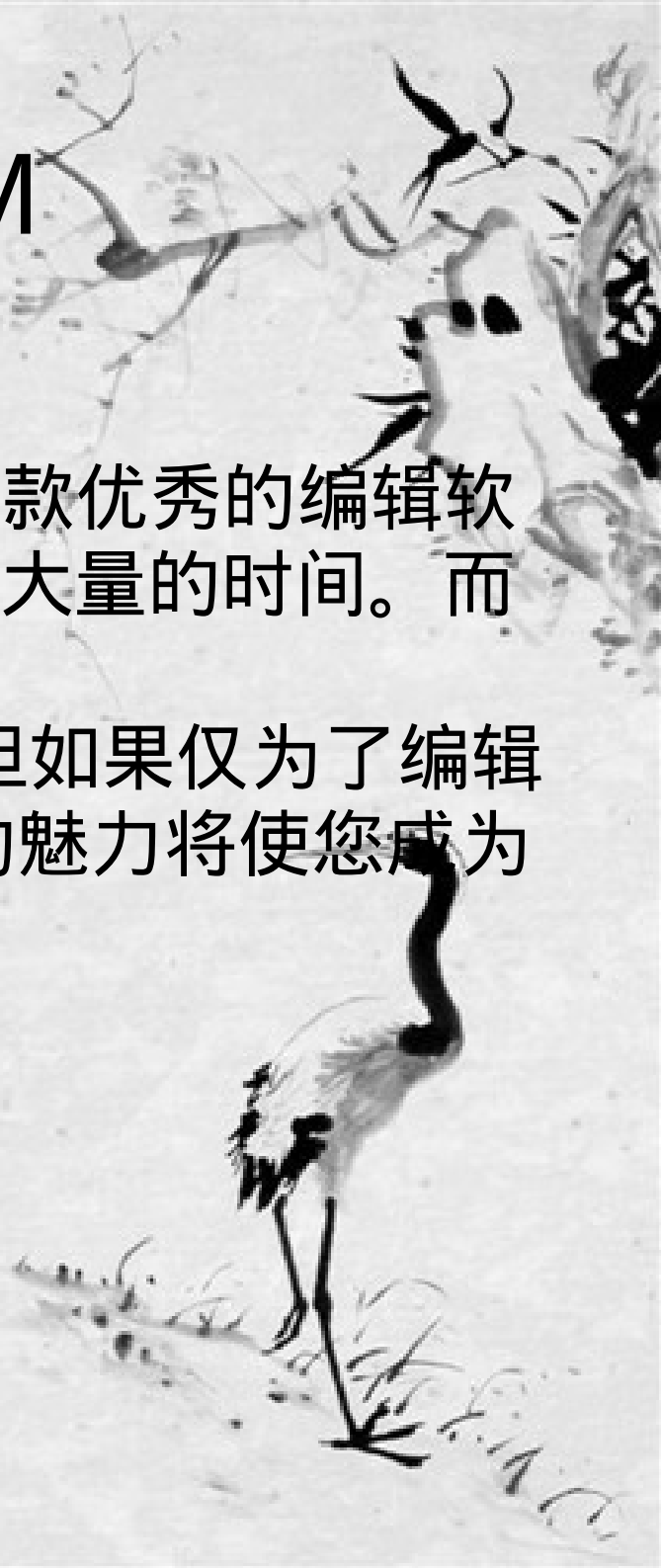
GNU/Linux
编辑器之神 --VIM

GNU/Linux--VIM

VIM

当今世界，文本编辑器种类繁多选择一款优秀的编辑软件至关不仅仅提升工作效率，更能够节省大量的时间。而 VIM 与 emasc 成为了首选之一。

emasc 的功能强大，确实无以伦比。但如果仅为了编辑文本，那么 VIM 以其强大的功能和无穷的魅力将使您成为不悔的选择。



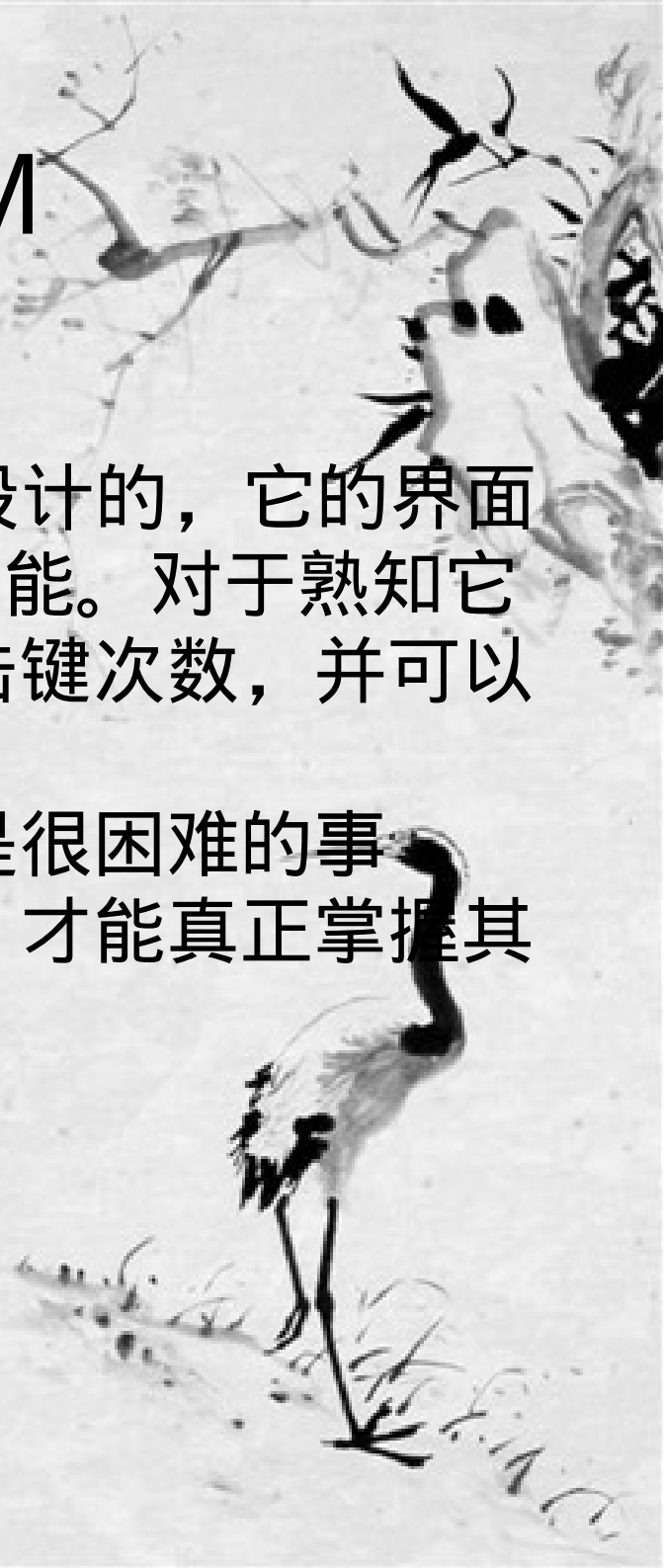
GNU/Linux--VIM

VIM

VIM 编辑器是专门为经验丰富的用户设计的，它的界面和使用方法提供了更快的速度和更强的功能。对于熟知它的用户， VIM 的许多特性节省了时间和击键次数，并可以完成一些其他编辑器无法完成的功能。

对于初学用户要掌握好 VIM 编辑器也不是很困难的事。

学习 VIM 的最好方法是实践，唯有如此，才能真正掌握其中的精髓。



GNU/Linux--VIM

命令：

#vi

或

#vim

功能：一个强大的文本编辑器

语法格式 :vim [选项] / 路径 / 文本文件名

GNU/Linux--VIM

VIM 基础使用
要使用 vim 编辑器只要在控制台中执行 vi 就可以。

命令格式：　vi [选项]　[文件名]
+num　　打开某个文件直接跳转到 num 行
-b　　以 binary 方式打开文件 , 用于编辑二进制文件
-R　　以只读方式打开文件

