第3章 Linux 的 vim 编辑器

本章学习目标

- 掌握 vim 的进入和退出。
- 掌握 vim 的 3 种工作方式及 3 种方式的转换。
- 掌握命令模式及末行模式下的文本编辑命令。

3.1 vim 简介

Linux 系统提供了一个完整的编辑器家族，如 ed、ex、vi 和 emacs 等。按功能可以将编辑器分为两大类：行编辑器（ed、ex）和全屏幕编辑器（vi、emacs）。行编辑器每次只能对一行进行编辑操作，使用起来很不方便。而全屏幕编辑器可以对整个屏幕进行编辑，用户编辑的文件直接显示在屏幕上，可以立即看到修改的结果，克服了行编辑器不直观的操作方式，便于用户学习和使用。vi 就属于全屏幕编辑器，也是 Linux 系统的第一个全屏幕交互式的编辑程序。该编辑器最初由加州大学伯克利分校为 BSD UNIX 开发，在大多数的 UNIX 系统中都默认提供该编辑工具。受到图形显示及键盘功能（当时键盘无功能键）的限制，最初的 vi 中没有提供鼠标的使用，只能通过键盘上的字母、数字、标点符号和 Esc 键对文件进行编辑。因此，在今天看来 vi 似乎有些过时，但却是 Linux 系统程序员和管理员最喜欢的编辑器之一。

Linux 系统提供了多个 vi 的版本，现在版本的 Linux 系统中运行的 vi 实际是 vim（vi improved），在 vi 的基础上增加了很多新的特性和功能，成为 Linux/UNIX 环境下最重要的开源编辑器之一。在 Linux 系统上运行的 vi 实际上就是 vim，vim 的基本使用方式和命令与原来的 vi 一致，因此，本章介绍的 vim 可以与 vi 互换使用。

需要提醒的一点是，vim 编辑器不是一个格式化文本的程序，不能调整版面，也不具有复杂的文字处理系统所具有的格式化输出功能。但 vim 是一个灵活的文本编辑器，可以用来编写代码（如 C、HTML、Java 等）、编写 Shell 脚本程序、记录简单的信息等。

3.2 vim 工作模式

vim 编辑器没有菜单，只有命令，而且命令繁多，vim 编辑器中所有的文本编辑工作

都是通过命令操作完成的，因此，使用 vim 必须记住这些命令。而 vim 中的命令大多数是键盘的字母和数字，如何区别这些字母和数字到底是文本编辑的内容还是 vim 的命令？vim 提供 3 种基本的工作方式来解决，分别是命令模式（Command Mode）、插入模式（Insert Mode）和末行模式（Last Line Mode），在这 3 种工作模式之间进行切换就可实现文本编辑的功能。

1. 命令模式

在 Shell 环境中启动 vim 时，初始就是进入命令模式。在该模式下，键盘上输入的任何字符都作为编辑命令来解释，如果输入命令合法，则命令直接执行；否则，响铃提示非法命令。命令包括编辑保存，移动光标，页面滚动，字符、字或行的删除、移动、复制等。需要注意的是命令方式下的所有命令并不在屏幕上显示出来，也不需要按回车键确认命令执行。不管用户处于何种工作模式，只要按 Esc 键，即可使 vim 进入命令模式。

2. 插入模式

只有在插入模式下才可以进行文本输入。在命令模式下输入插入命令 i、附加命令 a、打开命令 o、修改命令 c、取代命令 r 或替换命令 s 都可以切换到插入模式。在该模式下，输入的任何字符都被 vim 当作文件内容保存起来，并将其显示在屏幕上。在文本输入的过程中，若想回到命令模式下，按 Esc 键即可。

3. 末行模式

在命令模式下，按“：”键即可进入末行模式。此时，vim 会将光标停留在显示窗口最后一行显示一个“：”，“：”作为末行模式的命令提示符，等待用户输入命令。多数管理命令都是在此模式下执行的，如保存文件或退出 vim、查找字符串、显示行号等。末行命令执行时需要按回车键确认，执行完毕自动回到命令模式。有人把 vim 的工作模式简化成两种，即把末行模式也作为命令模式。

