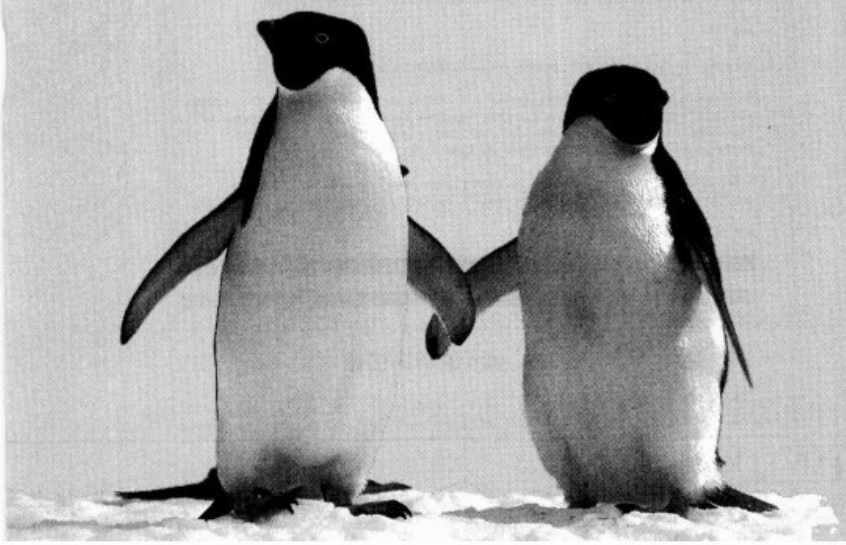
LINUX

第6章 编辑文件

在 UNIX 系统 vi 编辑器的基础上，Linux 系统提供了克隆版的 vim 编辑器（参见 http://www.vim.org）等。vim 编辑器是对 vi 的扩充与增强，提供了许多附加的功能特性，且与 vi 几乎是完全兼容的。vim 可以运行在许多平台上，包括 Windows、Macintosh、UNIX 和 Linux 系统。利用 vim，可以编辑文件，开发应用程序。本章讨论的主要内容包括：

- 启动 vim 编辑器；
- vim 编辑器的两种工作模式；
- 保存编辑的文件并退出 vim；
- vim 编辑器的基本命令；
- 使用 ex 命令；
- 检索与替换；
- 编辑多个文件；
- 定制 vim 编辑器的运行环境；
- 其他特殊说明；
- vim 编辑器命令总结。



vim 是一个极其强有力的文本编辑工具，如果能够掌握其规律，它将是一个非常好的开发工具。但在使用 vim 编辑器时，用户需要记住编辑器当前所处的工作模式。即便如此，只要经过一定的实践，很快就会熟练地掌握 vim，得心应手地处理文本文件。vim 提供了大量的命令供用户编辑文件。本章将按照操作类型，介绍最基本的 vim 命令。

6.1 启动 vim 编辑器

6.1.1 创建文件

在 Linux 系统的命令提示符下输入下列命令，即可启动 vim 编辑器。

```
$ vim myfile
```

如果名为 myfile 的文件存在，上述命令将会打开指定的文件，同时在编辑窗口中显示文件第一页的数据内容。如果指定的文件不存在，vim 将会打开一个新文件，出现图 6-1 所示的编辑窗口。

图 6-1 vim 编辑器

此时，光标将处于编辑窗口左上角的位置，屏幕左边的波浪符“～”表示空行，说明这是一个空文件。注意，启动 vim 时可以同时指定多个文件名参数，意味着同时编辑多个文件；也可以不指定文件名，等到完成文件编辑之后再使用“w”命令写入一个新文件，然后退出 vim。

如果运行 vim 命令时未指定文件的名字，将会显示图 6-2 所示的内容。此时，可以直接编写文件，等完成之后再使用“:w filename”命令保存写就的文件，然后使用“:q”命令退出 vim。初学者也可以使用“:help”命令获取 vim 的帮助信息，如图 6-3 所示。

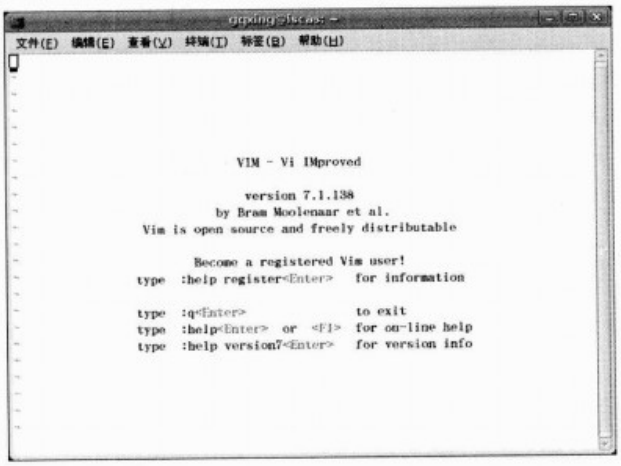


图 6-2 未指定文件名字的 vim 编辑器

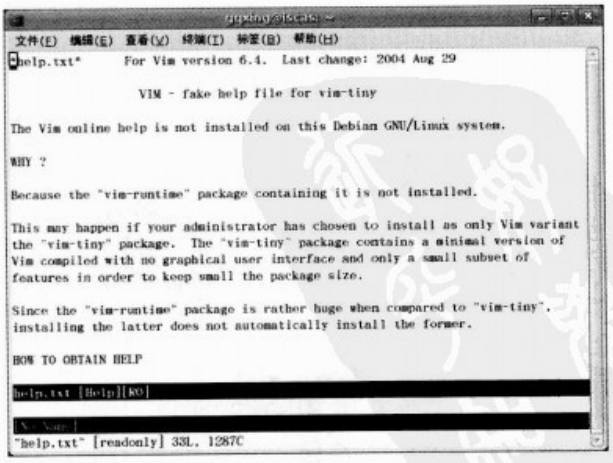


图 6-3 vim 的帮助信息

6.1.2 状态行

编辑窗口的最后一行是 vim 的状态行，用于显示编辑器的状态、编辑过程中出现的出错信息、光标所在的行列位置、删除或复制的行数等。初始启动时，状态行将会显示文件的名字、文件的行数以及文件的字节计数。在图 6-1 中，状态行表示打开的是一个新（空）文件。

6.2 vim 编辑器的两种工作模式

vim 分为命令和输入两种工作模式。任何时刻，vim 编辑器总是处于命令和输入两种工作模式之一。当启动 vim 编辑器、打开或创建一个文件时，vim 即处于命令模式。通过发布 vim 命令，可使 vim 处于输入模式。在输入模式下，可以输入数据，编写自己的应用程序；而在命令模式下，可以输入 vim 命令，执行特定的 vim 编辑功能。命令模式是 vim 的默认工作模式。通过使用 vim 命令和 Esc 键，可以在两种工作模式之间相互转换。

在某些情况下，vim 并不提示编辑器当前所处的工作模式，因此对于 vim 的新用户来讲，区分命令模式与输入模式也许是最感困惑或茫然的问题。但只要记住一点，就可以做到心中有数。即无论何时，只要按下 Esc 键，不管 vim 当前处于何种工作模式，总是进入命令模式。因此，多敲几次 Esc 键是 vim 用户的常见动作。

第一次使用 vim 打开文件时，vim 总是处于命令模式。在能够输入任何文本之前，首先必须输入 vim 的数据输入命令。例如，输入“i”（“插入”）字符命令，即可在当前光标所处字符位置之前插入数据；输入“a”（“附加”）字符命令，即可在当前光标所处字符位置之后附加数据。本章的后面将详细介绍各种数据输入命令。

