《Linux C 编程从初学到精通》

作者：华清远见

第 3 章 vi 与 Emacs 编辑器

本章目标

文本编辑器是计算机最基本的应用，修改配置文件、编写程序或者建立文件都需要用到它。Linux 提供了齐全的文本编辑器，可以让用户按照自己的喜好进行选择。在 Linux 众多的文本编辑器中，vi 和 Emacs 编辑器是 Linux 用户最为普遍的选择。本章将向读者介绍这两种文本编辑器。本章内容：

vi 编辑器的使用详解。
vi 使用实例。
Emacs 编辑器的使用详解。
Emacs 使用实例。

3.1 vi 的使用

vi 编辑器是 UNIX/Linux 下最基本的文本编辑器，工作在字符模式下。由于不需要图形界面，使它成为效率很高的文本编辑器。尽管在 Linux 上也有很多图形界面的编辑器可用，但 vi 在系统和服务器管理应用中的功能是那些图形编辑器所无法比拟的。

vi 是“Visual Interface”的简称，它在 Linux 上的地位就像 Edit 程序在 DOS 上一样。它可以执行输入、输出、删除、查找、替换、块操作等众多文本操作，而且用户可以根据自己的需要对其进行定制，这是其他编辑程序所没有的。

3.1.1 启动与退出 vi

在 Linux 终端命令提示符下输入 vi（或 vi 文件名），即可启动 vi 编辑器。如：

```
# vi filename
```

按“Enter”键执行该命令，系统便会自动打开文件名为“filename”的文件的 vi 编辑界面，其初始界面如图 3.1 所示。也可以在 X-Window 下通过选择“开始”→“编程”→“Vi Improved”命令来运行 X-Window 下的 vi。

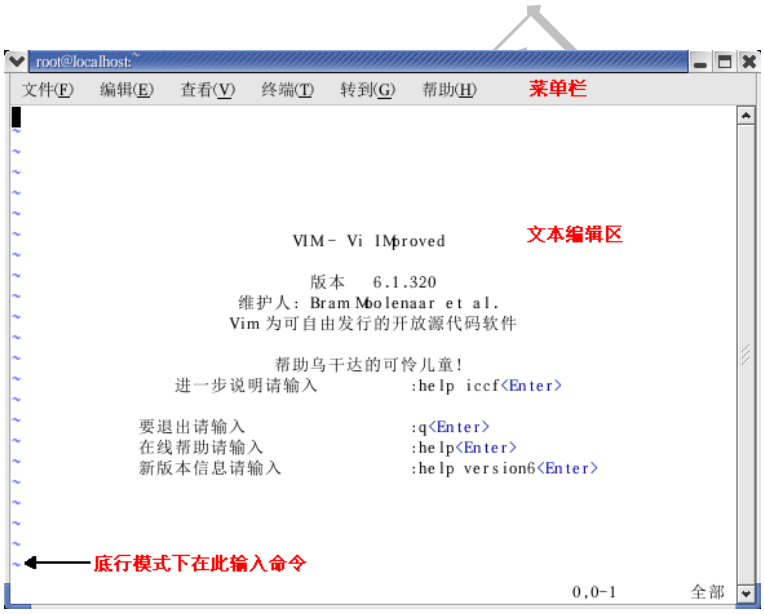

图 3.1 vi 编辑器的初始界面

提示

当使用“vi 文件名”的命令时，若进行编辑的是当前工作目录下已存在的文件，启动 vi 后便可看到该文件中的内容；若是当前目录下不存在的文件，则系统首先创建该文件，再使用 vi 进行编辑。

要退出 vi，必须先按“Esc”键回到命令行模式，然后输入“:”，此时光标会停留在最下面一行（底行模式），再输入“q”，最后按“Enter”键即可退出。

vi 拥有 3 种工作模式：命令行模式（command mode）、插入模式（input mode）与底行模式（last line mode）。3 种模式下的功能可描述如下：

