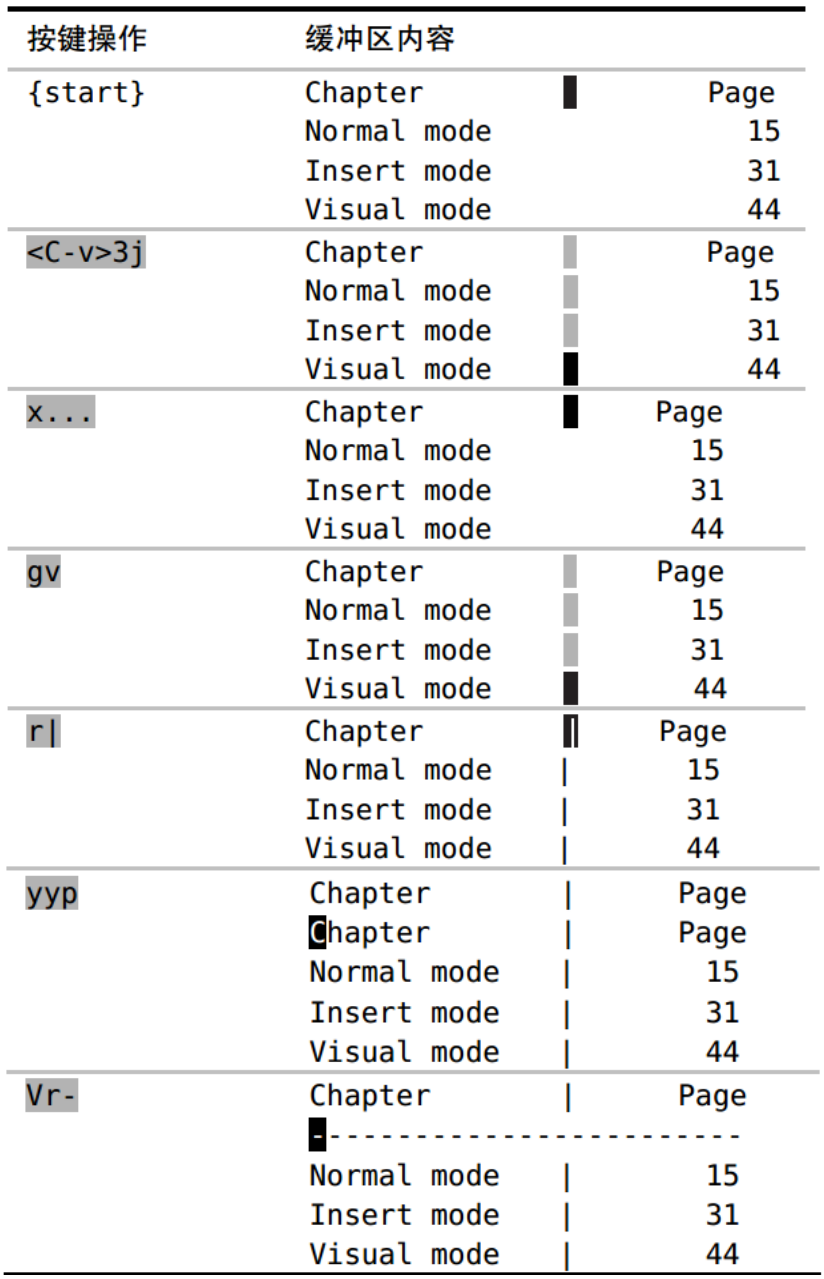
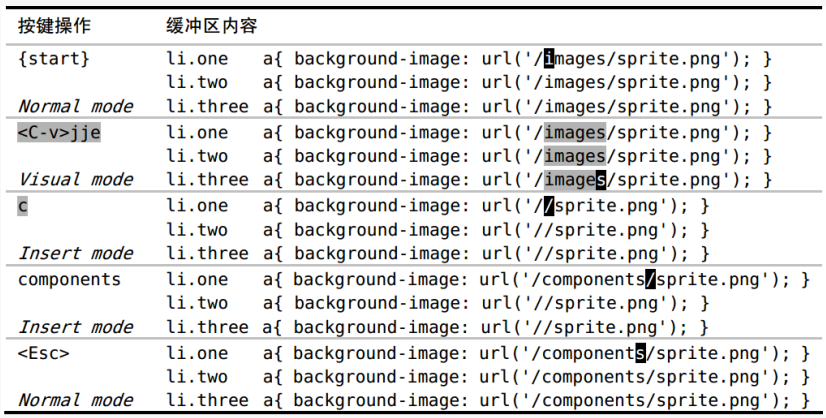
A;
jVG
:'<,'>normal .
或
:%normal A;
	在普通模式下重复上次的操作	按键操作 缓冲区内容
		jVG
		var foo = 1;
		var bar = 'a'
		var baz = 'z'
		var foobar = foo + bar
		var foobarbaz = foo + bar + baz
		:'<,'>normal .
		var foo = 1;
		var bar = 'a';
		var baz = 'z';
		var foobar = foo + bar;
		var foobarbaz = foo + bar + baz;