无论何时需要返回命令模式，按下 Esc 键即可。如果不能确定 vim 当前处于何种工作模式，只要按下 Esc 键，即可确保 vim 总是处于命令模式，然后再决定下一步怎么做。如果在 vim 处于命令模式时按下 Esc 键，或者按下其他不合法的键时，终端将会发出鸣叫或发生屏幕闪烁现象，但不会影响正在编辑的文件。

6.2.1 输入模式

为了在前面打开的 myfile 示例文件中输入数据，可输入 vim 的“插入”命令“i”。这一命令将使 vim 编辑器从命令模式转入输入模式。

现在，即可尝试输入几行数据，在每行数据输入结束后按下 Enter 键。在输入数据的过程中，可以利用退格键做简单的校正（在按下 Enter 键之前）。在整个输入过程结束之后，按下 Esc 键，即可返回命令模式。此时，光标将处于刚才输入的最后一个字符的位置。然后，还可以利用各种 vim 命令，对输入的数据进行校正。

6.2.2 命令模式

如上所述，当利用 vim 打开一个文件时，vim 将处于命令模式。在命令模式下，可以输入各种 vim 命令，以便完成各种编辑功能。vim 命令几乎都是由一个字符、两个字符或一个选用的数

字加字符组成的。通常，同一字母但大小写不同的字符命令，其意义是相同的，但其作用则完全不同。例如，字符命令“a”意味着在当前光标所处字符位置之后附加数据，而“A”则意味着在当前光标所在行的行尾附加数据。

大多数 vim 命令不需要按 Enter 键即可立即执行，但是，以冒号“:”开始的命令需要在输入命令之后再按 Enter 键。在命令模式下输入冒号“:”时，“:”将会出现在编辑窗口底部最后一行的左下角，然后即可接着输入编辑命令。

以冒号开始的命令实际上是 ex 命令。ex 与 vim 命令是同一编辑程序的两个不同的用户界面，vim 提供面向屏幕的用户界面，而 ex 则提供面向命令行的用户界面。所有的 ex 命令均可在 vim 中使用。在输入冒号之后，实际上已经切换到面向命令行的 ex 用户界面。这种切换方式使用户能够在不离开 vim 的情况下执行许多文件编辑命令，甚至可以执行其他 Shell 命令（参见 6.5 节）。

6.3 保存编辑的文件并退出 vim

在使用 vim 编辑文件期间，用户所做的任何编辑处理并未直接反映到实际的文件中。实际上，整个编辑过程将被保存到 vim 于内存中临时创建的一个文件副本中。仅当发布“w”（“写”）等命令时，内存缓冲区中的内容才能永久性地被保存到磁盘上的文件中。

vim 编辑器的这种处理方式既有积极的一面，也有丢失数据之忧。可取之处是在退出文件编辑时，用户可以放弃编辑期间所做的任何修改，而不影响原有的文件。缺点是当系统发生故障时，很有可能丢失内存缓冲区中保存的数据。

因此，最好的做法是注意随时保存数据，特别是当编辑重要的文件时。

vim 编辑器提供了许多命令，用于把内存缓冲区中的数据内容保存到磁盘文件中，然后退出 vim 编辑器。这些命令允许用户选择“保存并退出”、“强制退出而不保存”等编辑器退出方式（参见表 6-1）。

表 6-1　　vim 的保存文件和退出命令

命　令	简 单 说 明
:w	保存编辑后的文件内容，但不退出 vim 编辑器。这个命令的作用是把内存缓冲区中的数据写到启动 vim 时指定的文件中
:w!	强制写文件，即强制覆盖原有的文件。如果原有文件的访问权限不允许写入文件，例如，原有的文件为只读文件，则可使用这个命令强制写入。但是，这种命令用法仅当用户是文件的属主时才适用，而超级用户则不受此限制
:wq	保存文件内容后退出 vim 编辑器。这个命令的作用是把内存缓冲区中的数据写到启动 vim 时指定的文件中，然后退出 vim 编辑器。另外一种替代的方法是用 ZZ 命令
:wq!	强制保存文件内容后退出 vim 编辑器。这个命令的作用是把内存缓冲区中的数据强制写到启动 vim 时指定的文件中，然后退出 vim 编辑器
ZZ	使用 ZZ 命令时，如果文件已经做过编辑处理，则把内存缓冲区中的数据写到启动 vim 时指定的文件中，然后退出 vim 编辑器。否则只是退出 vim 而已。注意，ZZ 命令前面无需加冒号“:”，也无需按 Enter 键
:q	在未做任何编辑处理而准备退出 vim 时，可以使用此命令。如果已做过编辑处理，则 vim 不允许用户使用“:q”命令退出，同时还会输出下列警告信息： No write since last change (:quit! overrides)
:q!	强制退出 vim 编辑器，放弃编辑处理的结果。如果确实不需要保存修改后的文件内容，可输入“:q!”命令，强行退出 vim 编辑器
:w *filename*	把编辑处理后的结果写到指定的文件中保存
:w! *filename*	把编辑处理后的结果强制保存到指定的文件中，如果文件已经存在，则覆盖现有的文件
:wq! *filename*	把编辑处理后的结果强制保存到指定的文件中，如果文件已经存在，则覆盖现有的文件，并退出 vim 编辑器

6.4 vim 编辑器的基本命令

本节将分类介绍 vim 提供的各种编辑命令：

- 移动光标位置；
- 输入文本数据；
- 修改和替换文本；
- 撤销先前执行的文本编辑命令；
- 删除文本；
- 重复执行先前的命令。

注意，vim 命令是严格区分大小写字母的。即使命令形式完全相同，如果不注意区分大小写字母，将会产生完全不同的效果。

6.4.1 移动光标位置

当启动 vim 编辑器时，光标将处于编辑窗口左上角的位置，并处于命令模式。在命令模式下，可以使用表 6-2 所示的字符命令移动光标位置。

表 6-2 光标移动命令

命　令	简 单 说 明
←↑↓→	方向箭头键。可在编辑窗口上将光标左移、上移、下移和右移一个字符（行）位置
h k j l	其功能与箭头键完全相同。在无法使用箭头键（如远程访问）时，可以使用这 4 个键分别替代相应的箭头键
-	把光标移至上一行的第一个起始字符位置（第一个非空白字符位置）
Enter 键	把光标移至下一行的第一个起始字符位置（第一个非空白字符位置）
退格键	光标左移一个字符位置
空格键	光标右移一个字符位置
Ctrl-F	往下（文件结尾方向）滚动一屏。在命令方式下按 Ctrl-F 键，编辑窗口中将会显示文件下一页的内容，光标也同时移至下一页的左上角位置
Ctrl-B	往上（文件开始方向）滚动一屏。在命令方式下按 Ctrl-B 键，编辑窗口中将会显示文件前一页的内容，光标也同时移至前一页的左下角位置
Ctrl-U	往下滚动半屏
Ctrl-D	往上滚动半屏
Ctrl-E	编辑窗口中的文件内容整体上移一行
Ctrl-Y	编辑窗口中的文件内容整体下移一行
H	把光标移至编辑窗口顶部第一行的起始字符位置（第一个非空白字符位置）
M	把光标移至编辑窗口中间一行的起始字符位置（第一个非空白字符位置）
L	把光标移至编辑窗口底部最后一行的起始字符位置（第一个非空白字符位置）
w	光标右移一个字。如果相邻的两个字之间有标点符号，光标将移至标点符号位置

