2. 编辑模式
跳转: k j h l;ctrl+f（page down）;ctrl+b（page up）
更大范围的移动：
* 光标停在一个单词,*文件内搜索该单词,跳转到下一处
# 光标停在一个单词,# 文件内搜索该单词,跳转到上一处
(/) 移动到 前/后 句 的开始
{/} 跳转到 当前/下一个 段落 的开始。
g_ 到本行最后一个不是 blank 字符的位置。
fa 到下一个为 a 的字符处。
t, 到逗号前的第一个字符。逗号可以变成其它字符。
3fa 在当前行查找第三个出现的 a。
F/T 和 f 和 t 一样，只不过是相反方向;
gg 将光标定位到文件第一行起始位置；
G 将光标定位到文件最后一行起始位置；
NG 或 Ngg 将光标定位到第 N 行的起始位置。
H 移到屏幕上的起始行（或最上行）；
M 移到屏幕中间； L 移到屏幕最后一行。
注意字母的大小写。还可以加数字。如 2H 表示将光标移到屏幕的第 2 行，3L 表示将光标移到屏幕的倒数第 3行。
w 右移光标到下一个字的开头；
e 右移光标到一个字的末尾；
b 左移光标到前一个字的开头；
0 数字 0，左移光标到本行的开始；
$ 右移光标，到本行的末尾；
^ 移动光标，到本行的第一个非空字符。
2.2 搜索匹配
/str1 正向搜索字符串 str1；
n 继续搜索，找出 str1 字符串下次出现的位置；
N 继续搜索，找出 str1 字符串上一次出现的位置；
?str2 反向搜索字符串 str2 。
到达文件尾或头时，搜索会循环到文件的另一端继续执行。
2.3 替换和删除
rc 用c 替换光标所指向的当前字符；
nrc 用c 替换光标所指向的前 n 个字符；
5rA 用A 替换光标所指向的前 5 个字符；
x 删除光标所指向的当前字符；
nx 删除光标所指向的前 n 个字符；
3x 删除光标所指向的前 3 个字符；
dw 删除光标右侧的字；
ndw 删除光标右侧的 n 个字；
3dw 删除光标右侧的 3 个字；
db 删除光标左侧的字；ndb 删除光标左侧的 n 个字；
5db 删除光标左侧的 5 个字；
dd 删除光标所在行，并去除空隙；
ndd 删除（剪切） n 行内容，并去除空隙；
3dd 删除（剪切） 3 行内容，并去除空隙；
d$ 从当前光标起删除字符直到行的结束；
d0 从当前光标起删除字符直到行的开始；
J 删除本行的回车符（CR），并和下一行合并。
替换命令执行以后，通常会由 编辑模式 进入 插入模式 ：
s 用输入的正文替换光标所指向的字符；
S 删除当前行，并进入插入模式；
ns 用输入的正文替换光标右侧 n 个字符；
nS 删除当前行在内的 n 行，并进入插入模式；
cw 用输入的正文替换光标右侧的字；
cW 输入的正文替换从光标到行尾的所有字符（同 c$ ）；
ncw 用输入的正文替换光标右侧的 n 个字；
cb 用输入的正文替换光标左侧的字；
ncb 用输入的正文替换光标左侧的 n 个字；
cd 用输入的正文替换光标的所在行；
ncd 用输入的正文替换光标下面的 n 行；
c$ 用输入的正文替换从光标开始到本行末尾的所有字符；
c0 用输入的正文替换从本行开头到光标的所有字符。
2.4 复制粘贴
p 小写字母p，将缓冲区的内容粘贴到光标的后面；
P 大写字母P，将缓冲区的内容粘贴到光标的前面。
如果缓冲区的内容是字符或字，直接粘贴在光标的前面或后面；如果缓冲区的内容为整行正文，执行上述粘贴命令将会粘贴在当前光标所在行的上一行或下一行。
yy 复制当前行到内存缓冲区；
nyy 复制 n 行内容到内存缓冲区；
5yy 复制 5 行内容到内存缓冲区；
“+y 复制 1 行到操作系统的粘贴板；
“+nyy 复制 n 行到操作系统的粘贴板。
2.5 撤销和重复
u 撤消前一条命令的结果；. 重复最后一条修改正文命令。
3.1 进入插入模式
i 在光标左侧插入 a 在光标右侧插入
o 在光标所在行的下一行增添新行
O 在光标所在行的上一行增添新行
I 在光标所在行的开头插入
A 在光标所在行的末尾插入
3.2 退出插入模式:按 ESC 键或组合键 Ctrl+[，退出插入模式之后，将会进入编辑模式 。
4.1 打开、保存、退出
:e path_to_file/filename :w :w file_temp
ZZ 退出 Vim,该命令保存对正文所作修改,覆盖原始文件。

: q 在未作修改退出； : q! 放弃所有修改退出 :wq

4.2 行号与文件（与 编辑模式 下的 ngg 或 nG 相同）：

: n 将光标移到第 n 行命令模式下，可以规定命令操作的行号范围。数值用来指定绝对行号；字符“.”表示光标所在行的行号；字符符“$”表示正文最后一行的行号；简单的表达式，例如“.+5”表示当前行往下的第 5 行。