u/U	选中的字符大小写转换	
>>	向右缩进	
<<	向左缩进	
=	自动缩进	
gg=G	全文本自动缩进	
示例1	行首插入 # 注释	光标移至开始行首位置，ctrl +v 进入列操作模式，j 下选行数，shift + i, 跳到行首插入 #，ESC 退出 列操作模式，完成

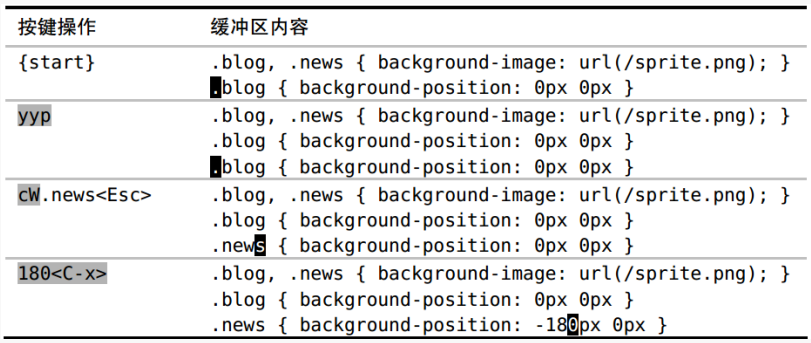
命令	描述	说明

示例2	你有一个字符串 `(map (+) ("foo"))` . 而光标键在第一个 o 的位置。	`vi"` → 会选择 `foo`
		`va"` → 会选择 `"foo"`
		`vi)` → 会选择 `"foo"`
		`va)` → 会选择 `("foo")`
		`v2i)` → 会选择 `map (+) ("foo")`
		`v2a)` → 会选择 `(map (+) ("foo"))`

示例3	多行行尾插入字符	

命令	描述	说明
示例4	列间增加竖线	
示例5	列式编辑，images 替换为 components	

数字计算

命令	描述	说明
ctrl +a	光标放置数字上，ctrl +a 进行加1	
10 ctrl +a	当前的光标的数值加上 10	
ctrl +x	当前的光标依次减1	
10 ctrl + x	当前的光标的数值减去 10	
i , ctrl + r , = , 134523+2349	表达式模式，将计算结果输入到文本中	

排版

命令	描述	说明
<<	向左缩进	
>>	向右缩进	
>G	从当前行到文本末尾进行缩进	
:ce	本行文字居中	
:le	本行文字左对齐	
:ri	本行文字右对齐	

