**光标移动：**

g\_ 光标移动当本行最后一个非空白字符

% 括号匹配移动

\* 匹配当前单词，移动到下一个单词

# 匹配当前单词，移动到上一个单词

f<字符> 移动到下一个字符。如：fa，移动到下一个字符a

3fa 移动到第3个a

F<字符> 移动到上一个字符。如：Fa，移动到上一个字符a

t<字符> 移动到下一个字符前的位置。如：ta，移动到下一个a之前的位置

T<字符> 移动到下一个字符后的位置。如：Ta，移动到上一个a之后的位置

dfa 删除到下一个a

dta 删除到下一个a之前

ctrl+e 屏幕向前滚一行，光标仍在当前行

ctrl+y 屏幕向后滚一行，光标仍在当前行

z[Enter] 光标所在行成为屏幕第一行

z. 光标所在行成为屏幕中间行

z- 光标所在行成为屏幕最后行

z命令前可加n，表示对第n行的操作，如：21z[Enter]，使21行成为屏幕第一行

enter/+ 光标移动到下一行的第一个字符

- 光标移动到上一行的第一个字符

H 光标移动到当前屏第一行

M 光标移动到当前屏中间行

L 光标移动到当前屏末行

nH 光标移动到当前屏第n行

nL 光标移动到当前屏倒数第n行

nG/ngg 光标移动到第n行

小写字母o 光标在选中的块区域两端交替移动

ctrl+g 显示光标所在行信息

Vu 光标所在行全小写

VU 光标所在行全大写

**编辑命令：**

u撤销修改

ctrl+r 撤销后重做

ctrl+n/p 单词联想

. 重复上次编辑操作

N<command> 重复command N次

~ 光标所在字符大小写转换

gu/gU 块文本转小写/大写

**删除命令：**

x 删除光标所在处字符

X 删除光标所在前字符

daw 删除光标所在的整个单词

dw 删除到下一个单词开头，含光标所在字符

de 删除到本单词末尾，含光标所在字符

dE 删除到本单词末尾包括标点在内，含光标所在字符

db 删除到前一个单词，不含光标所在字符

dB 删除到前一个单词包括标点在内，不含光标所在字符

dd 删除一整行

D或d$ 删除光标位置到本行结尾，含光标所在字符

d0 删除光标位置到本行开头，不含光标所在字符

**块操作：**

ctrl+v 开始，移动光标选择块，Iab[ESC]在块中所有行的光标前插入ab；

ctrl+v 开始，移动光标选择块，$到行尾，Aab[ESC]在块中所有行的尾部插入ab；

块选择后，输入大写的J，所有行连起来变成一行；

**分屏操作：**

split/vsplit 水平/垂直分屏

先按<ctrl+w>，然后按<方向键> 在分屏间移动

**寄存器：**

:reg 查看所有寄存器

:reg <register name> 查看指定寄存器

“<register name> normal mode访问指定寄存器

ctrl+r(vim打出”)之后输入<register name> command mode访问寄存器

insert mode访问寄存器与command mode相同

举例，复制一段文本后在command mode查找：

用v选中文本，copy到寄存器a: “ay，输入/进入command mode，输入ctrl+r(vim显示出”)，输入寄存器名a。

**文档：**

:help

:help usr\_02.txt

**文件操作：**

saveas <path/file> 另存为

:E 浏览当前打开的文件所在目录下的文件列表(在Ubuntu上这个命令不管用)

:e <path/file> 打开文件

**重复命令：**

. 文本编辑重复

; 行内查找重复(向前)

, 行内查找重复(向后)

& 文本替换重复(可以采用这样的形式:s/target/replacement/g来将行内出现的字符串target替换为另外一个字符串replacement,要是我们想在其他行执行相同的替换工作，跳到其他行，按&即可重复替换操作)

@[寄存器] 宏重复

**中文乱码解决：**

set encoding=utf-8

**vim字符编码设置：**

* encoding: Vim 内部使用的字符编码方式，包括 Vim 的 buffer (缓冲区)、菜单文本、消息文本等。默认是根据你的locale选择.用户手册上建议只在 .vimrc 中改变它的值，事实上似乎也只有在.vimrc 中改变它的值才有意义。你可以用另外一种编码来编辑和保存文件，如你的vim的encoding为utf-8,所编辑的文件采用cp936编码,vim会自动将读入的文件转成utf-8(vim的能读懂的方式），而当你写入文件时,又会自动转回成cp936（文件的保存编码).
* fileencoding: Vim 中当前编辑的文件的字符编码方式，Vim 保存文件时也会将文件保存为这种字符编码方式 (不管是否新文件都如此)。
* fileencodings: Vim自动探测fileencoding的顺序列表， 启动时会按照它所列出的字符编码方式逐一探测即将打开的文件的字符编码方式，并且将 fileencoding 设置为最终探测到的字符编码方式。因此最好将Unicode 编码方式放到这个列表的最前面，将拉丁语系编码方式 latin1 放到最后面。
* termencoding: Vim 所工作的终端 (或者 Windows 的 Console 窗口) 的字符编码方式。如果vim所在的term与vim编码相同，则无需设置。如其不然，你可以用vim的termencoding选项将自动转换成term的编码.这个选项在 Windows 下对我们常用的 GUI 模式的 gVim 无效，而对 Console 模式的Vim 而言就是 Windows 控制台的代码页，并且通常我们不需要改变它。

**显示tab：**

:set list tab键显示为^i，而$显示在每行的结尾。