vi 还有很多选项而且每个版本的 vi 选项也会有所不同详细的选项，可以查看 man 手册获得，如果执行 vi 的时候没有加上文件名则开始编辑一个新文件

GNU/Linux--VIM

VIM 基础使用

vi 编辑器共有三种工作模式

1. 命令行模式　　command line
2. 输入模式　　input mode
3. 末行模式　　last line mode

command line　主要做替换，删除，复制等工作。

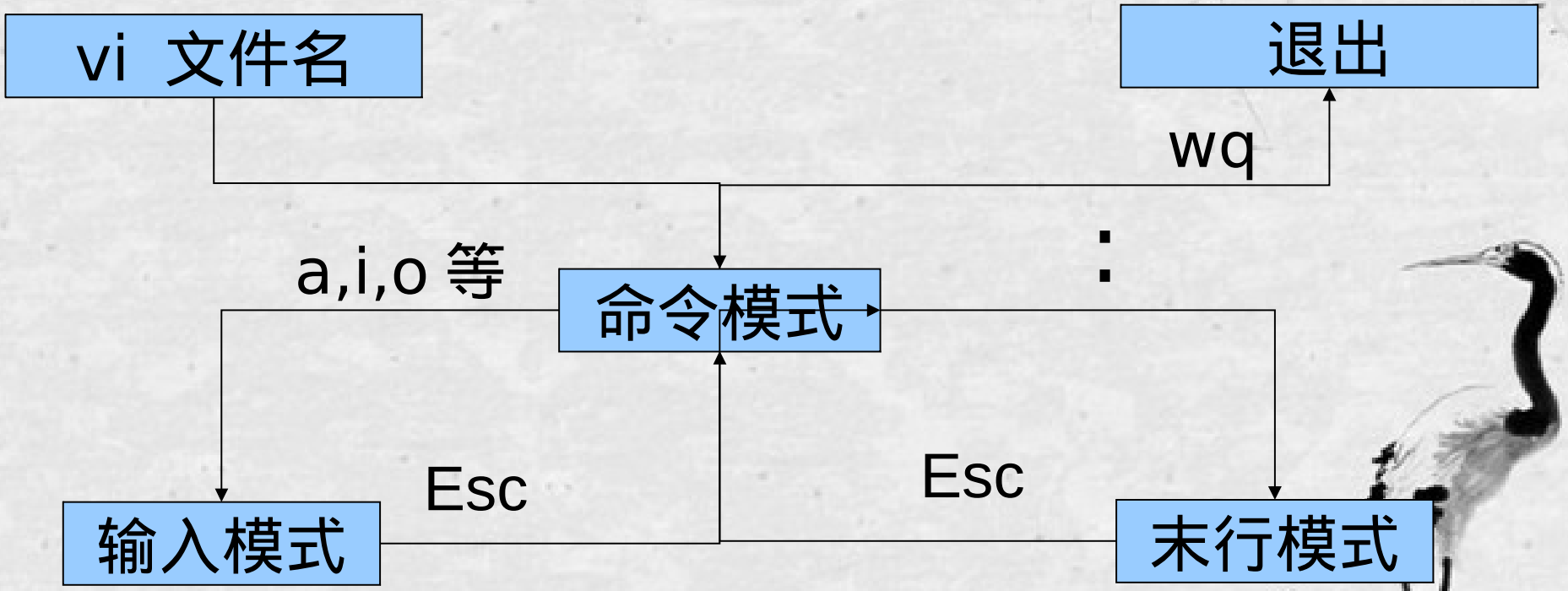
input mode　主要做内容编辑

last line mode　主要对文件进行编辑操作，如保存文件

GNU/Linux--VIM

VIM 基础使用

模式切换



GNU/Linux--VIM

VIM 基础使用

command mode

在 command mode 执行操作的时候都是先要将光标移动到要执行命令的地方然后再执行命令的，怎么移动光标呢？

j	光标向下移动
k	光标向上移动
h	光标向左移动
l	光标向右移动

如果你觉得记这些不太方便也可以用小键盘上的←↑→↓来代替

GNU/Linux--VIM

VIM 基础使用

如果要移动到文件的最后一行，一下下按方向键实在太麻烦

G(shift+g)	直接移动到文件末尾
gg	直接移动到文件头

如果要移动到指定的某一行的话 vi 也可以办到

假设光标当前在 500 行

1000G	向下移动到 1000 行
50gg	向上移动到 50 行

G 是向下移动 ,gg 是向上移动，别记错哦。

GNU/Linux--VIM

VIM 基础使用
如果要移动到行头或者行末 vi 也可以轻松完成
0(数字键)　　直接移动到行头
$　　直接移动到行末

单词移动
w　　直接移动到下一个单词
b　　直接移动到上一个单词

还有很多移动键 B,e 可以自己试着在 vi 里使用
当然前面讲到的 g 和 G 都可以和 w,b,B,e 配合使用
比如 gb,ge 之类的

GNU/Linux--VIM

VIM 基础使用

句子移动
(　　移动到前一个语句
)　　移动到后一个语句

段落移动
{　　移动到上一个段落
}　　移动到下一个段落

GNU/Linux--VIM

VIM 基础使用

屏幕的移动，屏幕移动不是指的移动显示器而是 vi 中的屏幕移动 .

PageUp	向下翻一页
PageDown	向上翻一页
H	将光标移动到当前屏幕的最上
M	将光标移动到当前屏幕的中间
L	将光标移动到当前屏幕的最下
zz	将光标当前行为基准放在屏幕中间

GNU/Linux--VIM

VIM 基础使用

光标移动到了想要操作的位置以后，接着就是操作了．

x	删除光标所在位置的字符
X	删除光标所在位置之前的一个字符
D	从光标开始到行末全部删除
dw	删除光标后的一个单词
dd	删除光标所在的一行
db	删除光标所在的前一个单词

GNU/Linux--VIM

VIM 基础使用

前面学了移动 G 和 gg, 又学了 dd 删除行。 现在把移动和删除结合起来使用

dG	删除光标所在行到文件末尾的所有内容
dgg	删除光标所在行到文件头的所有内容
dk	删除光标所在行和上面一行
dj	删除光标所在行和下面一行

GNU/Linux--VIM

VIM 基础使用

命令模式的功能还不止这些，在使用命令键的时候还可以加上数字 .

4x　　删除光标之后的 4 个字符 (包含光标)

4X　　删除光标前的 4 个字符

如果想要删除 4 个单词：可以用 d4w 或者 4dw

d4w 和 4dw 有什么区别 :