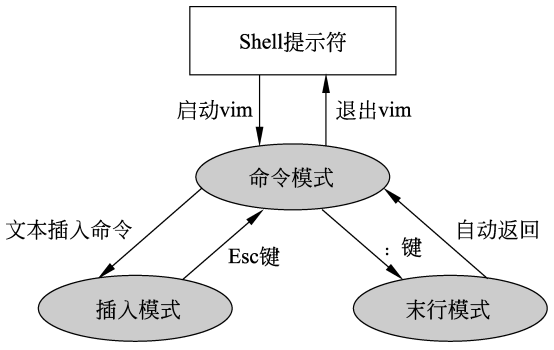

图 3.1　vim 的 3 种工作模式

vim 编辑器的 3 种工作模式及其之间的转换如图 3.1 所示。

需要提醒的是在输入 vim 命令时，请注意区分大小写。因为 vim 区分大小写，会把字母相同但大小写不同的两个命令认为是不同的命令。

3.3　vim 的进入与退出

vim 是在 Linux 系统终端运行的程序，它的所有操作都必须通过输入相应的命令完成。本节介绍如何启动 vim、输入文本、如何保存编辑的文件以及如何退出 vim。

3.3.1　进入 vim

在终端 Shell 提示符后输入 vi 命令或 vim 命令，系统将启动 vim 编辑器。

```
$vim
```

图 3.2 所示为 vim 的启动的窗口，窗口上说明了 vim 的维护人、版权等基本信息。

图 3.2 vim 的启动界面

进入 vim 编辑器的另外一种方式是输入 vim 和待编辑或新建的文件名。命令格式如下：

```
$vim filename
```

图 3.3 为输入命令 vim file 后的 vim 窗口。

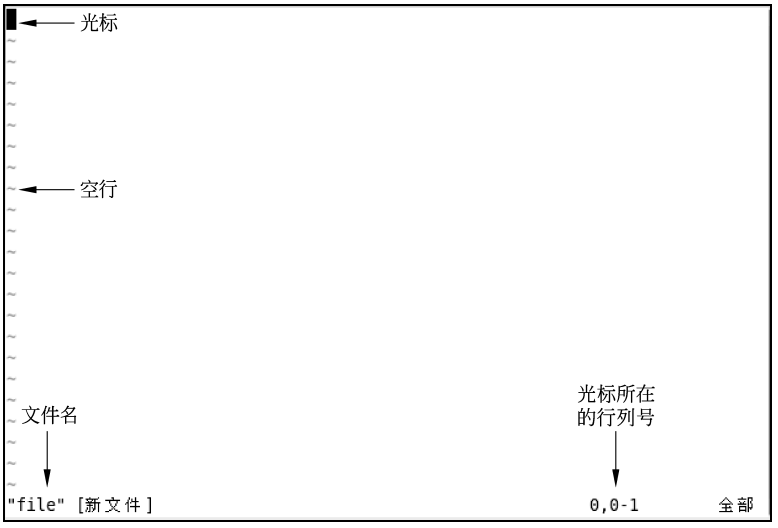

图 3.3 vim 编辑文件 file 窗口

进入 vim 后即进入命令模式，要输入文件内容则应进入插入模式，此时需要命令模式

与插入模式之间的转换。注意：初始的显示行数与用户所用的终端有关，一般的终端可显示 25 行。在窗口系统中，显示行数与运行 vim 的那个窗口有关，也可以对显示行数进行设置。

用 vim 建立一个新文件时，在进入 vim 的命令中可以不给出文件名，在编辑完文件后保存数据时由用户再指定文件名即可。

使用 vim 编辑文件时，光标默认情况下停留在第一行，通过使用相关选项可以将光标定位在指定的行，格式如下：

```
$vim +n filename
```

其中，n 表示行号，若 n 被去除，该命令默认进入文件 filename 后，光标停留在文件最末行。

如果想指定光标停留在某个指定的字符串，使用如下命令：

```
$vim +/pattern filename
```

pattern 为用户指定的字符串，如：

```
$vim +/void file
```

按回车键确认后，光标会停留在文件 file 第一次出现 void 字符串的位置。

当使用 vim 进行文件编辑时，如果系统突然崩溃了，vim 会将操作的交换文件临时存储起来，再次启动系统后，可以根据这些交换文件对文件进行恢复。使用的命令如下：

```
$vim –r filename
```