续表

命　令	简 单 说 明
W	光标右移一个字。即使相邻的两个字之间有标点符号，也忽略之
b	光标左移一个字。如果相邻的两个字之间有标点符号，光标将移至标点符号位置
B	光标左移一个字。即使相邻的两个字之间有标点符号，也忽略之
e	把光标移至当前字（或下一个字）的最后一个字符位置
E	同上，只是以空格字符作为字的分隔符
^	把光标移至当前行的起始位置，即当前行的第一个非空白字符位置
0（零）	同上
$	把光标移至当前行的行尾，即当前行的最后一个字符位置
[*n*]G	将光标移至指定行的行首位置。其中 n 表示行号，默认情况下转至文件的最后一行。因此，要转到当前文件的最后一行，只需输入“G”命令。要移至文件的第一行，可输入“1G”命令。实际上，利用“nG”命令，可以转至当前文件的任何一行。例如，要修改文件的第 60 行，输入“60G”命令，即可立即跳转到第 60 行的行首位置
(	把光标移至一个完整句子的句首位置
)	把光标移至一个完整句子的句尾位置
{	把光标移至一个完整段落的段首位置
}	把光标移至一个完整段落的段尾位置

6.4.2 输入文本

vim 提供了许多命令，使用户能够输入文本数据。表 6-3 给出了 vim 编辑器中最常用的几种数据输入命令。注意，这些命令将使 vim 进入数据输入方式，同时在编辑窗口的左下角（即状态行左边）显示“— INSERT —”状态信息，表示 vim 当前正处于输入模式。为了使用这些命令，首先必须确保 vim 处于命令模式（多按几次 Esc 键总能保证 vim 处于命令模式）。

表 6-3　　数据输入命令

命　令	简 单 说 明
a	使用“a”命令，可在光标当前所在字符位置之后输入数据，输入结束后可按 Esc 键退出输入方式。在发布“a”命令之前，光标可移至文本行的任何位置。发布“a”命令之后，输入的数据多少不限
A	使用“A”命令，可在光标当前所在行的行尾（即在最后一个字符之后）输入数据，输入结束后可按 Esc 键返回命令模式。在发布“A”命令之后，不管光标处于何位置，都会移至当前文本行的行尾，此时可输入任何数量的数据，直至按下 Esc 键
i	使用“i”命令，可在光标当前所在字符位置之前输入数据，输入数据的数量不限。输入结束后可按 Esc 键退出输入方式
I	使用“I”命令，可在光标当前所在行的行首（即在第一个非空白的起始字符之前）输入数据，不管光标之前处于何位置，都会移至当前文本行的行首，输入数据的数量不限，输入结束后可按 Esc 键退出输入方式
o	使用“o”命令，可在光标当前所在行之后插入数据，行数不限，直至按下 Esc 键结束
O	使用“O”命令，可在光标当前所在行之前插入数据，行数不限，直至按下 Esc 键结束

6.4.3 修改与替换文本

为了修改或替换文本数据，vim 提供了若干不同的命令，用于校正文本数据。用户可以根据

具体情况，灵活地予以选用（参见表 6-4）。在表 6-4 所列的编辑命令中，除了“r”和“～”命令不显示任何信息，“R”命令将会在编辑窗口的左下角显示“— REPLACE —”状态信息，其他命令均显示“— INSERT —”状态信息，表示 vim 当前正处于输入模式。

表 6-4　　修改与替换命令

命　令	简 单 说 明
C	替换从光标位置开始直至行尾的所有数据内容，以 Esc 键结束
cw	替换单个字。要替换一个整字，可将光标移至准备替换的字的字首位置，然后输入“cw”命令，接着输入新的文字。输入的字符数量不限，输入结束后可按 Esc 键返回命令模式。要修改单字的一部分，可将光标移至单字保留部分右边的字符位置，输入“cw”命令后可对该字进行校正，结束后按 Esc 键返回命令模式
[*n*]cc	替换行。要替换一行文本，只需把光标移至目标行的任何字符位置，然后输入“cc”命令。此时，当前文本行将会消失，留下一个空行位置，等待用户输入新的文本数据。此时可以输入任何数量或行数的数据。输入结束后，按 Esc 键可返回命令模式。要替换多行文本，可把光标移至目标行的第一行，然后输入“*n*cc”命令，其中 *n* 表示需要替换的文本行的数量
[*n*]s	替换字符。要替换光标当前所在位置的一个字符，可输入“s”命令，然后可以输入任何数量或行数的数据。输入结束后，按 Esc 键可返回命令模式。要替换从光标位置开始的多个字符，可输入“*n*s”命令，其中的 *n* 表示需要替换的字符的数量
S	替换当前行。要替换光标当前所在的行，可输入“S”命令，然后可以输入任何数量或行数的数据。输入结束后，按 Esc 键可返回命令模式
r	替换单个字符。使用“r”命令，可用随后输入的字符替换光标位置的单个字符。与“s”命令不同，“r”命令只能替换一个字符，替换后，vim 编辑器自动返回命令模式（不需要再按 Esc 键）
R	替换多个字符。使用“R”命令，可以从光标位置开始替换多个字符，数量不限，直至按下 Esc 键结束
[*n*]～	转换光标当前所在位置字母的大小写。一直按住波浪号“～”键能够连续转换多个字母的大小写。也可以在输入“～”之前输入一个数字，一次转换多个字母的大小写

6.4.4　撤销先前的修改

在编辑文本期间，以及准备把加工后的数据保存到文件之前，有时可能需要放弃某些编辑处理的结果。vim 的撤销命令能够撤销先前执行的编辑命令的处理结果，使 vim 回退到先前的处理状态。然后，可从先前的处理位置开始继续进行编辑加工。撤销命令及其说明如表 6-5 所示。

表 6-5　　撤销命令及其说明

命令	简单说明
u	用于撤销先前执行的编辑命令。如果在编辑过程中出现失误，或者编辑后又改变了决定，可以使用 vim 的“u”命令，撤销先前执行的编辑命令。注意，输入“u”命令后不需要按 Esc 键。连续输入“u”命令后能够以回溯的方式，依次撤销先前执行的所有编辑命令
U	撤销或恢复对当前文本行所做的全部编辑处理。使用 vim 的“U”命令可以撤销或恢复最近一次编辑处理的文本行。也就是说，“U”命令仅适用于最近一次修改的文本行。同样，输入“U”命令后也不需要按 Esc 键

6.4.5　删除文本

利用表 6-6 给出的 vim 命令，可以删除指定的字符、字或文本行数据。

表 6-6　　删除命令及其说明

命令	简 单 说 明
[*n*]x	删除字符。要删除单个字符，可将光标移至准备删除的字符位置，然后输入“x”命令。“x”命令除删除指定的字符之外，还将删除字符占用的空间位置——当从某个单字中间删除一个字符时，余下的字符将合并为一个新的字，中间不会留下任何间隙。利用“x”命令，也可以删除文本行中的空格字符。要删除多个字符，可将光标移至准备删除的字符串起始位置，然后输入“*n*x”命令，其中的 *n* 表示字符的数量
[*n*]X	删除字符。要删除光标当前所在位置的前一个字符，可以使用大写的“X”命令。要删除多个字符，可将光标移至准备删除的字符串的右边，然后输入“*n*X”命令，其中的 *n* 表示字符的数量
dw	删除单个字或部分字。要删除一个整字，可以把光标移至该字的起始字符位置，然后输入“dw”命令相应的字及其占用的空间位置将一并被删除。要删除单字的右边部分，可将光标移至该字要保留部分的后面，输入“dw”命令，即可删除单字的右边部分
[*n*]dd	删除文本行。要删除整个文本行，可以把光标移至文本行的任何位置，然后输入“dd”命令，整个文本行及其占用的空间将会一并被删除。要同时删除多个文本行，可将光标移至准备删除的第一个文本行，然后输入“*n*dd”命令，其中的 *n* 表示准备删除的行数（包括当前行在内）
D	删除文本行的行尾部分。要删除文本行右边的部分文字，可将光标移至文本行要保留部分的后面，输入“D”命令，即可删除文本行的行尾部分