Ex 命令模式

进入Ex 模式，VIM普通模式按 Q 进入（或vim -e filename）或bash 下执行 ex filename， 输入 visual 或 vi 退出 Ex 模式，回到normal 模式。

ex命令由**行地址**（行号）和**命令**组成，它们都以回车键结束。

Ex 模式的常用命令：

d (delete) 删除行；m (move) 移动内容；co (copy) copy 内容; **t**　复制行，和**co**同义

表 5-1　操作缓冲区文本的 Ex 命令

命令	用途
:[range]delete [x]	删除指定范围内的行[到寄存器 x 中]
:[range]yank [x]	复制指定范围的行[到寄存器 x 中]
:[line]put [x]	在指定行后粘贴寄存器 x 中的内容
:[range]copy {address}	把指定范围内的行拷贝到 {address} 所指定的行之下
:[range]move {address}	把指定范围内的行移动到 {address} 所指定的行之下
:[range]join	连接指定范围内的行
:[range]normal {commands}	对指定范围内的每一行执行普通模式命令 {commands}
:[range]substitute/{pattern}/{string}/[flags]	把指定范围内出现{pattern}的地方替换为{string}
:[range]global/{pattern}/[cmd]	对指定范围内匹配{pattern}的所有行，在其上执行 Ex 命令{cmd}

命令	描述	说明
ctrl + d	进行命令的补全（TAB 进行循环选择）	
:10	切换到第10行	
:20 p	打印第 20 行的内容	
:1, 20 p	打印1到20 行的内容	
:n1, n2 d	将n1到n2 的内容删除	
:n1, n2 co n3	将n1到n2 的内容拷贝到n3之后	
:n1, n2 m n3	将n1到n2 的内容移动到n3之后	
:set nu/ :set nonu	显示行号/不显示行号	
:=	查看文件的所有行	
: .=	重看文档的当前行	
: /pattern/=	显示第一个与模式pattern匹配的行号	
:s/hello/hehe	当前行的hello 替换为 hehe	
.(句点)/ $ / %	行地址符：当前行/ 最后一行 / 文件的全部行，等同1,$	
:@:	重复执行上一次的Ex 命令	
: .，$ d	删除从当前行到文件末尾	
: 20, . m $	把20 ~ 当前行的文本移动到文件末尾	
: % d	删除文件中的所有行	
: % t $	复制所有行并把它们粘贴到文件的尾部	
: . , . + 20 d	删除从当前行到当前行以下20行	
: 226 , $ m .-2	把226行到文件末尾的文本移动到当前行的上面两行的后面	
: .,.+20#	带行号显示当前行到第20行的内容	
: -, + t 0	复制三行（当前行，前一行，后一行），并粘贴到文件开头 (在+或-后面没有数字，那么就等价于+1和-1)	
: /pattern/ d	删除下一个包含pattern的行	
: /pattern/+ d	删除下一个包含pattern的行的下一行 （也可使用+1来代替)	

命令	描述	说明
：/pattern1/,/pattern2/d	删除第一个包含pattern1的行与第一个包含pattern2的行之间的所有行	
：. , /pattern/ m 23	从当前行到第一个包行pattern的行之间的文本移动到23行后面	
：100 ; +5 p	就会显示100～105行	当使用“分号”代替“逗号”时，就会将第一个行地址当成“当前行”
：/patter/ ; +10 p	显示下一个包含pattern的行和它下面的10行	
：g /pattern ：g /pattern/ p ：g!/pattern/nu ：20, 40g/pattern/p	寻找（移动到）模式pattern在文件中最后出现的位置 寻找并显示文件中所有包含模式pattern的行 找并显示文件中所有不包行模式pattern的行，并显示这些行号 寻找并显示第20到40行之间所有包含模式pattern的行	
：1 , 3d ｜ s/thier/their/	把第1行到第3行删除，然后在当前行（该行是调用ex提示符以前的第4行）进行替换	
：1 , 5 m 10 ｜ g/pattern/nu	把第1行到第5行移动到第10行后面，然后显示所有包含模式pattern的行	
：20 , $w newfile	把第20行到文件末尾文本保存到新文件newfile中	
：. , 500 w newfile	从当前行到500行文本保存到新文件newfile中	
：30 , $w >> file	把30行到文件末尾添加到文件file的末尾	
：r filename	把filename文件的内容插入当前编辑文本的光标位置后面一行	
：$r filename ：0r filename	把读取的文件filename内容加到当前编辑文件的末尾 (开头)	
：/pattern/r filename	将读取的文件filename内容加到当前文件包行模式/pattern的行的后面	