3.3.2　退出 vim

当编辑完文本准备从 vim 返回到 Shell 时，可以使用以下几种方式。

1. 命令模式下

在命令模式下，连续按两下大写字母 Z，若当前文件被修改过，则 vim 保存该文件后退出，返回到 Shell；若当前编辑的文件未被修改，则 vim 直接退出，返回到 Shell。

2. 末行模式下

在末行模式下，退出 vim 有以下 4 种情况。

（1）若当前文件被修改过，保存后退出，使用的命令如下：

```
: w  <newfile>
: q
```

其中，w 表示保存文件；newfile 表示将当前编辑内容保存到指定的文件 newfile 当中，如果去掉该参数，默认将编辑内容保存到原文件中。若保存文件时提示文件不可写，可以使用如下命令进行强制写保存（要慎重使用）：

```
: w! filename
```

其中，q 表示从当前编辑环境退出，返回到 Shell 中。

（2）如果当前编辑的内容保存到原文件退出，也可以使用如下命令：

```
：wq
```

（3）若当前文件被修改过，不保存退出到 Shell，使用的命令如下：

```
：q!
```

（4）若当前文件没有被修改过，从 vim 退出返回到 Shell，使用的命令如下：

```
：q
```

3.4 vim 的编辑命令

3.4.1 移动光标

vim 中的光标移动，既可以在命令模式下进行，也可以在插入模式下进行。在插入模式下，可直接使用键盘上的 4 个方向键移动光标；在命令模式下，有多种移动光标的方法。下面主要介绍在命令模式下如何移动光标。

1. 按字符移动光标

在命令模式下，可以使用键盘的 4 个方向键移动光标。

vim 还为所有终端定义了另外一组按字符移动光标的键，分别是 H、J、K、L 键。4 个键分别表示光标方向左移、下移、上移和右移，如图 3.4 所示。

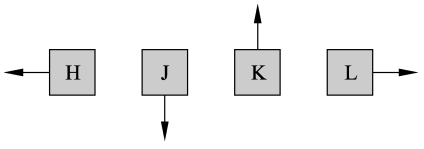

图 3.4 左、下、上、右移动光标

注：H、J、K、L 为小写字符。

键盘的空格键也可以将光标右移一个字符，与之对应的退格键（Backspace）是光标左移一个字符。

2. 按字移动光标

vim 中“字”有两种含义：一种是大写字，一种是小写字。当处于大写字方式时，“字”包括两个空格之间的所有内容。例如，对于如下一行文本：

```
Hello,  World!
```

其中，逗号（,）和字母 W 之间有一个空格字符分隔。对于大写字而言，该文本只有两个字，一个是“Hello,”，另一个是“World!”。

当处于小写字方式时，英文单词、标点符号和非字母字符都被当成是一个字。因此，上面的一行文本就包括 Hello、逗号（,）、World 和感叹号（!）4 个字。

在 vim 中，使用大写命令一般是将字作为大写字来处理，使用小写命令则是将字作为小写字来处理。空行既可以看作是小写字又可以看作大写字来处理。在命令模式下按字移动光标可以使用以下命令。

w/W 键：将光标向右移到下一个字。w 表示右移到下一个小写字的字首，W 表示右移

到下一个大写字的字首。在文本中如果有标点符号，则按照大写字和小写字移动是有区别的；否则两者之间没有区别。命令 5w 将把光标移到后面第 5 个字的字首处。

b/B 键：将光标向左移动到上一个字。b 表示左移到上一个小写字的字首，B 表示左移到上一个大写字的字首。举例如图 3.5 所示。

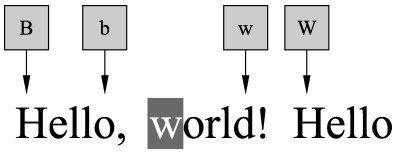

图 3.5　按字移动光标

类似地，e 键将光标移动到下一个字的末尾字符；E 键将光标移到下一个空白分隔字的末尾。

3. 按句子和段移动光标

在 vim 中，句子被定义为以句号（.）、问号（?）和感叹号（!）结尾，且其后面至少有一个空格或一个换行符的字符序列；段落被定义以一个空白行开始和结束的文本片段。