6.4.6　复制、删除与粘贴文本

许多字处理软件都提供“复制—粘贴”与“剪切—粘贴”的文本行处理方式。vim 编辑器也提供这样的功能。在 vim 编辑器中，与“复制—粘贴”等价的处理过程是先用“yy”命令复制文本行，接着再用“p（或 P）”命令实现文本行的实际复制；与“剪切—粘贴”等价的处理过程是先用“dd”命令删除文本行，接着再用“p（或 P）”命令实现文本行的移动。在上述两种组合方式的基础上，如果在“yy”或“dd”命令之前再输入适当的数字，还可以实现整块文本行的复制和移动处理（参见表 6-7）。

表 6-7　　复制与粘贴命令

命　令	简 单 说 明
[*n*]yy	复制文本行。实际上，“yy”命令只是把文本行的数据内容保存到粘贴板中。一个复制动作需要同时使用两个命令才能完成，即“yy”与“p（或 P）”命令。 因此，为了复制一个文本行，可按下列步骤执行： (1)　把光标移至准备复制的文本行的任何位置； (2)　输入“yy”命令； (3)　再把光标移至目标行的任何位置； (4)　输入“p”命令，将粘贴板中的数据内容复制到光标所在行的下面； (5)　输入“P”命令，将粘贴板中的数据内容复制到光标所在行的上面。 如果在输入“yy”命令之前输入数字，则可以同时复制多个文本行。例如，如果想要复制 10 行数据，可输入“10yy”命令，这将把从光标当前所在行开始的 10 行数据复制到粘贴板中。此时，vim 编辑器将会在编辑窗口底部显示一条信息“10 lines yanked.”，表示命令已成功地执行
[*n*]Y	其功能同“yy”命令
[*n*]dd	删除文本行。为了把一个或若干文本行移至某个位置，需要先删除文本行，然后再粘贴到适当的位置，因此也需要同时使用两个命令才能完成，即“dd”与“p（或 P）”命令 例如，为了移动 5 行数据，可以把光标移至要删除文本行的任何位置，发布“5dd”命令，然后把光标移至插入位置，接着输入“p（或 P）”命令，即可把文本行移至当前行的下方（或上方）
p（小写）	把粘贴板中的文本数据复制到光标所在行的下面
P（大写）	把粘贴板中的文本数据复制到光标所在行的上面

6.4.7 按指定的数量重复执行命令

许多 vim 命令前面都可以加一个计数值，表明相应的命令需要重复执行的次数。在前面的讨论过程中，实际上已经用到了这样的命令。例如，“3dd”命令意味着删除文本行的动作需要执行 3 次，最终结果是删除 3 行数据，“2dw”命令意味着删除两个字，“4x”命令意味着删除 4 个字符（包括空格）。

另外，也可以使用计数命令方式移动光标。例如，“3w”命令意味右移 3 个字，2Ctrl-F 意味往前滚动两屏。

使用句点“.”命令也可以重复执行先前的文本编辑命令。例如，如果用户刚刚使用“dd”命令删除了一个或多个文本行，此时可以把光标移至准备删除的文本行中，仅仅输入一个句点“.”命令即可重复执行删除文本行的处理动作。对于复杂的编辑命令，句点命令尤其方便。

6.5 使用 ex 命令

事实上，在需要处理大块文本行的情况下，与上述的“复制—粘贴”与“剪切—粘贴”处理方式相比，利用 ex 命令还可以实现更精确、更方便的编辑处理。使用 ex 命令时，无需在编辑窗口中计数文本行的数量，然后再寻找插入点。用户只需提供一个准备复制或移动的文本行的范围，然后指定插入点的行号即可（当然，删除时无需指定插入点）。

6.5.1 显示行号

使用 ex 命令时，通常需要知道文本行的编号。要在编辑的文件中显示行号，可输入下列命令。

```
:set nu
```

按下 Enter 键后，新加的行号将会出现在编辑窗口的左边。注意，这些行号实际上并不存在于文件中，只是为方便用户的编辑处理而出现在编辑窗口中，以增加文本数据的可读性，如图 6-4 所示。

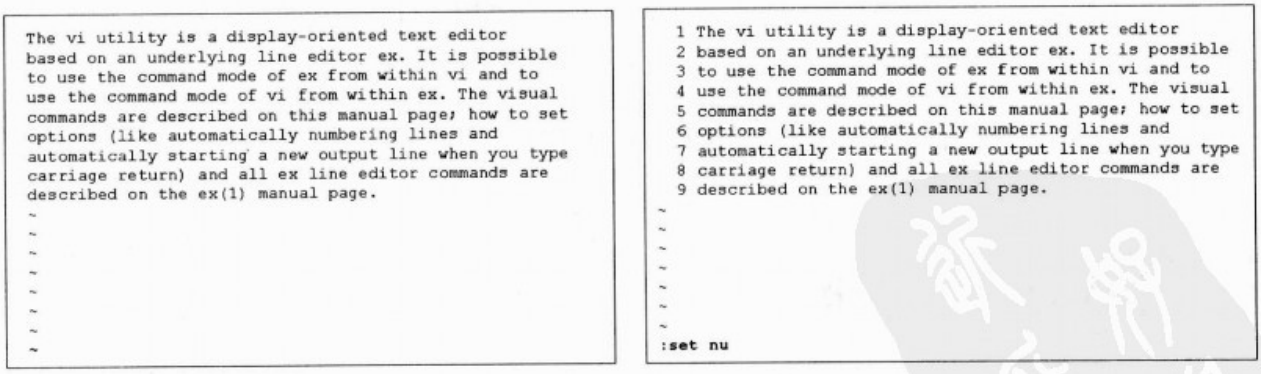


图 6-4 显示行号

如果想要关闭行号显示，可输入下列命令。

```
:set nonu
```

另外，要随时了解当前文本行的行号位置，可在命令模式中按下 Ctrl-G 键。vim 编辑器将会在编辑窗口的底部显示当前行的行号位置，以及当前文件的名字等信息。

6.5.2 多行复制

ex 复制命令的基本语法格式如下。

```
:line#1,line#2 co line#3
```

其中，前两个数字（即 line#1 和 line#2，中间以逗号分开）指定需要复制的文本行的范围，第 3 个数字（即 line#3）表示插入点的行号。例如，要把 myfile 文件的第 1～5 行复制到当前文件的第 12 行之后，可以使用下列命令。

```
:1,5 co 12
```