d4w　　一次删除 4 个单词

4dw　　删除一个单词重复 4 次

GNU/Linux--VIM

VIM 基础使用

在 vi 中撤销

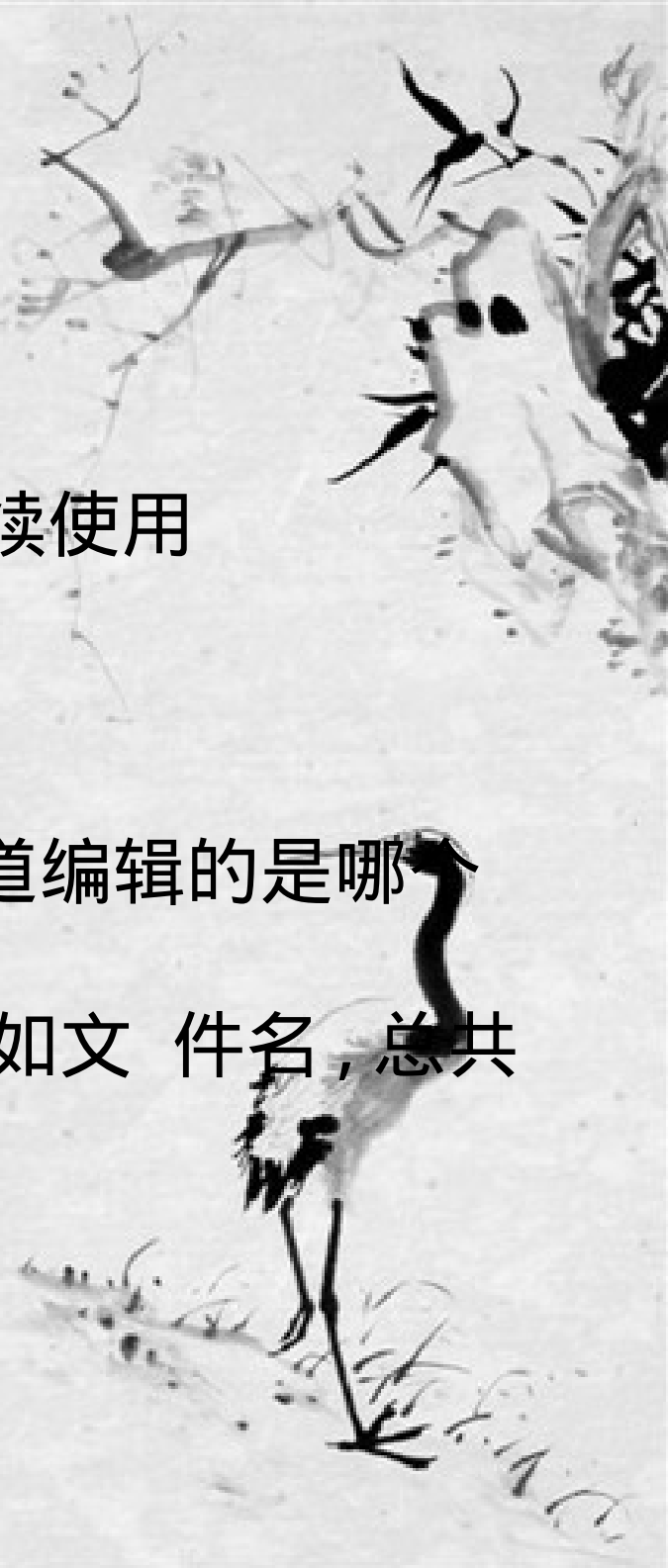
u　　撤销刚才的操作，可以连续使用

U　　撤销一行中的所有操作

ctrl + r　　取消撤销的内容

对文件做了这么多操作后，头晕了，不知道编辑的是哪个文件了。

ctrl + g　　显示当前编辑文件的信息 . 比如文 件名，总共的 行数，当前在总数中的百分比等信息



GNU/Linux--VIM

VIM 基础使用

% 移动到配对的符号如当前光标在 (键入 % 可以自动移动到配对的) 同样适用于 [] 和 {} 这些功能在编程的时候特别有用

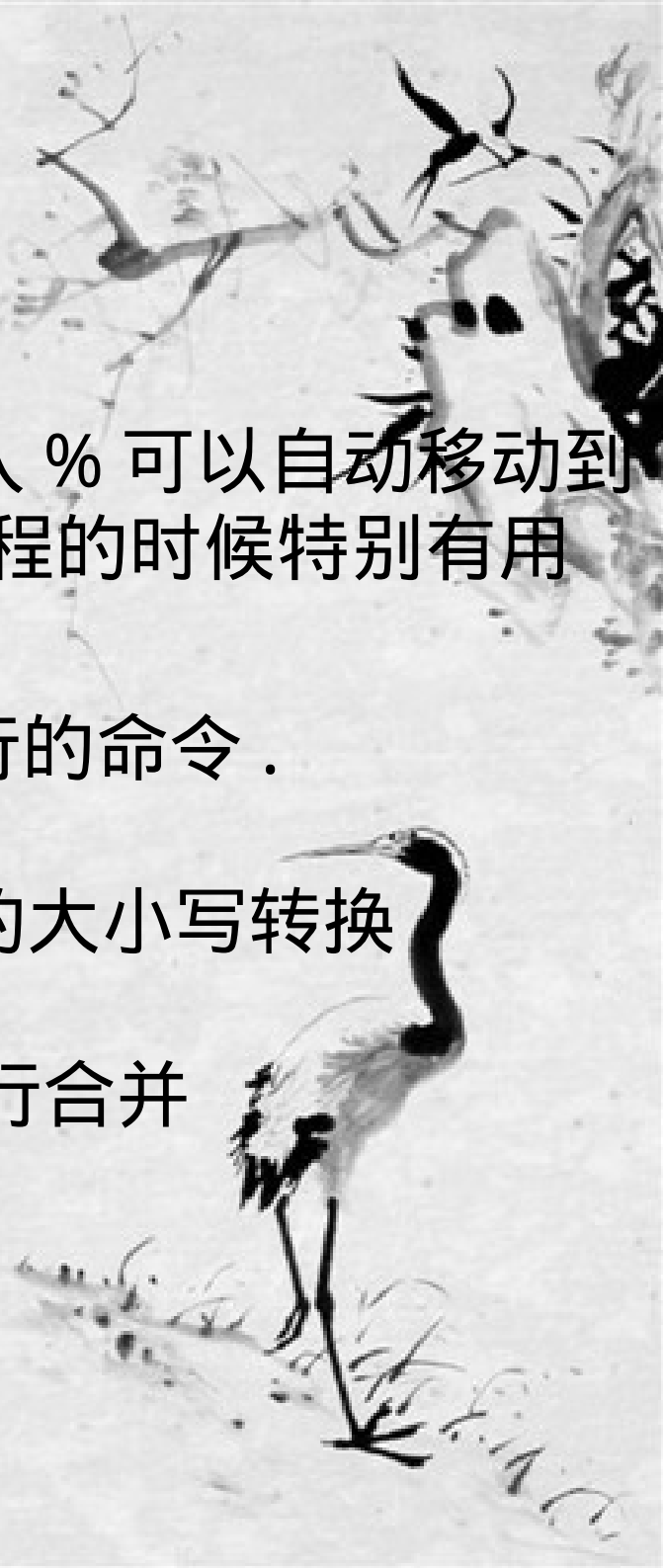
. 点 , 这个键的功能是重复最后一次执行的命令 .