(/) 键：将光标按句移动。其中，“(”表示将光标移动到上一句句首；“)”表示将光标移到下一句的句首，如果光标在本句中，则移动到本句句首。

{/}键：将光标按段移动。其中，“{”表示将光标移动到上一段落段首；“}”表示移动到下一段落段首，若当前光标处于段中，则移动到本段段首。

举例如图 3.6 所示。

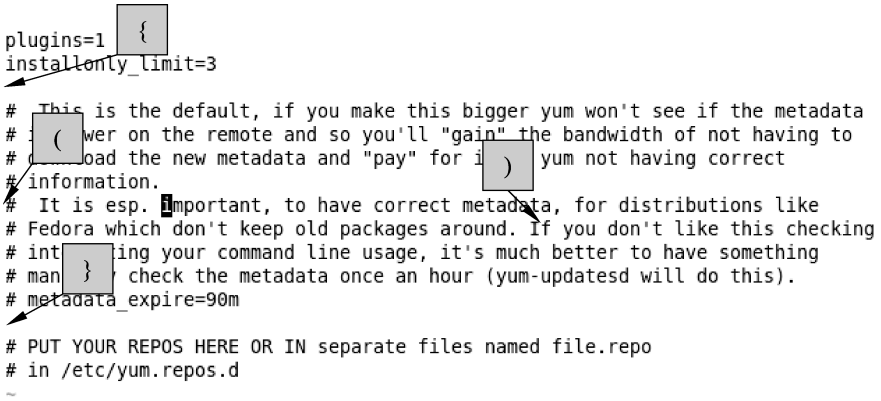

图 3.6　按句、段移动光标

4. 按行移动光标

数字 0 可将光标移动到当前行行首；$键，可将光标移到当前行行尾。G 键将光标移到文件最末行行首，若将光标移动到指定行，可使用[行号]G。

例如，5G 表示将光标移到文件的第 5 行，举例如图 3.7 所示。

5. 在屏幕内移动光标

H/M/L 中的 H（Home）键将光标定位到屏幕顶部一行的最左端；M（Middle）键将光标定位到屏幕的中间一行；L（Lower）键将光标定位到屏幕底部的一行。注意：这里的移动是指屏幕内，文件本身不发生滚动。

提醒：vim 中所有的命令都区分大小写，在使用时需要特别注意。

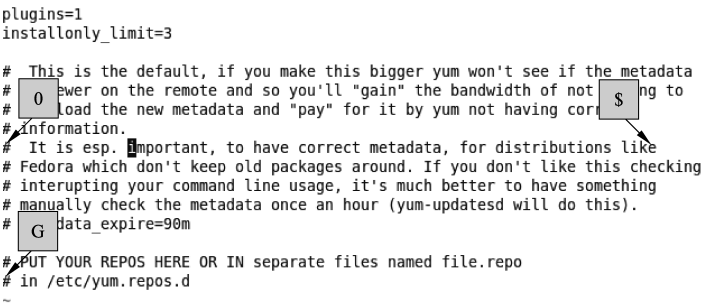

图 3.7　按行移动光标

3.4.2　文本插入

在命令模式下，用户输入的任何字符都被 vim 当作命令加以解释执行。如果想进行文本编辑，则需要从命令模式切换到文本插入模式。在命令模式下，可以通过插入命令、附加命令、打开命令进行模式切换。

