vi 使用手册(zt)

进入 vi 的命令

vi filename :打开或新建文件，并将光标置于第一行首

vi +n filename ：打开文件，并将光标置于第 n 行首

vi + filename ：打开文件，并将光标置于最后一行首

vi +/pattern filename：打开文件，并将光标置于第一个与 pattern 匹配的串处

vi -r filename ：在上次正用 vi 编辑时发生系统崩溃，恢复 filename

vi filename....filename ：打开多个文件，依次编辑

移动光标类命令

h ：光标左移一个字符

l ：光标右移一个字符

space：光标右移一个字符

Backspace：光标左移一个字符

k 或 Ctrl+p：光标上移一行

j 或 Ctrl+n ：光标下移一行

Enter ：光标下移一行

w 或 W ：光标右移一个字至字首

b 或 B ：光标左移一个字至字首

e 或 E ：光标右移一个字 j 至字尾

) ：光标移至句尾

(：光标移至句首

}：光标移至段落开头

{：光标移至段落结尾

nG：光标移至第 n 行首

n+：光标下移 n 行

n-：光标上移 n 行

n$：光标移至第 n 行尾

H ：光标移至屏幕顶行

M ：光标移至屏幕中间行

L ：光标移至屏幕最后行

0：（注意是数字零）光标移至当前行首

$：光标移至当前行尾

屏幕翻滚类命令

Ctrl+u：向文件首翻半屏

Ctrl+d：向文件尾翻半屏

Ctrl+f：向文件尾翻一屏

Ctrl＋b；向文件首翻一屏

nz：将第 n 行滚至屏幕顶部，不指定 n 时将当前行滚至屏幕顶部。

插入文本类命令

i ：在光标前

I ：在当前行首

a：光标后

A：在当前行尾

o：在当前行之下新开一行

O：在当前行之上新开一行

r：替换当前字符

R：替换当前字符及其后的字符，直至按 ESC 键

s：从当前光标位置处开始，以输入的文本替代指定数目的字符

S：删除指定数目的行，并以所输入文本代替之

ncw 或 nCW：修改指定数目的字

nCC：修改指定数目的行

删除命令

ndw 或 ndW：删除光标处开始及其后的 n-1 个字

do：删至行首

d$：删至行尾

ndd：删除当前行及其后 n-1 行

x 或 X：删除一个字符，x 删除光标后的，而 X 删除光标前的

Ctrl+u：删除输入方式下所输入的文本

搜索及替换命令 ：

/pattern：从光标开始处向文件尾搜索 pattern

?pattern：从光标开始处向文件首搜索 pattern

n：在同一方向重复上一次搜索命令

N：在反方向上重复上一次搜索命令

：s/p1/p2/g：将当前行中所有 p1 均用 p2 替代

：n1,n2s/p1/p2/g：将第 n1 至 n2 行中所有 p1 均用 p2 替代

：g/p1/s//p2/g：将文件中所有 p1 均用 p2 替换

选项设置

all：列出所有选项设置情况

term：设置终端类型

ignorance：在搜索中忽略大小写

list：显示制表位(Ctrl+I)和行尾标志（$)

number：显示行号

report：显示由面向行的命令修改过的数目

terse：显示简短的警告信息

warn：在转到别的文件时若没保存当前文件则显示 NO write 信息

nomagic：允许在搜索模式中，使用前面不带“\”的特殊字符

nowrapscan：禁止 vi 在搜索到达文件两端时，又从另一端开始

mesg：允许 vi 显示其他用户用 write 写到自己终端上的信息

最后行方式命令

：n1,n2 co n3：将 n1 行到 n2 行之间的内容拷贝到第 n3 行下

：n1,n2 m n3：将 n1 行到 n2 行之间的内容移至到第 n3 行下

：n1,n2 d ：将 n1 行到 n2 行之间的内容删除

：w ：保存当前文件

：e filename：打开文件 filename 进行编辑

：x：保存当前文件并退出

：q：退出 vi

：q!：不保存文件并退出 vi

：!command：执行 shell 命令 command

：n1,n2 w!command：将文件中 n1 行至 n2 行的内容作为 command 的输入并执行之，若不指

定 n1，n2，则表示将整个文件内容作为 command 的输入

：r!command：将命令 command 的输出结果放到当前行

从 shell 中启动可视化编辑器 ：

vi filename 指示 shell 启动 vi 编辑器，并将参数 filename 传给它。如果当前目前中存在该文件，则 vi 编辑器将它解释为要打开的文件；如果没有该文件，则 vi 编译器创建新文件