:345 将光标移到第 345 行

:345w file 将第 345 行写入file 文件

:3,5w file 将第 3 行至第 5 行写入file 文件

:1,.w file 将第 1 行至当前行写入file 文件

:.,$w file 将当前行至最后一行写入file 文件

:.,.+5w file 从当前行开始将 6 行内容写入file 文件

:1,$w file 所有内容写入file文件,相当于:w file 命令

:w 编辑的内容写入原始文件，用来保存编辑的中间结果

:wq 编辑的内容写入原始文件并退出（相当于 ZZ 命令）

:w file 编辑内容写入file 文件，保持原有文件内容不变

:a,bw file 将第 a 行至第 b 行的内容写入file 文件

:r file 读取 file 文件内容，插入当前光标所在行后面

:e file 编辑新文件 file 代替原有内容

:f file 将当前文件重命名为 file

:f 打印当前文件名称和状态

4.3 字符串搜索

:/str/ 正向搜索，将光标移到下一个包含字符串 str 的行

:?str? 反向搜索，光标移到上一个包含字符串 str 的行

:/str/w file 正向搜索，将第一个包含字符串 str 的行写入file 文件

:/str1/,/str2/w file 正向搜索，将包含 str1 的行至包含 str2 的行写

4.4 Vim 中的正则表达式

:/^struct/^ 字符比较每行开头的字符串。也可以用类似办法在搜索字符串后面加上表示行的末尾的特殊字符$ 来找出位于行末尾的字：

:/^struct/

下表给出大多数特殊字符和它们的含义：

^ 放在字符串前面，匹配行首的字；

$ 放在字符串后面，匹配行尾的字；

\<匹配一个字的字头；

\>匹配一个字的字尾；

. 匹配任何单个正文字符；

[str] 匹配 str 中的任何单个字符；

[^str] 匹配任何不在 str 中的单个字符；

[a-b] 匹配 a 到 b 之间的任一字符；

* 匹配前一个字符的 0 次或多次出现；

\ 转义后面的字符。

4.5 正文替换

利用:s 命令可以实现字符串的替换。具体的用法包括：

:%s/str1/str2/ str2 替换行中首次出现的字符串 str1

:s/str1/str2/g str2 替换行中所有出现的字符串 str1

:.,$ s/str1/str2/g 用str2 替换当前行到末尾所有的 str1

:1,$ s/str1/str2/g 用str2 替换正文中所有出现的 str1

:g/str1/s//str2/g 功能同上

:m,ns/str1/str2/g 将从 m行到 n行的 str1 替换成 str2

从上述替换命令可以看到：

1. `g` 放在命令末尾，表示对搜索字符串的每次出现进行替换,不止匹配每行中的第一次出现；不加`g`，表示只对搜索字符串的首次出现进行替换；`g` 放

2. `s` 表示后面跟着一串替换的命令；

3. `%` 表示替换范围是所有行，即全文。

另外一个实用的命令，在 Vim 中统计当前文件中字符串 str1 出现的次数，可用替换命令的变形：

:%s/str1/&/gn

4.6 删除正文

在命令模式下，同样可以删除正文中的内容。例如：

:d 删除光标所在行

:3d 删除 3 行

:.,$d 删除当前行至正文的末尾

:/str1/,/str2/d 删除从字符串 str1 到 str2 的所有行

:g/^\(.*\)$\n\1$/d 删除连续相同的行，保留最后一行

:g/\%(^\1$\n\)\@<=\(.*\)$/d 删除连续相同的行，保留最开始一行

:g/^\s*$\n\s*$/d 删除连续多个空行，只保留一行空行

:5,20s/^#//g 删除 5 到 20行开头的# 注释

总之，Vim 的初级删除命令是用d ，高级删除命令可以用正则替换 的方式执行。

4.7 恢复文件

Vim 在编辑某个文件时，会另外生成一个临时文件，这个文件的名称通常以. 开头，并以 .swp 结尾。Vim 在正常退出时，该文件被删除，若意外退出，而没有保存文件的最新修改内容，则可以使用恢复命令:recover 来恢复文件，也可以在启动 Vim 时用-r 选项。