1. 插入命令

```
i/I
```

其中，i（Insert）命令表示从光标所在位置前插入文本；I 命令表示将光标移动到当前行首开始插入文本。

2. 附加命令

```
a/A
```

其中，a（Append）命令表示从光标所在位置之后开始追加文本；A 命令表示首先将光标移到所在行的行尾，从行尾开始插入文本。

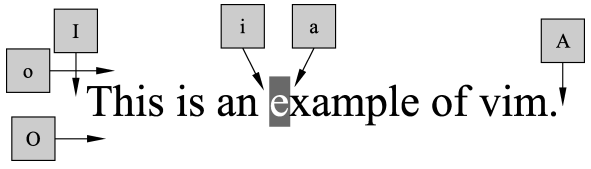

图 3.8　命令 i/I/a/A/o/O

3. 打开命令

```
o/O
```

其中，o（Open）命令表示在当前行的下面打开一行；O 命令表示在当前行的上面打开一行。

举例如图 3.8 所示。

3.4.3　文本删除

vim 提供了命令模式和末行模式两种模式下的删除命令。

1. 命令模式下文本删除

在命令模式中，vim 提供了很多删除命令进行文本的删除。文本删除通常由删除命令 x 或 d 与要删除的文本对象组成，而且在 x 和 d 前面可以加上数字来表示要删除的文本对象的个数。表 3.1 列出了一些常用删除文本的命令。

表 3.1　删除文本命令

命　令	功　能
x	删除当前字符
X	删除光标左边字符

续表

命　令	功　能
dw	删除当前字
d0（数字）	删除从当前字符的前一个字符到行首
d$	删除从当前字符开始到行尾
dd	删除当前行
ndd	删除当前行开始连续的 n 行
d)	从当前字符开始删除到句子尾
d(	从当前字符开始删除到句子首
d}	从当前字符开始删除到段落尾
d{	从当前字符开始删除到段落首
dH	删除当前行到屏幕首行的内容
dM	删除当前行到屏幕中间行的内容
dL	删除当前行到屏幕末行的内容
d/text	删除当前行到 text 单词出现的位置
dG	删除到文件末尾

2. 末行模式下文本删除

在 vim 末行模式下，可实现对文本指定行的删除，使用如下命令：

```
：行 x，行 y  d
```

该命令表示删除 x 至 y 行的文本内容。例如：

```
：1, 5 d
```

按回车键确认后，1～5 行文本内容被删除，其后的文本上移。

在 vim 编辑环境中，默认情况下是不显示行号的，如果想使用行号进行文本操作，需要使用相应的命令对行号进行显示，命令如下：

```
：set number
```

该命令被执行后，编辑文件的行号会在每一行的左边显示。需要说明的是，这里的行号只是便于用户查看使用，并不是文件内容的一部分。

3.4.4　文本复制与粘贴

vim 提供了与 Windows 的剪贴板类似的功能。剪贴板是内存的一块缓冲区，复制命令就是把指定的内容复制到剪贴板上，然后粘贴命令将剪贴板的内容粘贴到光标处。文本的复制即可在命令模式下完成又可在末行模式下实现。

1. 命令模式下的文本复制

在命令模式下，vim 中的文本复制命令如下。

（1）yw：将光标所在位置到字尾的字符复制到缓冲区。
（2）nyw：将从光标所在位置开始的 n 个字复制到缓冲区。
（3）yy：将光标所在的行复制到缓冲区。
（4）nyy：将光标所在的行开始连续 n 行复制到缓冲区。
（5）p：将缓冲区的字符粘贴到光标所在位置。

注意：所有的 y 命令都要与 p 命令配合使用才可以完成复制和粘贴的功能。

2. 末行模式下的文本复制

在末行模式下，文本复制的命令如下：

```
：行 x，行 y  co 行 z
```

即将文本 x～y 行的内容，复制到 z 行下。例如：

```
：1, 4 co 5
```

该命令执行的结果为，当前编辑的文件 1～4 行复制到 5 行下，即原来文件内容新增了 4 行。

3.4.5 文本移动

在 vim 中移动文本也可以在命令模式和末行模式两种工作模式下完成。

1. 命令模式下文本移动

命令模式下通过以下几步完成：
（1）使用文本删除命令将要移动的文本删除。
（2）使用光标移动命令将光标移动到目标位置。
（3）使用 p 命令将刚刚删除的文本粘贴在目标位置。