vi file1 file2 file3 shell 传递 3 个参数给 vi，vi 将它们解释为要打开的文件。可以使用:w 命令保存文件，使用:n 命令访问下一个文件

vi +# filename 打开文件，并将光标移到指定的行。例如，命令 vi +100 records 从第 100 行开始编辑文件 records

vi +/the filename 打开文件，并将光标移动包含有目标字符串的行。例如，命令 vi +/Jason friends 从第 1 个含有字符串 Jason 的行开始编辑文件 friends

view filename 打开文件进行编辑，但是拒绝保存对文件的修改，除非使用 w!命令

光标移动命令 ：

hjkl 将光标分别向左、下、上、右移动一个字符

0(零) 将光标移到当前行的行首

^(脱字符) 同 0 一样将光标移到当前行的行首

$ 将光标移到当前行的行末

##G 将光标移到 G 前面的数字指定的行。例如，42G 将光标移到文件的第 42 行

G 将光标移到文件的最后一行

w 将光标向前移到下一个单词的首字母

e 将光标向前移到下一个单词的最后一个字母

b 将光标向后移到上一个单词首字母

- 将光标定位到上一行的行首

+ 将光标定位到下一行的行首

12| 将光标定位到当前行的第 12 列

L 将光标定位到屏幕的最下面一行

M 将光标定位到屏幕中间的一行

H 将光标定位到屏幕的最上面一行

'' 两个单引号将光标移到它的先前的位置

光标定位命令（上下文的）：

fb 将光标向前移到当前行上的下一个字母 b（或者其他的任意指定的字符）

Fb 将光标向后移到当前行上的上一个字母 b（或指定的字符）

t# 将光标移到当前行上字符＃的第 1 个实例的右侧。例如，命令 tM 将光标移到当前行上第 1 个 M 的右侧

T# 在当前行上向左移动光标，将它移到字符＃的第 1 个实例的前一字符

/word 将光标向前移到单词 word 的下一个实例

?word 将光标向后移到单词 word 的上一个实例

n 将光标移到前面命令/word 或?word 中指定模式的下一个实例

显示调整命令:

Ctrl+D 显示文件中的下半屏文本

Ctrl+U 显示文件中的上半屏文本

Ctrl+F 显示文件中的下一屏文本

Ctrl+B 显示文件中的上一屏文本 :

设置显示选项 :

:set number 将行号作为屏幕显示的一部分，但是行号并不是文件的一部分。它的缩写形式为:set nu

:set nonumber 清除屏幕上的行号。也可以使用缩写形式:set nonu

:set showmode 在屏幕的右下角显示追加模式信息

:set list 在每行的行末显示美元符号，并用 Ctrl+I 表示制表符

:set showmatch 在输入）或］时，将光标移到与之匹配的（或[

:set window=value 定义屏幕上显示的文本行的行数

:set autoindent 自动缩进。也可以使用缩写形式:set ai

:set tabstop=value 设置显示制表符的空格字符个数。也可以使用缩写形式 ts=value

:set wrapmargin=value 设置显示器的右页边。当输入进入所设置的页边时，编辑器自动回车换行

:set ignorecase 指示编辑器搜索字符串，并忽略目标中字母的大小写

:set 显示设置的所有选项

:set all 显示所有可以设置的选项

文本删除命令 ：

dd 删除当前光标所有的文本行

#dd 删除＃行文本

dw 从文本中删除一个单词

#dw 从文本中删除＃个单词

x 删除光标所在的一个字符

#x 从文本中删除＃个字符

D 删除当前行上光标后面的部分

:#,#d 例如，:12,37d 将删除第 12～37 行之间的所有文本，包括第 12 和 37 行

撤销命令 ：

u 撤销。恢复最近一次的文本修改操作，即使已经移动了光标。在 Linux 系统中，再次使用撤销命令将恢复更前一次的文本修改操作。

在 BSD 的 vi 中，第 2 次撤销操作将撤销第一次撤销操作，恢复第 1 次撤销前修改的文本

:redo 在 Linux 系统中，取消撤销操作恢复文本修改。在标准的 UNIX 系统中，第 2 个 u 命令取消第 1 个 u 命令，结果就是一个“redo”