命令行模式，也叫做“普通模式”，它是启动 vi 编辑器后的初始模式。在该模式下，主要是使用隐式命令（命令不显示）来实现光标的移动、复制、粘贴、删除等操作。但是在该模式下，编辑器并不接受用户从键盘输入的任何字符来作为文档的编辑内容。

插入模式，在该模式下，用户输入的任何字符都被认为是编辑到某一个文件的内容，并直接显示在 vi 的文本编辑区。

底行模式，在该模式下，用户输入的任何字符都会在 vi 的最下面一行显示，按“Enter”键后便会执行该命令（当然前提是这是一个正确的命令）。

使用 vi 编辑器，首先必须能够熟练掌握各种工作模式下的功能，以及各种工作模式间的切换。如图 3.2 所示为 vi 3 种工作模式间的切换方法。

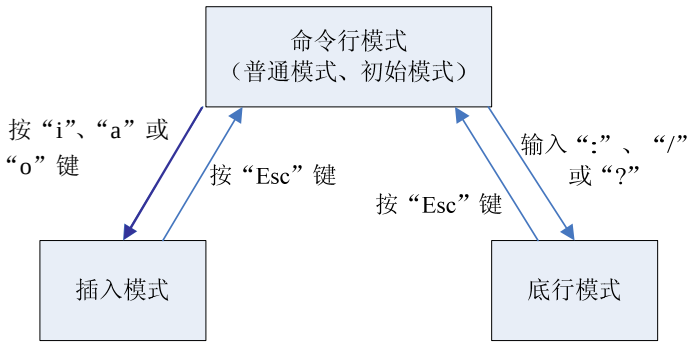

图 3.2 vi 3 种工作模式间的切换

从图 3.2 中可以看到，命令行模式是 vi 编辑器的初始模式，从该模式下可以实现到任何模式的切换。而插入模式和底行模式之间不能相互切换，因为在插入模式下，任何输入的字符都被认为是编辑到某一个文件的内容，而不是命令！而在底行模式下，任何输入的字符都被看做是底行命令（尽管可能是不合法的），二者都必须先通过命令行模式才能进入对方，即需要先按“Esc”键回到初始模式。

下面向读者介绍各种工作模式下的常用命令。

3.1.2 vi 的命令行模式

命令行模式是进入 vi 后的初始模式，在该模式下主要是使用方向键来移动光标的位置，并通过相应的命令来进行文字的编辑。在插入模式下按“Esc”键，或是在底行模式下按“Esc”键，或是在底行模式下执行了错误的命令，vi 都会自动回到命令行模式。本节介绍命令行模式中常用的操作命令，由于这些命令比较多，在此仅进行简单介绍，用户在使用时也可以查阅帮助文档。

1.移动光标

在命令行模式下，一般通过使用上、下、左、右 4 个方向键来移动光标的位置。但是在有些情况下，例如使用 telnet 远程登录时，方向键就不能使用，必须用命令行模式下的光标移动命令。这些命令及作用如表 3.1 所示。

表 3.1 移动光标的常用命令

命令	操作说明
h	向左移动光标
l	向右移动光标
j	向下移动光标
k	向上移动光标
^	将光标移动到该行的开头（指第一个非空字符上）
$	将光标移动到该行行尾，同键盘上的“End”键
0	将光标移动到该行行首，同键盘上的“Home”键
G	将光标移动到文档最后一行的开头（第一个非空字符）
nG	将光标移动到文档的第 n 行的开头（第一个非空字符），n 为正整数
w	光标向后移动一个字（单词）
nw	光标向后移动 n 个字（单词），n 为正整数
b	光标向前移动一个字（单词）

nb	光标向前移动 n 个字（单词），n 为正整数
e	将光标移动到本单词的最后一个字符。如果光标所在的位置为本单词的最后一个字符，则跳动到下一个单词的最后一个字符。“.”、“,”、“#”、“/”等特殊字符都会被当成一个字
{	光标移动到前面的“{”处。这在使用 vi 进行 C 语言编程时很适用
}	同“{”的使用，将光标移动到后面的“}” 处
Ctrl+b	向上翻一页，相当于 Page Up
Ctrl+f	向下翻一页，相当于 Page Down
Ctrl+u	向上移动半页
Ctrl+d	向下移动半页
Ctrl+e	向下翻一行
Ctrl+y	向上翻一行