在指定文本行的范围时，可以使用下列缩写形式：

- 句点“.”　　表示当前行，意味着从当前行开始；
- 美元符号“$”　　表示文件的结尾，即文件的最后一行。

因此，要把当前行直至后续第 5 行复制到第 12 行之后，可以使用下列命令。

```
:.,5 co 12
```

为了把第 6 行直至文件最后一行复制到第 2 行之后，可以使用下列命令。

```
:6,$ co 2
```

6.5.3 移动文本行

ex 移动命令的基本语法格式类似于复制命令，如下所示。

```
:line#1,line#2 m line#3
```

在移动文本行时，可以采用与复制文本行相同的方式指定文本行的范围和插入点，包括使用缩写的句点“.”和美元符号“$”。两者的差别仅在于，“移动”命令将会把指定范围的文本行从一个位置整块地搬到另一个指定的位置。

例如，为了把第 1～5 行移至第 12 行之后，可以使用下列命令。

```
:1,5 m 12
```

6.5.4 删除文本行

删除文本行时，也可以采用相同的方式指定文本行的范围，包括使用缩写的句点“.”和美元符号“$”。要删除多个连续的文本行，可以使用下列命令形式实现。

```
:line#1,line#2 d
```

例如，要删除文件中的第 1～5 行，可输入下列命令。

```
:1,5 d
```

6.6 检索与替换

vim 提供了若干命令，使得用户能够以检索指定字符串的方式，直接跳转至期望的文件位置。另外，vim 还提供了强有力的全局检索与替换功能。

6.6.1 检索字符串

字符串由一个或多个连续的字符组成，它可以包含字母、数字、标点符号、特殊字符、空格、制表符或回车字符。字符串可以是一个语法意义上的单词，也可以是单词的一部分。要检索字符

串，可以使用如表 6-8 所示的命令。

表 6-8　　检索命令

命　令	命 令 描 述
:/str	检索给定的字符串。vim 将从当前光标位置开始检索，当找到指定的字符串时，光标将移至第一个出现的字符串位置。例如，要检索 meta，可以输入“/meta”命令，然后按 Enter 键
:?str	从当前位置开始，反向检索给定的字符串
n	从当前位置开始，继续检索下一个匹配的字符串
N	从当前位置开始，反向检索下一个匹配的字符串
/	同“n”命令，从光标当前所在字符位置开始，重复执行先前的检索，但在输入“/”字符之后还需按 Enter 键
?	同“N”命令，从光标当前所在字符位置开始，反向重复执行先前的检索，但在输入“?”字符之后还需按 Enter 键
:/pat/+n	将光标移至匹配的字符串“pat”所在行之后的第 n 行
:?pat?-n	将光标移至匹配的字符串“pat”所在行之前的第 n 行

如果想要检索字符串，可在“:/”后面附加准备检索的字符串，然后按 Enter 键，vim 将会从光标当前所在位置开始向下（文件结尾方向）检索。如果发现匹配的字符串，vim 将会把光标移至检索方向第一个出现的字符串位置。

例如，要检索字符串“meta”，可输入“:/meta”命令，然后按 Enter 键。如果接着输入“n”命令，可以继续检索下一个匹配的字符串。如果输入大写的“N”命令，则可以逆向检索前一个匹配的字符串。

如果想要在当前编辑的文件中向前（文件开始方向）逆向检索，可以使用“:?”代替“:/”命令。在任何情况下，“n”和“N”的检索方向仍然保持不变。但从逻辑上讲，则恰好与“:?”的检索方向相反。

通常，vim 的字符串检索是严格区分大小写字母的。例如，在检索“china”时，vim 不会发现“China”。如果想在检索期间忽略大小写的差异，可以输入“:set ic”命令。检索完成后，若要返回默认的匹配方式（区分大小写），可输入“:set noic”命令。

如果发现检索的字符串，vim 将会把光标移至（停留在）目标字符串的第一个字符位置。如果未发现检索的字符串，vim 将会在编辑窗口的底部（状态行）中输出“Pattern not found”信息，说明检索失败。

在检索过程中，某些特殊字符（/、&、!、.、^、*、$、\和?）具有特定的意义，如果检索字符串中本身也包含这样的字符，则必须在其前面增加转义符号——反斜线“\”，使 vim 能够将其按普通字符处理。例如，要检索字符串“anything?”，可输入“:/anything\?”命令。要检索一个本身就包含转义符号“\”的字符串，可在转义符号之前再加一个转义符号，即输入两个反斜线“\\”。

6.6.2　模式检索

利用 vim 提供的模式检索命令（参见表 6-9），还可以使字符串的检索更精确、更有效：

- 仅检索出现在行首位置的字符串；
- 仅检索出现在行尾位置的字符串；

- 仅检索出现在字首位置的字符串；
- 仅检索出现在字尾位置的字符串；
- 使用通配符检索字符串。

表 6-9　　模式检索命令

命　令	命 令 描 述
:/^*search*	要检索仅仅出现在行首位置的字符串，可以在字符串前面冠以一个上箭头“^”字符，使 vim 仅仅匹配行首位置，而忽略出现在其他位置的字符串。例如，若要从当前行开始检索以“From”为起始字符串的文本行，可以输入“/^From”命令。如果找到匹配的字符串，光标将会移至目的行的行首位置。如果想直接定位匹配的字符串，可接着输入“n”命令。如果找不到匹配的字符串，vim 将会在编辑窗口底部显示“Pattern not found”信息，而光标则继续停留在原来的位置
:/*search*$	要检索仅仅出现在行尾位置的字符串，可以在字符串后面附加一个“$”字符，使 vim 仅仅匹配行尾位置，而忽略出现在其他位置的字符串。例如，若要从当前行开始检索以“end”字符串结尾的文本行，可以输入“/end$”命令。同样，如果找到匹配的字符串，光标将会移至目的行的行首位置。如果想直接定位匹配的字符串，可接着输入“n”命令。如果找不到匹配的字符串，vim 将会在编辑窗口底部显示“Pattern not found”信息，而光标则继续停留在原来的位置
:/\<*search*\>	要匹配某个单字起始部分的字符串，可在检索字符串的前面冠以“\<”。若要匹配一个单字的结尾部分，可在检索字符串的后面附加“\>”。因此，若要准确地匹配一个完整的字而非字符串，可在检索字符串前后加上“\<”和“\>”。例如，如果想在整个文件中检索“search”这一英文单词，则可以输入“:/\<search\>”命令
使用“.”、“*”和“[...]”等通配符	要匹配任何一个字符，可以在检索字符串的相应位置中使用句点通配符“.”。例如，为检索“fun”或“gun”，可以输入“:/.un”命令。为了指定一个匹配范围，也可以使用范围通配符进行检索。例如，可以使用“:/[a-z]string”命令，使第一个字符仅限于小写字母。另外，也可以使用列举的方式匹配限定的几个字符。例如，“:/[dm]string”命令表示仅检索以“m”或“d”为起始字符的字符串“string”，即仅检索“dstring”或“mstring”。要检索以某个字符范围开头，中间含有某种模式的字符串，可以组合使用多个通配符进行检索。例如，为了检索以小写字母开头，中间含有“string”的字符串，可以使用“:/[a-z]*string*”命令

6.6.3　替换字符串

字符串替换是在前述字符串检索的基础上实现的。因此，在检索与替换的过程中可以使用任何通配符。

替换字符串命令的基本语法格式如下。

```
:[g]/search-string/s//replace-string/[g][c]
```

其中，第一个字符命令“g”表示全文检索，“s”表示替换，第二个字符命令“g”表示替换匹配的所有字符串，“c”表示在替换之前须经用户确认。因此，要把文件中的所有字符串“BankA”替换为“BankB”，可输入下列命令。

```
:g/BankA/s//BankB/g
```