U 如果在修改后还没有将光标移出当前行，则可以撤销对当前行进行的所有的文本修改

向文本中添加文本 ：

a(小写) 从光标的右侧开始插入文本

A(大写) 从当前行的行末开始添加文本

i(小写) 从光标的左侧开始插入文本

I(大写) 从当前行的行首插入文本

o(小写) 在光标的下面打开（或插入）一个新行

O(大写) 在光标的上面打开一个新行

:#r filename 例如，:8r report.old 读取文件 report.old，并将它的内容放到当前文件的第 8 行之后

Esc 无论使用什么命令进入了追加/插入模式，都可以通过按 Esc 键离开追加模式返回到 vi 的命令模式

Ctrl+V 允许输入控制字符。按 Ctrl+V 键后再按回车键将把 Ctrl+M 插入到文件中

在文件中修改文本 ：

cw 仅仅修改光标处的单词（删除单词，然后进入追加模式中，以在被删除单词的位置添加文本）

s(小写) 替换单个字符

S(大写) 替换整行文本

cc 替换整行文本(同 S)

r 用输入的下一个字符替代当前光标处的字符，并自动返回到命令模式

R 将编辑器放到覆盖模式，用输入的字符来逐个替换光标处的字符

C(大写) 修改行上从光标到行末之间的文本

ct# 修改行上从光标到前向第 1 个目标字符之间的文本。例如 ctY 将删除当前行上从光标到向前第 1 个字符 Y 之间的所有文本，并进入追加模式以在删除文本的位置添加文本

cf# 修改行上从光标到前向第 1 个目标字符之间的文本（包括目标字符）。例如 cfY 将删除当前行上从光标到向前第 1 个字符 Y 之间（包括 Y）的所有文本，并进入追加模式以在删除文本的位置添加文本

cT# 修改行上从光标到后向第 1 个目标字符之间的文本。例如 cTY 将删除当前行上从光标到向后第 1 个字符 Y 之间的所有文本，并进入追加模式以在删除文本的位置添加文本

cF# 修改行上从光标到后向第 1 个目标字符之间的文本（包括目标字符）。例如 cFY 将删除当前行上从光标到向后第 1 个字符 Y 之间（包

括 Y）的所有文本，并进入追加模式以在删除文本的位置添加文本

接出和粘贴行的单词

yy 将当前行复制或接出到内在缓冲区。20yy 将当前行和它后面的 19 行（共 20 行）文本复制到内存。目标行仍然保留在文件中，可以使用 p 命令将这些内存中的文本粘贴到文件中

dd 删除当前行，并将它放到与 yy 命令使用的相同的内存缓冲区。目标行从文件中删除，但是可以使用 p 命令将它粘贴到文件中的其他地方

yw 将当前光标所在的单词接出或复制到内在缓冲区。6yw 命令将把当前单词和它后面的 5 个（共 6 个）单词复制到内存

dw 删除当前的单词，并将它放到与 yw 命令使用的相同的内存缓冲区。可以使用 p 命令将单词粘贴到文件的其他地方

yt# 接出从光标到向前一个字符（不包括该字符）之间的文本。例如，ytB 命令将从光标到字符 B 的下一实例（不包括字符 B）之间的文本接出或复制到内存

yf# 接出从光标到向前一个字符（包括该字符）之间的文本。例如，yf:命令将从光标到字符:的下一个实例（包括字符:）之间的文本接出或复制到内存

yT# 后向接出（不包括目标字符）。例如，yTN 命令将从光标到字符 N 的后向第 1 个实例之间的文本（不包括字符 N）接出或复制到内存

yF# 后向接出（包括目标字符）。例如，yFJ 命令将把从光标到字符 N 的向后第 1 个实例之间的文本（包括字符 N）接出或复制到内存

p 将内存中的文本行粘贴到文件中光标所在行的下面，或将内存中的单词粘贴到文件中光标的右侧

P(大写) 将接出或删除的文本行粘贴到文件中光标所在行的上面。或将接出或删除的单词粘贴到文件中光标的左侧

文件移动命令 ：

J 将下行文本同当前行合并成一行

:#,# move # 将指定的行移到目标位置。:12,35 move 58 命令将第 12～35 行之间的所有文本移到第 58 行的后面。缩写为 mo

:1,26 co 82 将第 1～26 行之间的所有文本复制到第 82 行的后面（可以选择行号）

使用可视化编辑器进行全局编辑 ：

:s /target/replacement/ 查找当前行上目标字符串的第 1 个实例并删除，然后用字符串 replacement 替换。只修改当前行上的第 1 个目标实例