g~ 这个功能非常有意思 ,g~ 能将当前行的大小写转换

J 这个键的功能是将光标当前行和下一行合并

注 : 这个键不能加上数字

r 替换光标所在的字符



GNU/Linux--VIM

VIM 基础使用
在使用编辑器的时候最常用的应该算复制，剪切和粘贴了

y	复制，如 yw 复制一个单词
yy	复制一行
p	粘贴（光标后）
P	粘贴（光标前）

在 vi 中的剪切实际上是删除 + 粘贴来完成的，比如要剪切一行，首先可以用 dd 删除一行然后再移动到想要粘贴的位置，按 p 就可以完成剪切了。

GNU/Linux--VIM

VIM 基础使用
在命令行中有一个 v 键：

这个键是用来代替鼠标用的。 v 的意思是‘可视’的意思，按过 v 键以后再使用方向键便可以象鼠标那样选中文字，然后可以对选中的文字进行操作。

例如：删除，复制等。值得注意的是此功能只有 vim 才有。

GNU/Linux--VIM

VIM 基础使用
在命令行中有一个大写 V 键：这个键是用来选择某个行

还可以用 ^v 来选择某个列

GNU/Linux--VIM

VIM 基础使用

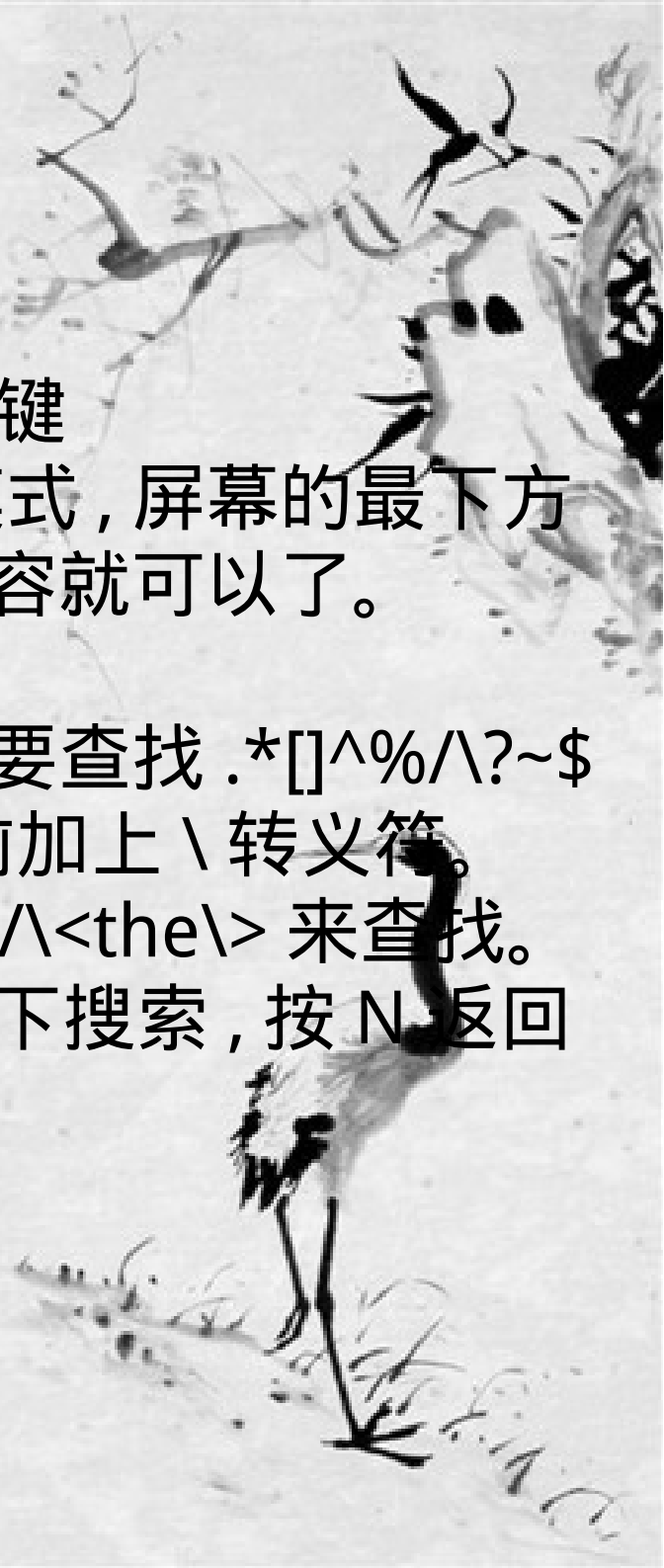
在命令模式中查找某个单词首先要按 / 键

这个时候就进入了命令模式中的搜索模式，屏幕的最下方会出现 / 提示符，然后输入你想要查找的内容就可以了。

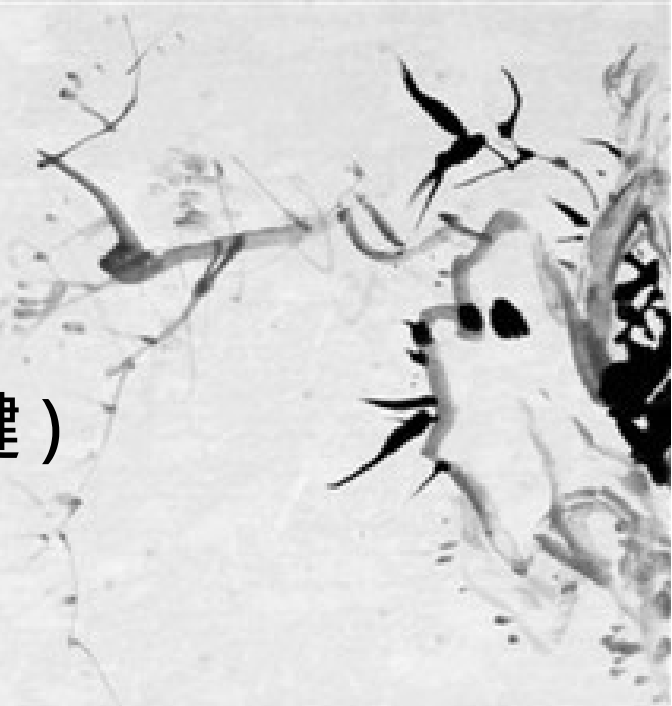
这里的搜索使用的是正则表达式，如果要查找 .*[]^%/\?~$ 这些具有特殊含义的字符需要在这些字符前加上 \ 转义符。