执行上述命令后，vim 将会把从当前行开始，直至文件结尾范围内的所有“BankA”全部替换为“BankB”。若要在处理之前须先经用户确认，然后再替换，可以增加“c”命令，使 vim 在执行每个替换之前提请用户予以确认。例如，使用下列命令，可以使 vim 在用“BankB”替换“BankA”之前，先请用户予以确认。输入“y”表示同意替换，输入“n”表示不同意替换。

```
:g/BankA/s//BankB/gc
```

注意，利用 Ctrl-C 按键，可以在中途停止这种“确认—替换”的交互式字符串替换方式。

6.7 编辑多个文件

6.7.1 编辑多个文件

vim 允许用户同时编辑多个文件。例如，若想在编辑 file1 文件的同时，也编辑 file2 文件，可以使用下列命令打开两个文件。

```
vim file1 file2
```

此时，vim 首先显示第一个文件 file1，因此可以先行编辑 file1；编辑结束后输入“:w”命令，保存 file1 文件。若要编辑 file2，可以输入“:n”或“:n file2”命令；编辑结束后输入“:w”命令，保存 file2 文件。编辑结束后，可以输入“:q”命令，退出 vim 编辑器，或者使用“:q!”命令，中途强制退出 vim 编辑器。

在编辑多个文件期间，可以随时使用“:e *filename*”或“:n *filename*”命令，直接转到指定的文件，也可以使用“:n”命令转到下一个文件，或使用“:n #”命令交替编辑最近处理过的两个文件。如果对文件做过任何编辑加工，在跳转之前，vim 要求用户使用“:w”命令保存当前文件，除非设置了 vim 的 autowrite 选项——vim 会在跳转之前自动保存修改后的文件。如果想放弃当前的修改，可以使用“:e! *filename*”命令，强行转至指定的文件，或者使用“:q!”命令，强制退出 vim 编辑器。

6.7.2 合并文件与合并文本行

在编辑过程中，使用 vim 的“r”命令能够很方便地把指定的文件读入（插入）当前文件光标所在的位置，该命令的一般语法格式如下。

```
:line# r filename
```

如果未指定行号，vim 将在光标当前所在行的后面插入文件。例如，如果想在文件的第 10 行之后插入文件 chap2，可以输入下列命令。

```
:10 r chap2
```

另外，也可以先把光标移至第 10 行的位置，然后输入下列命令（这有助于确认插入位置的正确与否）。

```
:r chap2
```

若要把相邻的两个文本行合并为一行，可以把光标移至第一行，然后输入 J 命令。

6.8 定制 vim 编辑器的运行环境

6.8.1 临时设定 vim 的运行环境

通常，vim 编辑器采用一系列默认的选项定义作为自己的运行环境。这些预置的默认选项基本上能够满足大多数用户的日常编辑处理工作，但有时出于某些特定的需要，或者为了提高编辑效率，需要改变部分选项的默认值。若要查询 vim 编辑器各种选项的具体设置，可以使用下列命令。

```
:set all
```

“set all”将会输出 vim 编辑器当前支持的所有选项及其默认设置，如图 6-5 所示。

图 6-5　vim 编辑器当前使用的选项及其默认设置

如果想要改变某个选项的设置，可以使用下列 set 命令。

```
:set option
```

其中，option 是 vim 编辑器支持的选项名称。要取消某个编辑器选项的设置，可在选项前增加 no 字样，如下所示。

```
:set nooption
```

表 6-10 给出了 vim 编辑器支持的部分重要选项及其说明。

表 6-10　　vim 编辑器支持的部分选项

选　项	缩　写	默 认 值	简 单 说 明
all	无	无	在编辑窗口中列出编辑器支持的所有选项
magic	无	magic	在检索字符串时，下列字符默认情况下具有特殊的意义： ❑ “.”表示匹配任何一个字符； ❑ “[...]”表示匹配指定字符集合或字符范围中的任何一个字符； ❑ “*”表示匹配任何字符或字符串。 在设置了 nomagic 选项之后，上述字符将失去其特殊的意义，使用 magic 选项可以恢复其原有的特殊意义。注意，“^”（表示行首位置）和“$”（表示行尾位置）总是具有特殊的意义，不受此选项设置的影响
autoindent	ai	noai	这个选项应与 shiftwidth 选项一起发挥作用，使新输入的文本行与上一行起始位置自动对齐。当 vim 处于输入模式时，按下制表键 Tab 将会使光标移至下一个制表位置，按下 Enter 键将会使光标移至下一行，并与上一行的第一个字符位置对齐，按下 Ctrl-T 键将会使当前行移至下一个制表位置，而 Crtl-D 键将会使当前行反向移至前一个制表位置。这一选项比较适用于程序员
autowrite	aw	noaw	在编辑多个文件的情况下，如果设置了 autowrite 选项，当使用“:e”或“:n”命令在文件之间切换时，vim 将会在切换之前自动保存对当前文件所做的编辑处理。否则，vim 将会提示用户把缓冲区中的内容写到磁盘文件中（如果当前编辑的文件中发生了任何变动）
ignorecase	ic	noic	在 vim 编辑器中，字符或字符串的匹配检索通常是严格区分大小写字母的。如果设置了这一选项，可以使 vim 在匹配检索过程中忽略大小写字母的差别
list	无	nolist	通常，vim 将会按照正常的方式显示文本文件，如按照要求把制表符扩展为适当数量的空白字符，不显示回车字符等。如果设置了这个选项，vim 将会把制表符显示为“^I”，在每一个文本行的后部附加一个美元符号“$”等