:g /target/s//replacement/ 查找所有行上目标字符串的第 1 个实例并删除，然后用字符串 replacement 替换。修改所有行上目标的第 1 个实例

:#,# s/target/replacement/ 在指定的行上进行替换。例如，:7,37

s/march/walk/将查找第 7～37 行之间的所有文本行，并用字符串 walk 替换每行中的第 1 个目标字符串 march。所有指定行上的第 1 个目标字符串修改

:#,# s/target/replacement/g 在指定的行上进行全局替换。例如，:1,$ s/fun/joyful/g 将在第 1 行到文件最后一行之间查找目标字符串 fun 的所有实例并删除，然后用字符串 joyful 替换。指定行上的目标字符串的所有实例都被修改

:g /target/s/replacement 查找所有行上目标字符串的第 1 个实例并删除，然后用字符串 replacement 替换。所有行上的第 1 个目标都被修改

:#,# target/s/replacement/ 在指定的行上进行替换，例如，:7,37 march/s/walk 命令在第 7～37 行之间每个文本行上查找目标字符串的第 1 个实例并删除，然后用字符串 walk 替换。所有指定行上的第 1 个目标字符串都被修改

:#,# target/s/replacement/g 在指定的行上进行全局替换。例如，:1,$ fun/s/joyful/g 将在第 1 行到文件最后一行之间查找目标字符串 fun 的所有实例并删除，然后用字符串 joyful 替换。指定行上的所有目标字符串都被修改

编辑工具：映射，缩写和标记

m# 用字母标记当前行。例如，ma 命令表示用 a 标记当前行。即使

移动了标记行，它仍然标记为 a，可以用 a 来定位该行

'# 定位标记行。例如，命令'a 将光标移到标记 a 的行。命令'a,$d 将删除从标记行到文件末尾之间的所有行

:map # command string 在命令模式中输入＃时，将其作为一个命令串。例如，:map #o#!/bin/ksh 产生一个新的命令模式指令，您输入＃时，它被解释为:o 打开个新行，并添加文本#!/bin/ksh 到文件中。为了在命令中包含回车和其他控制字符，可以在它们的前面用 Ctrl+V 命令

:ab abbreviation char-string 设置追加模式缩写。例如，在命令模式中输入:ab mv Milky Way Galaxy，则建立缩写。如果在追加模式中输入字符串 mw，然后按 Esc 键，mw 将被 Milky Way Galaxy 替代

在 vi 编辑器中向 shell 发出命令 ：

:!ls 启动一个 shell，并让 shell 运行 ls 程序。在运行完指定的程序后，必须按回车键以返回到编辑器中

:!ksh 启动一个 shell，它允许运行多个命令。退出 shell 可以回到编译器中

:Or!speel% 对当前文件(%)进行拼写检查，并将 spell 的输出诗篇到当前文件中，从第 1 行(0 行后面)开始放置这些输出

:31r!command% 运行 UNIX 命令（如 cal 或 date）并将它的输出读入到当前文件，从第 31 行开始放置这些输出

Ctrl+Z 用于挂起当前编辑会话进程的 csh 和 ksh 命令，它允许您向父 shell 发出命令

fg 重新激活挂起的编辑进程的 csh 和 ksh 命令

读、写和退出编辑器 ：

:wq 保存编辑会话期间对文件所做的修改，退出编辑器返回到 shell

:q 如果没有对文件进行修改或添加文件，可以用 q 退出对一个文件的编辑

:q! 退出对文件的编辑返回到 shell 模式，但是不保存在编辑会话期间对文件所做的修改

:w filename v 将文件的缓冲区副本（修改版本）保存到一个新文件

[color=#DC143C][b]:#,# w newtest 例如，:1,6 w newtext 命令创建一个名为 newtext 的文件，并将当前文件的第 1～6 行文本复制到文件 newtext 中

:1,6 w >> oldfile 将当前文件的第 1～6 行文本的一个副本追加到已有文件 oldfile 的末尾

:1,6 w! oldfile 用当前文件的第 1～6 行文本覆盖文件 oldfile