要精确匹配某个单词，比如 the 需要用 /\<the\> 来查找。

查找到的字符会高亮提示，按 n 继续向下搜索，按 N 返回上一个搜索。



GNU/Linux--VIM



输入模式

a　在光标后开始插入 (相当于 insert 键)

i　在光标前开始插入

A　在光标所在行末尾开始插入

I　在光标所在行首开始插入

o　在光标所在行下插入一行

O　在光标所在行上插入一行

R　进入替换模式

s　替换光标所在字符后开始编辑

S　删除光标所在行后开始编辑

GNU/Linux--VIM

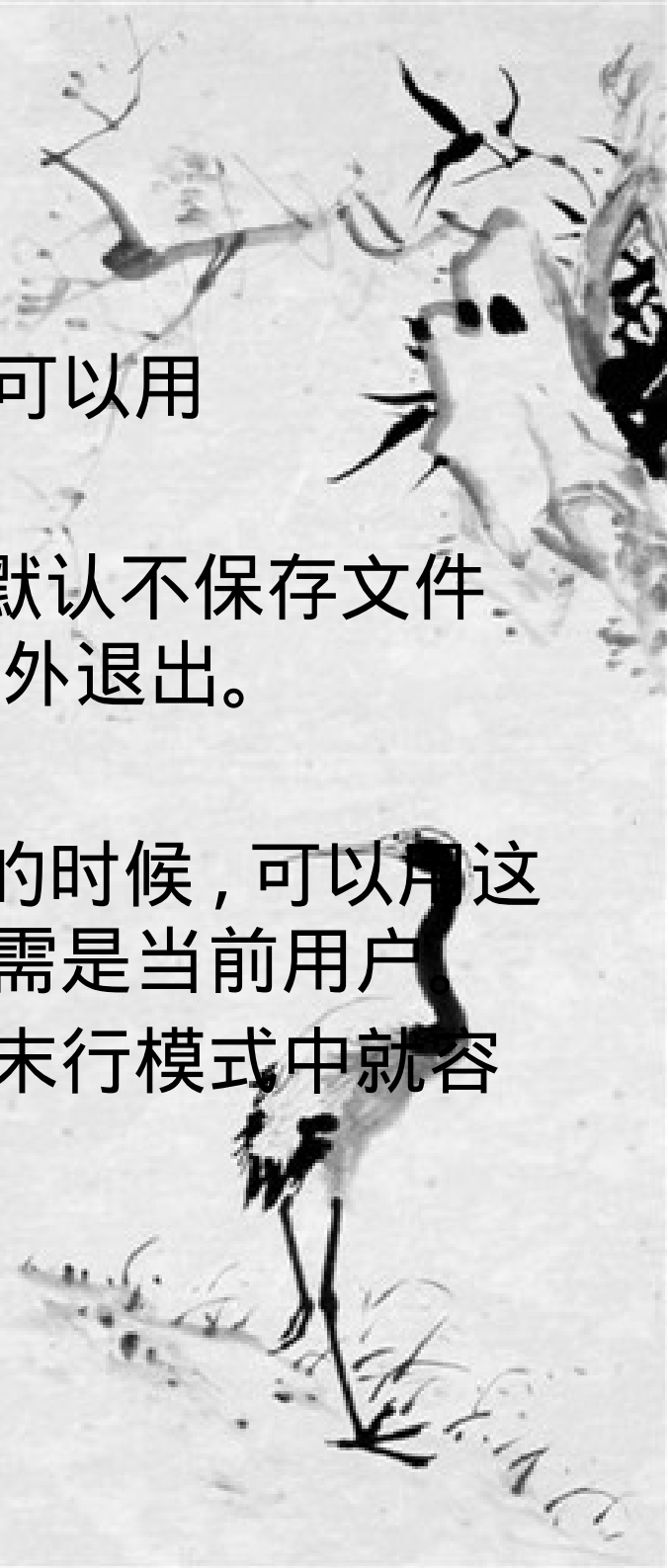
末行模式 last line mode

是 vi 编辑器里最重要的一个模式。因为在这里会提供能很多有用的功能，所以需要很好的掌握。

在命令模式下按下 : (shift+;) 键，就进入了末行模式。
进入末行模式的时候会在屏幕的最下面的地方显示 : 提示符，这就说明进入了末行模式。

:w 保存文件
:q 退出 vi 编辑
:wq 保存文件并退出 vi 编辑器 (无论是否修改了文件)
:e 不离开 VI， 开始编辑一个新的文件
:w <filename-new> 存储当前编辑的文件到一个新的文件
:x 文件仅被修改时才写入 并退出，未写入则直接退出