2. 末行模式下文本移动

末行模式下进行文本的移动使用如下命令：

```
：行 x，行 y  m 行 z
```

该命令表示当前编辑的文件 x～y 行的文本内容移动到 z 行下。例如：

```
：1, 4 m 6
```

该命令执行的结果为，当前编辑的文本 1～4 行移动到 6 行下。

3.4.6 文本查找与替换

vim 编辑器为在当前行上查找字符提供了较简单的命令，同时也为在工作缓冲区中查找和选择性的替换字符串提供了较复杂的命令。

1. 命令模式下的文本查找与替换

1）文本查找

f 命令：f（Find）命令可以在当前行查找指定字符，并将光标移到该字符下一次出现

的位置。

t 命令：t 命令将光标定位在指定字符下一次出现的前一个字符位置处。

/text：即斜杠（/）后加文本，然后按下回车键，开始搜索（向下）字符串下一次出现的位置。

?text：即问号（?）后加文本，然后按下回车键，开始搜索（向上）字符串前一次出现的位置。

n 命令：n 命令在不必再输入搜索字符串的情况下，重复上一次搜索。

2）文本的替换

文本的替换是指用指定的文本代替原来的文本。vim 提供的替换命令有取代命令、替换命令和字替换命令等。

（1）取代命令 r 和 R。

命令 r<字符>，表示用随后输入的字符代替当前光标处的字符。

例如，rx 表示将当前光标处的字符用 x 取代。

大写的命令 R，表示用其后输入的字符串代替从当前光标到其后面的若干字符，每输入一个字符就取代原来的字符，直到按 Esc 键结束。

（2）替换命令 s 和 S。

命令 s，表示用随后输入的文本替换当前光标处的字符。与 r 命令功能相同，但完成替换后，将工作模式从命令模式转换到插入模式，这一点与 r 命令不同。

命令 S，表示用新输入的文本替换光标当前行（整行），输入 S 命令后当前行被删除，工作模式切换到插入模式，等待输入替换文本。若想回到命令模式，一定要按 Esc 键。

（3）字替换命令 cw 和行替换命令 cc/C。

cw 命令，表示将某个字的内容用其他字符串替换。输入 cw 命令后，光标所在位置到该字字尾的内容被删除，然后可输入任何文本内容。输入 cw 命令后，工作模式会被切换到插入模式，需要按 Esc 键回到命令模式。

命令 cc，表示将光标所在行整行字符用新输入文本替换，输入 cc 后，当前行被删除，工作模式被切换到插入模式，等待输入替换文本。若想回到命令模式，一定要按 Esc 键。大写字符 C 与其功能相同。

2. 末行模式下的文本查找与替换

在末行模式下，替换命令的语法格式如下：

```
：[g][address] s/search-string/replacement-string[/option]
```

其中，address 表示查找的地址；s 为替换命令；search-string 表示查找的字符串；replacement-string 表示用来替换匹配的 search-string。表 3.2 为查找替换示例。

表 3.2　查找替换示例

命　令	描　述
：s/smaller/smallest	将当前行中第一次出现的字符串 smaller 替换为 smallest
：1,. s/ch1/ch2/g	将当前行之前所有行中的字符串 ch1 替换为字符串 ch2
：1,$ s/seven/7/g	将所有出现的字符串 seven 替换为 7
：g/chapter/s/seven/7/	将第一次出现包含字符串 chapter 的所有行中的字符串 seven 替换为 7
：.,.+10 s/int/void/g	将出现的从当前行到后续 10 行内的每个字符串 int 替换为 void

3.4.7 重复与取消

1. 重复命令

重复命令可以方便地执行一次前面刚完成的某个复杂命令。在命令模式下按“.”键即可实现。例如，命令模式下按 dd，之后连续按..，表示删除当前光标所在行开始的连续三行。

2. 取消命令

取消命令，也称为复原命令，可以取消前一次的误操作。在命令模式下，使用 u 键即可实现。

3.4.8 vim 中执行 Shell 命令

使用 vim 时，在末行模式下可以执行 Shell 命令。有以下多种方式可以实现 Shell 命令的执行。

```
：sh
```

按回车键确认后会生成一个新的交互式 Shell 等待用户输入 Shell 命令，执行完 Shell 命令后，通过按 Exit 键回到 vim 编辑环境。

输入下面的命令，可以在 vim 中执行 Shell 命令行。其中，command 为要执行的命令，按回车键结束输入，开始命令的执行。

```
:! command
```

