目录

1.1 VI使用

1.1.1 VI用法总汇

vi(Visual)是以视觉为导向的全屏幕编辑器、共分为三种方式（mode）：

command 方式

任何输入都会作为编辑命令，而不会出现在屏幕上，若输入错误则有“岬”的声音；任何输 入都引起立即反映

insert 方式

任何输入的数据都置于编辑寄存器。在 command 方式下输入（I,a,A 等）,可进入 insert 方 式，insert 方式下按 ESC，可跳回 command 方式。

escape 方式

以“:”或者“/”为前导的指令，出现在屏幕的最下一行，任何输入都被当成特别指令。

vi 基本用法请参考下表

功能	命令	含义
进入 vi	vi	进入 vi 而不读入任何文件
	vi filename	进入 vi 并读入指定名称的文件（新、旧文件均可）
	vi +n filename	进入 vi 并且由文件的第几行开始
	vi +filename	进入 vi 并且由文件的最后一行开始
	vi + /word filename	进入 vi 并且由文件的 word 这个字开始
	vi filename(s)	进入 vi 并且将各指定文件列入名单内，第一个文件先读入
	vedit	进入 vi 并且在输入方式时会在状态行显示“INSERT MODE"
编辑数个文件(利用 vi filenames 进入 vi 后)	:args	显示编辑名单中的各个文件名
	:n	读入编辑名单中的下一个文件

	:rew	读入编辑名单中的第一个文件
	:e#	读入编辑名单内的前一个文件
	:e file	读入另一个文件进 vi(此文件可不在编辑名单内),若原文件经修改还 没有存档,则应先以: w 存档
	:e! file	强迫读入另一个文件进入 vi，原文件不作存档动作
存储及退出	:w filename	存入指定文件，但未退出 vi（若未指定文件名则为当前工作的文件 名）
	:wq 或者 :x 或者 zz	存文件，并且退出 vi
	:q	不作任何修改并退出 vi
	:q!	放弃任何修改并退出 vi
	:!command	暂时退出 vi 并执行 shell 指令，执行完毕后再回到 vi
	:sh	暂时退出 vi 到系统下,结束时按 Ctrl + d 则回到 vi
加数据指令	i	在关标位置开始插入字符，结束时候按 ESC 键
	I	在光标所在行的最前面开始加字，结束时按 ESC 键
	a	在光标位置后开始加字，结束时按 ESC 键
	A	在光标所在行的最后面开始加字，结束时按 ESC 键
	o	在光标下加一空白行并开始加字，结束时按 ESC 键
	O	在光标上加一空白行并开始加字，结束时按 ESC 键
	!command	执行 shell 指令，并把结果加在光标所在行的下一行
删除指令	nx	删除由光标位置起始的 n 个字符（含光标位置，按一个 x 表示删除光标 所在的字符）
	nX	删除由光标位置起始的 n 个字符（不含光标位置）
	ndw	删除光标位置其实的 n 个字符组（word）
	d0	将行的开始到光标位置的字符全部删除

	d#或 D	将光标位置起始到行尾的字符全部删除
	ndd	将光标位置起始的 n 行（整行）删除（dd 表示删除光标所在行）
	:start,endd	删除文件的第 start 到 end 行
光标移动	0	移到一行的开始
	$	移到一行的最后
	[	移到文件开始位置
	]	移到文件结束位置
	nh	往左移 n 位
	nl 或者 spacebar	往右移 n 位
	nk	向上移 n 行
	n+	向上移 n 行，光标在该行的起始
	ni	向下移 n 行
	n-	向下移 n 行，光标在该行的起始
	H	移到屏幕的左上角
	M	移到屏幕的中间行开头
	L	移到屏幕的最后一行
	G	移到文件的最后一行
	nG 或者:n	移到文件的第 n 行
	nw	右移 n 个字组，标点符号属于字组
	nW	右移 n 个字组，标点符号不属于字组
	nb	左移 n 个字组，标点符号属于字组
	nB	左移 n 个字组，标点符号不属于字组
	Ctrl + u	屏幕上卷半个菜单
	Ctrl + d	屏幕下卷半个菜单
	Ctrl + b	屏幕上卷一个菜单
	Ctrl + F	屏幕下卷一个菜单
修改指令	r	修改光标文件的字符
	R	从光标位置开始修改，结束时按 ESC 键
	new	更改 n 组字符，结束时按 ESC 键
	ncc	从光标所在位置开始更改 n 行，结束时按 ESC 键