GNU/Linux--VIM



有时候对修改的不满意，需要退出不保存可以用

:q!　这个！具有强制的作用，因为在 vi 中默认不保存文件是无法退出 vi 编辑器的，主要为了防止意外退出。

:w!　强制保存，有的时候文件是只读属性的时候，可以用这个方式来保存，当然前提是文件所有者必需是当前用户。

在命令模式中要移动到一行比较麻烦，在末行模式中就容易多了，现在需要移动到第 543 行

:543　　就移动到 543 行了。

GNU/Linux--VIM

在末行中还有更实用的操作，假设需要删除 52 行 123 行的中的内容，在末行模式中轻而易举就能完成

:52,123d　　删除 52 到 123 行的内容

末行模式还能选择保存，现在需要将 123 行到 555 行的内容保存到 /home 目录下

:123,555w /home/lastlinemode.txt

如果要在当前编辑的文件中的某一行读入其它文件的内容

:23r /home/last.txt　　在当前文件的 23 行开始读入 /home　　目录下的 last.txt 文件

GNU/Linux--VIM

vi 还支持更高级的功能 ,vi 可以同时打开多个文件 , 打开方式 vi 加上要打开的文件名 , 文件和文件之间用空格隔开

vi 文件 1 文件 2 文件 3 文件 4

进入 vi 后 ,vi 打开的是文件 1

:next 切换到下一个文件

:previous 切换到上一个文件

:last 切换到最后一个文件

:first 切换到最前一个文件

:2next 切换到下二个文件

GNU/Linux--VIM

:args　　可以显示多个编辑文件中当前所编辑的文件

:args 还有另外一个功能就是对打开的多个文件进行重新排序，排列方法很简单，在末行模式执行

:args 5 3 1 2 4 6
这样原先 1 2 3 4 5 6 文件的排列顺序就变成 5 3 1 2 4 6 了

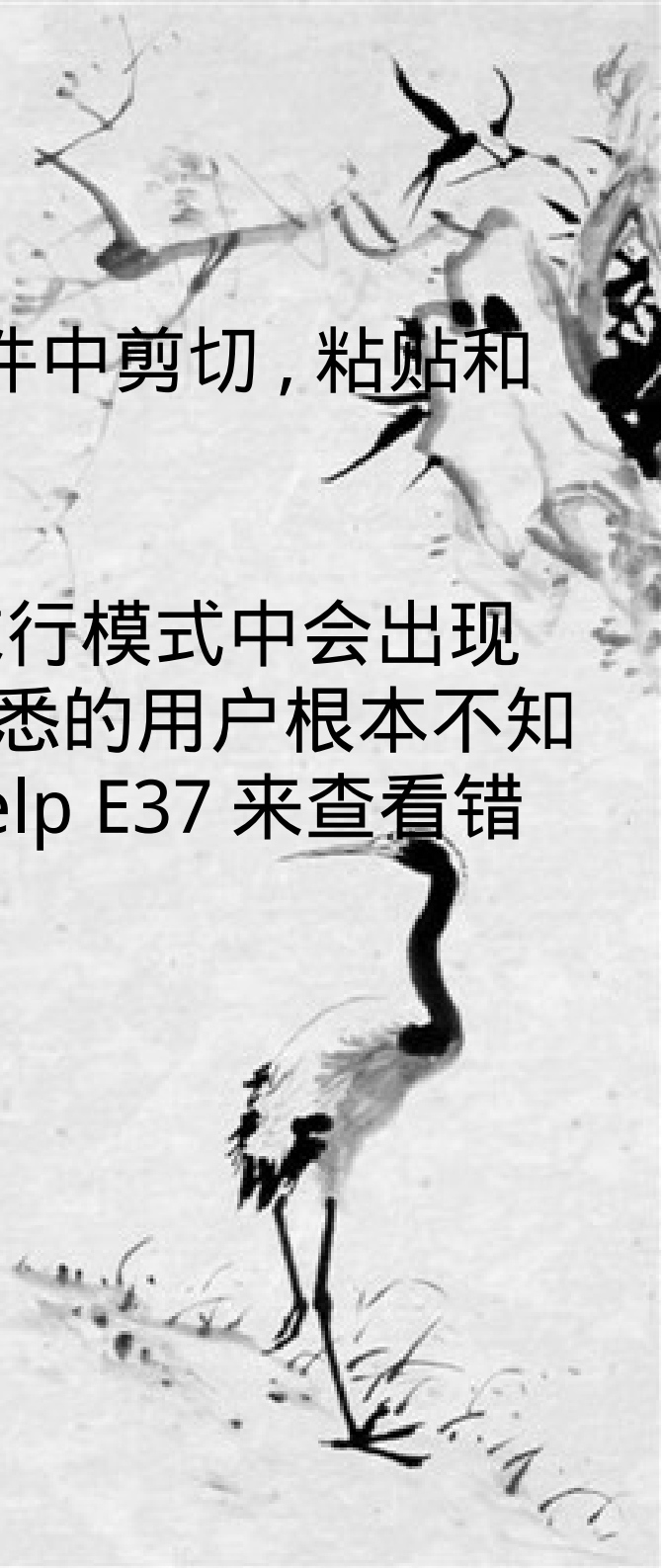
GNU/Linux--VIM

在编辑一个文件的时候需要查看此文件的其他内容，在 vi 中有一个非常好的功能 -- 分屏，分屏功能是将现在屏幕拆分成多个窗口

:split	将屏幕分成多个窗口
ctrl+w + or -	用来调整窗口的大小
ctrl+w 方向键	切换窗口
:close	关闭光标所在的窗口

有了这些功能编辑和修改文档是不是容易多了 ~!