3.4.9 文件的读写

在末行模式下，vim 可将当前工作缓冲区中的内容写到另外一个文件中，或者将一个已存在的文件打开，读入当前的工作缓冲区。

读（：r），该命令表示将文件读入工作缓冲区。读入缓冲区的文本并不覆盖原工作缓冲区的内容，而是被定位到指定的地址之后，如果没有指定地址，那么就定位到当前行。语法格式如下：

```
：[address] r [filename]
```

其中，address 表示指定地址，可以用行号表示；filename 表示待打开的文件。输入完毕后按回车键确认，命令开始执行，执行结束后文件 filename 被读入当前工作缓冲区。

写（：w），该命令表示将工作缓冲区内容的部分或者全部写到某个文件中。使用地址可以将工作缓冲区的部分内容写到由文件名指定的文件中。如果不指定地址和文件，那么 vim 将把整个缓冲区的内容写回到正在编辑的文件中，并更新文件，即保存现在打开的文件内容。语法格式如下：

```
：[address] w [filename]
：[address] w >> [filename]
```

其中，address 表示指定地址，可以用行号表示；filename 表示待写入的文件。输入完毕后按回车键确认，命令开始被执行，执行结束后，当前工作缓冲区内容的部分或全部写入文件 filename 中。

第二种格式中的>>，表示向原有文件追加文本。

3.5　使用 vim 创建 Shell 脚本

vim 是一个灵活的文本编辑器，可以用来编辑 Shell 脚本程序。在 vim 中编辑 Shell 脚本程序的过程和普通的文本编辑是一样的，不同的是编辑的内容都是 Shell 命令。例如：

```
$vim helloShell （Enter 确认，创建文件 helloShell）
```

进入 vim 命令模式下，使用 i 命令切换到插入模式，输入以下文本。

```
clear
echo "============"
echo "Hello World"
echo "============"
sleep 5
clear
echo Host is $HOSTNAME
echo User is $USER
```

输入完成后，按 Esc 键返回命令模式，然后在该模式下，按 ZZ 命令退回到 Shell。关于 Shell 脚本程序编写的详细介绍，请参考本书第 6 章内容。

3.6　使用 vim 创建 C 程序

使用 vim 创建 C 程序的源文件，创建过程也是与普通的文本编辑过程一样的。不同的是，编辑的内容符合 C 语法的程序。例如：

```
$ vi hello.c （vim 编写 hello.c 源程序）
```

按回车键确认后，进入 vim 的命令模式，切换到插入模式，输入以下内容：

```
#include<stdio.h>
main (int argc, char *argv[])
{  int i;
  for(i=0; i<argc; i++)
  printf("%s\n", argv[i]);
  exit(0);
}
```

输入完成后，按 Esc 键回到命令模式，然后在该模式下，按 ZZ 命令退回到 Shell。关于编程部分的详细介绍，请参考本书第 7 章内容。

本章小结

vim 是 Linux 系统中的文本编辑器，功能强大。本章首先介绍了 vim 的 3 种工作模式，说明了每种工作模式下可进行的操作及模式之间的转换关系。接着简单介绍了 vim 的进入和退出，并对 vim 中的编辑操作进行了详细的介绍。然后介绍了编辑操作可分为命令模式下的编辑操作和文本插入模式下的编辑操作，给出了具体的编辑操作命令。最后介绍了如何使用 vim 创建 Shell 基本程序及 C 程序。

本章习题

1．vim 有几种工作模式？各模式之间如何转换？

2．进入 vim 有几种方式？退出 vim 有几种方式？

3．在命令模式下，如何将光标定位到指定行？如何删除文本中的字符、行？如何查找匹配某个模式的行？

4．在末行模式下如何复制一段文本？移动一段文本？替换一段文本？

5．举例说明 vim 在命令模式下的插入命令（I/i）、附加命令（A/a）和打开命令（O/o）的区别。

6．将文本 To err is human. a computer.变成 To err is human，在 vim 中如何操作？

7．将文本 There is something wrong here 中的 wrong 删除，在 vim 中如何操作？

8．使用哪个命令可以在当前工作编辑环境中向后搜索以单词 hello 开始的行？

9．使用哪个命令可以将所有出现的 HELLO 替换为 hello？

10．如何撤销上次操作？