重排	i	并按 Enter 将该行由光标所在处断开,并进入 insert 方式
	J	把下一行的数据连接到本行之后
寻找指令	/text	从光标位置往下找字串 text
	?text	从光标位置往上找字串 text
	n	继续找下一个字串（在输入上面的寻找指令之后使用）
	/\<test\>	精确定位字符串
寻找并且取代指令	:getxt1/s/ /text2/options	将各行的 text1 替换为 text2 option=g 表示文件中所有的 text1 均被取代，若未输入任何 option,则只有各行中的第一个出现的 text1 被取代 option=go 在屏幕显示各取代的行 option=gc 在每个字串取代之前要求确认
	Start,endgtext1/s/ / text2/options 或:Start,ends/text1/text2/options	同上，只寻找并取代第 start~end 行
复制及移动文件	:first,last co dest	将 first 到 last 行的数据复制到目标行(dest)下面
	:Start,end m dest	将 start 到 end 行的数据移动到目标行（dest）下
	:r filename	将指定文件的内容读入光标所在行下
	nY	将光标所在位置开始的 n 行数据暂存
	p	复制暂存数据在光标的下一行
	P	复制暂存数据在光标的上一行
其他命令	.	重复前一指令
	u	取消前一指令
	Ctrl + l	刷新屏幕显示
	:set number	显示文件的行号，但不会存文件

	:set nonumber	解除行号显示
	:set ai	设置每行起始位置（以光标当前位置为起始）
	:set noai	取消行起始位置设定
	:f 或<Ctrl> + g	告诉用户有关现行编辑文件的数据

1.1.2 如何在VI中进行批量替换

批量替换操作很容量在 vi 中实现,当 vi 处在命令状态时输入：

:s/*old*/*new*/g （替换当前行）

:%s/*old*/*new*/g（替换全文）

:[范围]s/*old*/*new*/[cgi],(c-替换时提示,g-全局替换,i-忽略大小写)其中”范围”取值可为：

%:当前文件

$:文件末尾或某一行中的最后

^:文件首行或某一行中的最前

.,+5:表示下面 5 行

另外,vi 中的搜索字串可以使用正规表达式.支持以下正则表达式：

& 所有查找时匹配到的东西

\[1-9] 1 到 9 号用\(和\)括起来的东西

\u 下一个字符将被变成大写.

\U 以后的字符都变成大写,直到遇到\e 或\E

\l 下一个字符将被变成小写.

\L 以后的字符都变成大写,直到遇到\e 或\E

\[Ee] 更改大小写的选择区域的终点

vi 批量替换操作请参考 vi 实例.

1.1.3 如何定制vi

vi 编辑器利用”.exrc”文件来定制其相关特性,该文件一般位于用户的主目录下.当用户打开 vi 时,vi 在该用户的主目录下寻找是否存在”.exrc”文件,如果存在,则使用该文件中所定义的参数来定制 vi.以下是该文件的一个例子：

#more /.exrc

set nu （显示行号）

set ai （自动缩进）

set showmode （显示当前所处于的模式,如替换/插入模式）

set ts=4 (将 tabstop 的宽度设为 4,默认为 8)

set sw=4 (将 shiftwidth 的宽度设为 4,默认为 8)

1.1.4 vi实例

- 将文件内容反转

利用 vi,可以将某个文件的内容前后反转,即文件中最后一行变成首行中的内容,第一行的内容则移到文件末尾.

在 vi 的命令模式下输入指定:

```
:g/^/m0
```

- 对文件的内容进行排序(包括对数字及字符):

```
:%!sort (其中"%"表示全文)
```

- 精确查找单词:

```
/\<hello\>
```

上述指令只查找"hello",不会搜索到"hello_world"的字符串,对批量替换很有用.

- 上下两行调换

在 vi 命令模式下,键入指令:

```
ddp
```

- 批量替换

```
:g/\(foo\)\(bar\)/s/\2/\1baz/g 将 foobar 替换成 foobaz
:%s/.*/printf("& is:%d\\n",&)/ (其中"&"为查找到的内容)
:%s/\t/ctrl+v+m/g (其中"ctrl+v+m"同时按下,产生^M,即回车符,将文件中的 TAB 键转换成回车)
```

- vi 应用于代码定位

当使用vi编写C代码时,如果有多个函数位于不同的源文件中,这时想查看函数的内容就比较麻烦,通过以下操作将简化操作:

step1:利用"ctags"命令对当前目录下的 C 源程序扫描,生成 tags 文件.

step2:在 vi 的命令模式下直接定位函数

```
:ta function_name
```

这样,就可以直接打开函数 function_name 所在的文件,并将光标置于函数的开头.