GNU/Linux--VIM



有了分屏功能以后，就可以实现在多个文件中剪切，粘贴和复制了

:edit　　在窗口中打开一个文件

:help　　查看帮助文档，有的时候在末行模式中会出现一些错误的代码，比如 E37 如果对 vi 不熟悉的用户根本不知道错误代码的含义，这个时候就可以用 :help E37 来查看错误代码的具体含义

GNU/Linux--VIM

字串的查找和替换

在末行模式下的查找和替换功能是非常强大的

查找格式

: 范围　命令 / 查找字串 / 替换字串 / 参数

范围

%	所有的行
$	文件最后一行
.	光标所在行
1,50	1-50 行

GNU/Linux--VIM

命令：
查找的命令全称是 substitute
但为了方便期间可以使用简写 sub 或者干脆用 s
这三个单词都可以用做查找的命令
查找：
查找的内容中可以使用正则表达式 ,ragular expression , 这样可以让查找方式非常灵活多变

GNU/Linux--VIM

替换的内容当然就是替换的内容了
最后的参数有很多，可以同时使用多个参数

g	替换所有查找到的内容
c	每次替换的时候都手工确认 y 同意 n 不同意仍然继续替换， q 退出替换
i	忽略大小写
I	不忽略大小写
e	不显示出错信息

GNU/Linux--VIM

下面来看几个查找替换的实例
替换所有的 this 为 that
:% s/this/that/g
将文件中所有 /usr/bin 目录替换成 /home
:% s/\/usr\/bin/\/home/g
在每行开头加入一个单词 linux
:% s/^/linux/g
在每个单词后面加上一个 s
:% s/$*\>/s/g

GNU/Linux--VIM

末行模式还支持另外一项功能，查找到某个匹配的字符以后执行指定的命令
: 范围 global/ 查找字串 / 命令

范围和前边提到的范围相同

global　　命令，也可以用 g 来代替，用来查找需要查找的字串 (可以使用正则表达式)

命令　　用的是命令模式中的命令如 dd

示例查找到 this 字串执行 d 命令
:% g/this/d

GNU/Linux--VIM

vi 可以根据用户的不同需要来做一些设定，这些设定都是在末行模式下进行的。

:set nu　　显示行号，打开这个功能以后会在每一行的最左面显示行号，行号不算在文件本身

:set nonu　　关闭显示行号的功能

:set nohlsearch　消除搜索的记号

:set ic　　忽略大小写，主要是为了方便搜索

:set noic　　不忽略大小写

GNU/Linux--VIM

:syntax on
打开色彩支持 , 在 linux 中编辑文件和编辑程序源代码等工作都是在 vim 中完成的 , 打开色彩支持可以在查看或编写程序的时候发现语法等错误 .
:syntax off　关闭色彩支持

:set backup
自动生成备份文件 ,vi 在打开或编辑一个文件的时候会自动备份文件 , 备份的件一般会在文件名后加 ~ 。 例如 : foo.txt 会自动生成 foo.txt~

:suspend　　把 vi 暂时放到后台休息 . 用 fg 恢复 (^z)

GNU/Linux--VIM

定义快捷键，在末行模式下有一个 map 功能这个是用来给用户自定义快捷键用的

定义格式：　map　　F2-F4　　执行命令
使用方法，在 vim 中执行
:map　　<F2>　　dd　　将 F2 功能键定义为 dd 作用
退出末行模式，然后到命令模式下试试 F2
注：这里定义的快捷键在退出 vim 之后将会失效

GNU/Linux--VIM

如果在使用 vim 编辑的时候需要执行一下系统的命令可以在末行模式中用 ! 来执行

示例： 在 vim 编辑器中执行 ls 命令
:! ls
执行完系统命令以后再按回车就可以回到 vim 编辑器中

GNU/Linux--VIM

为了更方便的使用 vim 可以在个人主目录下可以编辑用户主目录下的 .vimrc 文件，如果没有可以手工创建一个 .

格式如下：

```
set nu    // 打开文件的时候显示行号
set ic    // 查找字串的时候忽略大小写
set tabstop=2  // 用 tab 键时缩进每次宽度为 2
syntax enable          // 打开色彩支持
```

GNU/Linux--VIM

在命令行中进行文件比较的命令为 vimdiff：

$vimdiff file_C file_A

在 vi 中进行文件比较的操作如下：

$vi file_A file_C

:diffsplit file_A~:

GNU/Linux--VIM

缓存区

打开多个文件时，你也就打开了多了缓冲区。缓冲区的两个格式：隐藏的和活动的

列出两个缓冲区的命令：

:buffers 和 :ls　第一个文件　第二个文件

第一个缓冲区入口的标志 %a 表示文件 file_A 的缓冲区当前处于打开状态。

GNU/Linux--VIM

其他的一些标志：

% 当前缓冲区

置换缓冲区

a 正在使用并处于显示的缓冲区

h 正在使用的隐藏缓冲区

= 只读缓冲区

- 缓冲区不可修改或可修改模式被关闭

+ 缓冲区已经被修改

GNU/Linux--VIM

正在处于打开状态的缓存区间正在切换用 :buffers 命令 .

编辑缓存区，三种方式：

缓存区标号： :buffer 1

缓存区名： :buffer file_A

部分缓存区名： :buffer_ab

GNU/Linux--VIM

对缓存区进行操作的命令：

:bnext　　下一个缓冲区

:bprevious　前一个缓冲区

:bfirst　　转到第一个缓冲区

:blast　　转到最后一个缓冲区

删除缓冲区命令：　:bdelete　缓存区标号

例：

bdelete 2　就是删除第 2 个缓存区。

GNU/Linux--VIM

如果对 vim 使用非常感兴趣，可以参看 vim 教程

请执行一下命令观看教程

#vimtutor

或使用中文教程

#vimtutor zh_CN

GNU/Linux--gedi

图形编辑器

帮助文件 :

#yelp help:gedit