续表

选　项	缩　写	默 认 值	简 单 说 明
laststatus	ls	ls=1	这个选项用于确定是否在编辑窗口中显示状态行。状态行中包括当前文件的名字、加号“+”标志（如果编辑的文件发生变动后尚未被写到磁盘上）和光标的位置等。这个选项的有效值是 0、1 和 2。其中，0 表示关闭状态行的显示，1 表示仅当存在两个以上的文件编辑窗口时才显示状态行，2 表示总是显示状态行
number	nu	nonu	在编辑文本文件时，vim 编辑器通常不会显示文本行的行号。为了在每个文本行前面增加一个行号，可以设置这个选项。注意，vim 编辑器显示的行号并非文件中的一部分，也不会被存储到文件中，只是为了编辑的方便，提供一种临时的标记而已。默认值为 nonu
readonly	无	noreadonly	对正在编辑的文件启用写保护机制，当编辑文件时，vim 将会向用户发出警告提示，以避免修改或损害重要的文件
report	无	report=2	这个选项使 vim 能够确定在删除或复制多少文本行时需要在状态行中显示影响的文本行信息，如“8 lines deleted”。当删除或复制较少的文本行（少于等于指定值）时，vim 不会显示任何信息。这个选项的默认值为 2
scroll	scr	scr=*nn*	这个选项用于控制按下 Crtl-D 或 Ctrl-U 键时，前滚或后滚多少文本行，其默认值为编辑器窗口高度（窗口行数）的一半。要修改 Crtl-D 或 Ctrl-U 键滚动的行数，通常有两种实现方式：一是在按下 Crtl-D 或 Ctrl-U 键之前首先输入一个数字，二是使用这个选项事先设定一个数值，如“:set scroll=10”
shell	sh	sh=*path*	在使用 vim 编辑器时，用户可以临时或长久地调用一个 Shell，运行单个 Linux 命令，或者交互地访问 Shell,运行多个 Linux 命令。这个选项用于确定 vim 调用哪一个 Shell。默认情况下，vim 把这个选项设置为用户的注册 Shell。为了选用不同的 Shell，可以采用“:set sh=*path*”形式定义 Shell，其中 path 为 Shell 的绝对路径名
shiftwidth	sw	sw=8	这个选项用于设定制表符的跳转位置，以便在输入模式下，当按下制表键 Tab、Crtl-T 或 Ctrl-D 键时，光标能够自动地跳转到下一个或前一个制表符位置。这个选项的默认值为 8
showmatch	sm	nosm	输入右圆括号、花括号或方括号时提示相应的左圆括号、花括号或方括号。此选项对输入数学表达式或编写程序用到圆括号或花括号时比较有用
showmode	smd	smd	根据 vim 编辑器当前所处的工作模式，在编辑窗口左下角显示输入模式“--INPUT--”及替换模式“-- REPLACE --”等信息
tabstop=8	ts	ts=8	设置制表键 Tab 的右移距离。默认值为 8 个空格
wrap	无	wrap	wrap 选项用于控制 vim 编辑器怎样显示较长的文本行。为了使 vim 能够把较长的文本行延续到下一行，可以利用这个选项实现自动折行。如果设置了 nowrap，vim 将会截断较长文本行后部的超长部分（只是不显示，并非真正截掉超常部分）。这个选项的默认值为 wrap
wrapmargin	wm	wm=0	指定编辑窗口的右边距。假定终端窗口为 80 列，如果使用“:set wm=8”命令设置右边距，则文本行的逻辑长度不能超出第 72 列的位置。当输入的文本行到达指定的边界，vim 编辑器将会在最接近边界的字（以空格字符为分界符）右边插入一个换行字符。这个选项可以帮助用户摆脱在输入每一行之后再按 Enter 键的麻烦。数值 0（默认值）表示关闭这一功能，其他非 0 数值表示启用并设定编辑窗口的右边距
wrapscan	ws	wrapscan	这个选项影响字符串的检索方式。如果设置了 ws 选项，当检索到达文件的结尾仍未发现匹配的字符串时，vim 将会从头开始继续检索；否则，在到达文件的结尾时，即使仍未发现匹配的字符串，vim 也会停止检索
compatible	cp	cp	除非.vimrc 文件存在，默认情况下 vim 将会尝试采用与 vi 兼容的工作方式

另外，为了加快数据输入的速度，还可以使用下列命令形式，定义用户自己常用的缩写形式。

```
ab abbr string
```

其中，abbr 为 string 的缩写形式，string 为实际上应出现在编辑器中的数据。例如，如果使用

下列命令：

```
ab iscas 中国科学院软件研究所
```

则每当输入一个 iscas 单词时，vim 编辑器会自动代以“中国科学院软件研究所”字符串。

6.8.2 永久性地定制 vim 的运行环境

上面介绍的方法只能临时地设置 vim 编辑器的当前运行环境，一旦退出 vim，这种临时设置也随之作废。若要永久性地设置各种选项，以免每次进入 vim 时都要重新设置，可以设置 VIMINIT 变量。如果使用的是 bash，可以在.bash_profile 中增加下列形式的变量设置。

```
export VIMINIT='set para1 para2 …'
```

例如，为了总是显示状态行，在匹配检索过程中忽略大小写字母的差别，可以把 VIMINIT 变量设置如下。

```
export VIMINIT='set laststatus=2 ignorecase'
```

另外，也可以把常用的 vim 选项及其定义（即 set 命令可以设置的选项和定义）加到系统范围的初始化文件/etc/vim/vimrc 文件，或用户主目录下的.vimrc（或.exrc）文件中。如此，则每当调用 vim 编辑器时，vim 都会自动读取此文件，使定制的选项成为 vim 的永久性运行环境。下面是一个.vimrc 文件示例（注意，不要在命令前加冒号）。

```
set showmode
ab abc 中国农业银行
ab boc 中国银行
ab cbc 中国建设银行
ab icbc 中国工商银行
……
```

实际上，还可以在每个目录中创建一个自己的.vimrc 文件，以适应不同的需要。例如，可以在 C++源程序所在的目录中创建一个适应于程序开发的.vimrc 文件，在编写文档的目录中创建一个适应于文档编写的.vimrc 文件。但应注意，仅当用户主目录的.exrc 文件包含“set exrc”命令时，其他目录中的.vimrc 文件才能发挥作用。

6.9 其他特殊说明

6.9.1 删除或替换特殊字符

在编辑文件时，有时会遇到文件中含有不可打印的特殊字符。例如，由于 Windows 与 Linux 系统使用的行终止符不同，对于 Linux 系统而言，取自 DOS 或 Windows 系统中的文本文件会包含多余的回车字符，如图 6-6 所示。

为了替换或删除这种特殊字符，可以在 vim 的删除或替换命令中，利用“Ctrl-V”键输入特殊字符的 ASCII 编码，以删除或替换特殊字符。例如，回车字符的 ASCII 编码为 13，对应于 Ctrl-M 键，可以采用下列替换命令，删除多余的“^M”字符。

```
1,$ s/^M//
```

注意，上述命令中的“^M”字符不能直接输入，必须先按 Ctrl-V 键，接着再按 Ctrl-M 键，才能输入“^M”字符，如图 6-7 所示。

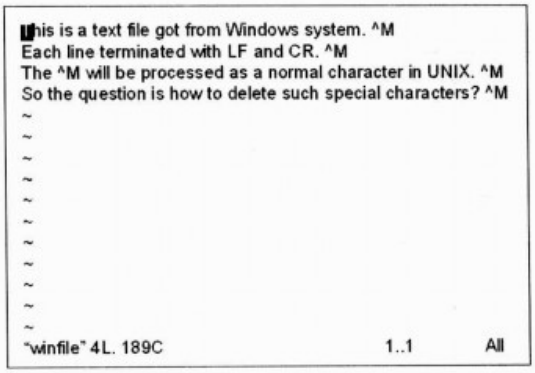


图 6-6 DOS/Windows 文本文件

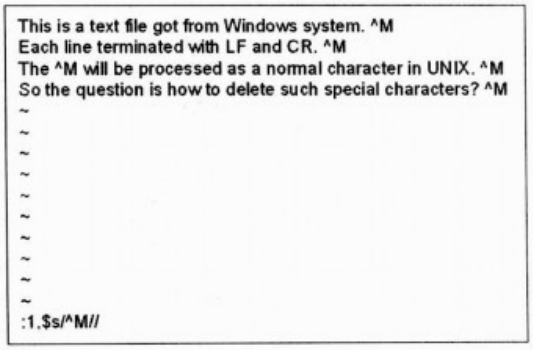


图 6-7 替换回车字符

6.9.2 在编辑期间运行 Linux 命令

有时，可能需要在编辑过程中运行 Linux 命令。例如，如果需要把某个命令的输出作为文件的一部分，则可以利用表 6-11 列出的命令，临时或长时间地运行其他 Linux 命令，或者把 Linux 命令的运行结果引入正在编辑的文件中。

表 6–11 辅助编辑命令

命　令	命 令 描 述
:sh	在编辑文件期间，如果想长时间地运行 Shell 命令，然后再返回 vim 编辑器，可输入“:sh”命令。此时，用户将处于正常的 Shell 命令行工作方式，当使用 Ctrl-D 键或 exit 命令退出 Shell 时，即可返回原来的 vim 编辑器，继续进行其他编辑处理
:!*command*	在编辑文件期间，如果想临时地运行某一个 Shell 命令，然后继续进行编辑处理，可输入“:!command”命令。此时，vim 将会在不退出编辑器的情况下，执行指定的 Shell 命令。在 Shell 命令运行结束之后，按下 Enter 键，即可恢复原来的编辑处理状态
!!*command*	在编辑文件期间，如果想把某个 Shell 命令的运行结果直接加到当前编辑的文件中，使命令的输出替代当前行，然后继续进行编辑处理，可输入“!!command”命令