宏操作

命令	描述	说明
q[a-z]	如，qa 开始记录宏 为a 的操作，使用 q 中值宏的记录	:qA 进行追加录制， :reg a 查看宏的内容
:reg a	查看寄存器a 的信息	
@a	在当前的光标出执行 定义的宏 a	
@@	执行最近的一个宏定义	
示例	从1 开始输入到 1000，每行一个数字	第一行输入 数字1，光标放置在1上 `qaYp<C-a>q` → `qa` 开始录制 `Yp` 复制行. `<C-a>` 增加1. `q` 停止录制. 100@@ 执行
示例	文本行首插入行号	首行开头插入1，光标放置在1上 `qa` 开始录制 `^` 首行 `yw` 复制上一行数字 `j^P^` 到行首粘贴上一行的数字 `ctrl + a` 数字加1 `q` 结束录制 `@@` 或 `@a`

快捷键绑定

命令	描述	说明
:map <C-b> <C-v> I # <Esc>	定义ctrl +b 执行 ctrl +v I # ESC 这些组合键	

别名使用

命令	描述	说明
:ab alias_string origin_string	命令模式定义出现 alias_string字符，使用 origin_string进行替换	
示例	邮箱替换	:ab email xue_yongbo@163.com

变量补全（自动提示）

命令	描述	说明
ctrl + n	下一个提示	前提是之前的文本中含有你的，当前输入的匹配字符
ctrl + p	上一个提示	

窗口操作

命令	描述	说明
:split （:new） （ctrl + w s)	为当前的编辑文件横向分出一个窗口	
:split filename	打开 filename 为其横向分配一个窗口	
:5split filename1	打开filename1 为其分配一个显示5行的窗口	
:vsplit filename1 (ctrl + w v)	打开filename1 为其分配一个竖向的窗口	
:diffthis	vim -O filename1 filename2 打开两个文件后，分别输入:diffthis 后进入比较	
:vert[ical] diffsplit filename2	已经打开了文件file1，再打开另一个文件file2进行比较	
[c	跳到上一个不同点	
]c	跳到上一个不同点	
:diffput 或 :dp	把当前光标所在文本的不同当 put 给另一个文档	
:diffget	把当前光标所在文本的不同当 get 到当前文档	
vim -o a b c	打开 a,b,c 横向切分窗口	
vim -O a b c	打开a,b,c 竖向切分窗口	
ctrl +w + j	跳到当前窗口下一个的横向窗口	
ctrl + w + k	跳到当前窗口的上一个横向窗口	
ctrl + w +l 或 ctrl + w + h	左右之间的竖向窗口进行切换	
Ctrl+w J	将当前视窗移至下面	
Ctrl+w K	将当前视窗移至上面	
Ctrl+w H	将当前视窗移至左边	
Ctrl+w L	将当前视窗移至右边	
ctrl + ww	各个窗口之间的循环切换	
ctrl + w + _	竖直方向最大化	
ctrl + w + |	垂直方向最大化	
<ctrl + w> - 或 <ctrl + w > +	增加减小尺寸(高度)	
:wa	保存所有的窗口编辑文件	

命令	描述	说明
:qa	关闭所有的窗口	

VIM 常用配置

命令	描述	说明
:nu (number)	显示光标所在行行号	
:set all	显示vim 的所有配置参数	
:set option?	显示option的设定值	:set maxmemtot? 显示 maxmemtot=501716
:set nu	显示文本所有的行号	
:set nonu	取消所有行号的显示	
:set hlsearch	/ 或 ？ 搜索时高亮显示，配置字符	
:set nohl	取消搜索时的高亮显示	
:set ic (ignorecase)	搜索忽略字母大小写	
:set noic	取消忽略字母大小写	
:set readonly	设置文件为只读模式	
:syntax on	设置语法高亮	
:set shiftwidth=4	设置缩进空格数为4	
:set autoindent	设置自动缩进	
:set spell	拼写检查	
:set showmatch	括号匹配高亮	
:set backup(bk)	设置自动备份	
set tabpagemax=15	设置最多打开的标签的数量	