4.8 选项设置

利用:set 命令可以设置选项。基本语法为：

:set option 设置选项 option
常见的功能选项包括：
autoindent 设置该选项，则正文自动缩进
ignorecase 设置该选项，忽略规则表达式中大小写区别
number 设置该选项，则显示正文行号
ruler 在屏幕底部显示光标所在行、列的位置
tabstop 设置按 Tab 键跳过的空格数。例如 :set tabstop=n，n 默认值为 8
mk 将选项保存在当前目录的.exrc 文件中
4.9 Shell 切换
:!shell_command 执行完 shell_command 后回到 Vim
4.10 分屏与标签页
:split（可用缩写:sp） 上下分屏；
:vsplit（可用缩写:vsp） 左右分屏。
另外，也可以在终端里启动 vim 时就开启分屏操作：
vim -On file1 file2...垂直分屏
vim -on file1 file2... 水平分屏
Ctrl+w+h 切换到左边一屏；Ctrl+w+l 切换到右边一屏；
Ctrl+w+j 切换到下方一屏；Ctrl+w+k 切换到上方一屏。
即键盘上的 h,j,k,l 四个 Vim 专用方向键，配合 Ctrl 键和 w 键（window 的缩写），就能跳转到目标分屏。另外，也可以直接按 Ctrl+w+w 来跳转分屏，不过跳转方向则是在当前 Vim 窗口所有分屏中，按照逆时针方向跳转。
下面是改变尺寸的一些操作，主要是高度，对于宽度你可以使用[Ctrl+W <] 或是 [Ctrl+W >] ，但这可能需要最新的版本才支持。
Ctrl+W = 让所有的屏都有一样的高度；
Ctrl+W + 增加高度； Ctrl+W - 减少高度。
标签页
Vim 的标签（Tab）页，类似浏览器的标签页，一标签页打开一个 Vim 窗口，一个 Vim 的窗口可以支持 N 个分屏。
新建一个标签：:tabnew 新建标签页同时打开一个文件：:tabnew filename
Vim 中的每个标签页有一个唯一的数字序号，第一个标签页的序号是 0，从左向右依次加一。关于标签页有一系列操作命令，简介如下：
:tN[ext] 跳转到上一个匹配的标签
:tabN[ext] 跳到上一个标签页
:tabc[lose] 关闭当前标签页
:tabdo 为每个标签页执行命令
:tabe[dit] 在新标签页里编辑文件
:tabf[ind] 寻找 'path'里的文件，在新标签页里编辑之
:tabfir[st] 转到第一个标签页
:tabl[ast] 转到最后一个标签页
:tabm[ove] N 把标签页移到序号为 N 位置
:tabnew [filename] 在新标签页里编辑文件
:tabn[ext] 转到下一个标签页
:tabo[nly] 关闭所有除了当前标签页以外的所有标签页
:tabp[revious] 转到前一个标签页
:tabr[ewind] 转到第一个标签页
4.11 与外部工具集成
vimdiff file1 file2
即可在 Vim里分屏显示两个文件内容的对比结果，对文件内容差异部分进行高亮标记，还可以同步滚动两个文件内容，更可以实时修改文件内容。
vimdiff a.txt b.txt
如果直接给 -d 选项是一样的
vim -d a.txt b.txt
除了在终端里开启 vimdiff 功能，也可以在打开 Vim 后，在 Vim 的命令模式输入相关命令来开启 vimdiff 功能：
:diffsplit abc.txt
如果你现在已经开启了一个文件，想 Vim 帮你区分你的文件跟 abc.txt 有什么区别，可以在 Vim 中用diffsplit 的方式打开第二个文件，这个时 候 Vim 会用split（分上下两屏）的方式开启第二个文件，并且通过颜色，fold 来显示两个文件的区别
这样 Vim 就会用颜色帮你区分开 2 个文件的区别。如果文件比较大（源码）重复的部分会帮你折叠起来。
:diffpatch filename
通过 :diffpatch 你的 patch 的文件名，就可以以当前文件加上你的 patch 来显示。vim 会 split一个新的屏，显示patch 后的信息并且用颜色标明区别。
如果不喜欢上下对比，喜欢左右（比较符合视觉）可以在前面加 vert ，例如：
:vert diffsplit abc.txt
:vert diffpatch abc.txt
看完 diff，用:only 回到原本编辑的文件，觉得 diff 的讨厌颜色还是在哪里，只要用:diffoff 关闭就好了。
还有个常用的 diff 中的就是 :diffu ,这个是 :diffupdate 的简写，更新的时候用。
Vim 的 diff 功能显示效果如下所示：
sort
Linux 命令 sort 可以对文本内容进行按行中的字符比较、排序，但在终端里使用sort 命令处理文件，并不能实时查看文件内容。具体用法请自查手册。
xxd

vim+xxd 是 Linux 下最常用的二进制文本编辑工具，xxd 其实是 Vim 外部的一个转换程序，随 Vim一起发布，在 Vim里调用它来编辑二进制文本非常方便。首先以二进制模式在终端里打开一个文件：

vim -b filename

Vim 的 -b 选项是告诉 Vim 打开的是一个二进制文件，不指定的话，会在后面加上 0x0a ，即一个换行符。

然后在 Vim 的命令模式下键入：

:%!xxd　　即可看到二进制模式显示出来的文本,

然后就可以在二进制模式下编辑该文件，编辑后保存，然后用下面命令从二进制模式转换到普通模式：

:%!xxd -r

另外，也可以调整二进制的显示模式，默认是 2 个字节为一组，可以通过 g 参数调整每组字节数：

:%!xxd -g 1 表示每 1 个字节为 1 组

:%!xxd -g 2 表示每 2 个字节为 1 组(默认)

:%!xxd -g 4 表示每 4 个字节为 1 组