6.10 vim 编辑器命令总结

表 6-12 分类列出了 vim 编辑器的基本命令及其简要说明。

表 6–12 编辑器命令一览表

命　令	简 单 描 述
启动 vim 编辑器：	
vim *filename*	打开原有的文件或创建一个新文件
vim	打开一个新文件，在编辑过程中或结束编辑时再指定文件名
vim -r *filename*	恢复因意外停机或终端连接中断而未及时保存最终编辑结果的文件
view *filename*	以只读方式打开文件。除了不能把编辑处理的最终结果写入文件保存之外，view 的所有编辑功能均与 vim 无异
光标定位命令：	

续表

命　令	简 单 描 述
←↑↓→	将光标左移、上移、下移或右移一个字符（行）位置
h j k l	同上
-	光标上移一行
Enter 键（或加号“+”）	光标下移一行
退格键	将光标左移一个字符位置
空格键	将光标右移一个字符位置（命令模式）
Ctrl-F	往下（文件结尾方向）滚动一屏
Ctrl-B	往上（文件开始方向）滚动一屏
Ctrl-D	往下滚动半屏
Ctrl-U	往上滚动半屏
Ctrl-E	编辑窗口中的文件内容整体上移一行
Ctrl-Y	编辑窗口中的文件内容整体下移一行
w	将光标右移一个字。光标停留在下一个字的字首位置
W	将光标右移一个字。光标停留在下一个字的字首位置（即使两个字之间存在标点符号）
b	将光标左移一个字。光标停留在下一个字的字首位置
B	将光标左移一个字。光标停留在下一个字的字首位置（即使两个字之间存在标点符号）
e	把光标移至当前所在字（或下一个字）的最后一个字符位置
E	同上，只是以空格字符作为字的分隔符
^	把光标移至当前行的起始位置，即当前行的第一个非空白字符位置
0（零）	同上
$	把光标移至当前行的行尾，即当前行的最后一个字符位置
H	把光标移至编辑窗口顶部第一行的行首位置
M	把光标移至编辑窗口中间一行的行首位置
L	把光标移至编辑窗口底部最后一行的行首位置
插入文本数据：	
a	在光标当前所在字符位置的后面输入文本数据
A	在光标当前所在行的行尾（即最后一个字符位置）后面输入文本数据
i	在光标当前所在字符位置的前面输入文本数据
I	在光标当前所在行的行首（即在第一个非空白的起始字符）前面输入文本数据
o	在光标当前所在行下面的行首位置输入文本数据
O	在光标当前所在行上面的行首位置输入文本数据
修改文本：	
C	替换当前文本行光标所在字符位置之后的所有数据，以 Esc 键结束
cw	替换光标当前所在字符位置及之后的整个字或部分字，以 Esc 键结束
[*n*]cc	替换当前行，或从当前行开始的 n 行文本，以 Esc 键结束

续表

命　令	简 单 描 述
[*n*]s	替换光标当前所在位置的单个字符，或从光标当前位置开始的 n 个字符，以 Esc 键结束
S	替换当前行，以 Esc 键结束
r	替换光标当前所在位置的单个字符
r Enter	断行。也可使用“a”或“i”命令加 Enter 及 Esc 键实现
R	从光标当前所在的字符位置开始，替换随后的所有字符，直至按下 Esc 键
xp	交换字符位置。交换光标当前所在位置开始的两个字符位置
~	转换光标当前所在位置字符的大小写
u	撤销最近一次执行的编辑命令，或者依次撤销先前执行的编辑命令
:u	同上（ex 编辑命令）
U	撤销施与当前文本行的编辑处理
删除文本：	
[*n*]x	删除光标当前所在位置的字符，或者删除从光标当前位置开始的 n 个字符
[*n*]X	删除光标当前所在位置的前一个字符，或者删除光标当前所在位置之前的 n 个字符
dw	删除光标当前所在位置的一个整字或部分字符。如果光标在字首，则删除整字；如果光标在字的中间任何位置，则删除光标位置及之后的字符
[*n*]dd	删除光标当前所在的文本行，或者删除从当前行开始的 n 个文本行
D	删除当前文本行从光标位置开始之后的所有字符
dG	删除从当前行开始直至文件最后一行的所有文本行
d[*n*]G	删除从文件的第 n 行开始直至当前行的所有文本行
:*line#1*,*line#2* d	删除指定的行号 line#1～line#2 之间的所有文本行
复制与移动文本：	
[*n*]yy	复制光标当前所在的文本行，或者从当前行开始的 n 个文本行
[*n*]Y	同上
p（小写）	把复制或删除（“dd”命令）的文本行粘贴到光标所在行的下面
P（大写）	把复制或删除（“dd”命令）的文本行粘贴到光标所在行的上面
:*line#1*,*line#2* co *line#3*	把第 line#1～line#2 行复制到第 line#3 行之后
:*line#1*,*line#2* m *line#3*	把第 line#1～line#2 行移至第 line#3 行之后
设置行号显示：	
:set nu	在编辑期间增加临时行号
:set nonu	撤销行号显示（默认情况）
Ctrl-G	显示当前文件的名字和当前文本行的行号
设置大小写字母检索准则：	
:set ic	检索字符串时忽略字母的大小写
:set noic	检索字符串时严格区分字母的大小写（默认情况）
定位文本行：	
G	将光标移至文件的最后一行

续表

命　令	简 单 描 述
[*n*]G	将光标移至文件的第 n 行
检索与替换：	
:/*string*	向前（文件结尾方向）检索指定的字符串
:?*string*	向后（文件开头方向）检索指定的字符串
n	按检索方向找出下一个匹配的字符串
N	逆检索方向找出前一个匹配的字符串
:[g]/*search*/s//*replace*/[g][c]	检索并替换字符串
清除屏幕：	
Ctrl-L	清除因其他进程的输出信息而干扰的编辑窗口
合并文件与合并行：	
:r *filename*	在光标所在行之后插入指定文件的内容
: *line#1* r *filename*	在第 line#1 行之后插入指定文件的内容
J	把相邻的两个文本行合并为一行（把下一行合并到光标当前所在行的后面）
保存编辑结果与退出 vim 编辑器：	
:w	保存编辑处理后的结果（把内存缓冲区中的数据写到文件中）
:w!	强制保存编辑处理后的结果
:wq	保存编辑处理后的结果，然后退出 vim 编辑器
:wq!	强制保存编辑处理后的结果，然后退出 vim 编辑器
ZZ	保存编辑处理后的结果，然后退出 vim 编辑器
:q	在未做任何编辑处理时，可以使用此命令退出 vim 编辑器
:q!	强制退出 vim 编辑器，放弃编辑处理后的结果
:w *filename*	把编辑处理后的结果写到指定的文件中保存
:w! *filename*	把编辑处理后的结果强制写到指定的文件中保存，即使文件已经存在
:wq! *filename*	把编辑处理后的结果强制写到指定的文件中保存，即使文件已经存在，然后退出 vim 编辑器
其他：	
:f 或 Ctrl-G	显示文件的名字、编辑状态、文件总的行数、光标当前所在的行号和列号，以及当前行之前的行数占整个文件总行数的百分比
Ctrl-V	用于输入其他控制字符