2. 复制粘贴

复制、粘贴是在编辑文档时最常用的操作之一，可以大大节约用户重复输入的时间。vi 的命令行模式下常用的复制、粘贴命令如表 3.2 所示。

表 3.2 复制粘贴的常用命令

命令	操作说明
yy	复制光标所在行的整行内容
yw	复制光标所在的单词的内容
nyy	复制从光标所在行开始向下的 n 行内容，n 为正整数，表示复制的行数
nyw	复制从光标所在字开始向后的 n 个字，n 为正整数，表示复制的字数
p	粘贴，将复制的内容粘贴在光标所在的位置

3. 删除

vi 编辑器中的删除操作可以是一次删除一个字符，也可以是一次删除多个字符，或者整行字符。vi 命令行模式常用的删除命令如表 3.3 所示。

表 3.3 删除文本的常用命令

命令	操作说明
x	删除光标所在位置的字符，同键盘上的“Delete”键
X	删除光标所在位置的前一个字符
nx	删除光标所在位置及其后的 n-1 个字符，n 为正整数
nX	删除光标所在位置及其前的 n-1 个字符，n 为正整数
dw	删除光标所在位置的单词
ndw	删除光标所在位置及其后的 n-1 个单词，n 为正整数
d0	删除当前行光标所在位置前的所有字符
d$	删除当前行光标所在位置后的所有字符
dd	删除光标所在行
ndd	删除光标所在行及其向下的 n-1 行，n 为正整数
nd+上方向键	删除光标所在行及其向上的 n 行，n 为正整数
nd+下方向键	删除光标所在行及其向下的 n 行，n 为正整数

4. 其他命令

命令行模式下其他常用的命令包括字符替换、撤销操作、符号匹配等，这些也是在我们使用 vi 时经常遇到的命令，其操作说明如表 3.4 所示。

表 3.4 其他常用命令

命令	操作说明
r	替换光标所在位置的字符，例如 rx 是指将光标所在位置的字符替换为 x
R	替换光标所到之处的字符，直到按“Esc”键为止
u	表示复原功能，即撤销上一次操作
U	取消对当前行所做的所有改变
.	重复执行上一次的命令
ZZ	保存文档后退出 vi 编辑器
%	符号匹配功能，在编辑时若输入“%(”，系统会自动匹配相应的“)”

3.1.3 vi 的插入模式

插入模式是 vi 编辑器最简单的模式，因为在此模式下没有那些烦琐的命令，用户从键盘输入的任何有效字符都被看做是写进当前正在编辑的文件中的内容，并显示在 vi 的文本编辑区。

也就是说，只有在插入模式下，才可以进行文字的输入操作。表 3.5 所示为从命令行模式切换至插入模式的几个常用命令。当在插入模式下时，可以按“Esc”键回到命令行模式（参考图 3.2）。

表 3.5 命令行模式切换至插入模式的命令

命令	操作说明
i	从光标所在的位置开始插入新的字符
I	从光标所在行的行首开始插入新的字符
a	从光标所在位置的下一个字符开始插入新的输入字符
A	从光标所在行的行尾开始插入新的字符
o	新增加一行，并将光标移动到下一行的开头开始插入字符
O	在当前行的上面新增加一行，并将光标移动到上一行的开头开始插入字符

3.1.4 vi 的底行模式

vi 的底行模式，也叫“最后行模式”，是指可以在界面最底部的一行输入控制操作命令，主要用来进行一些文字编辑的辅助功能，比如字串搜寻、替代、保存文件，以及退出 vi 等。不同于命令行模式，底行模式下输入的命令都会在最底部的一行中显示，按“Enter”键 vi 便会执行底行的命令。

在命令行模式下输入“:”，或者是使用“？”和“/”键，就可以进入底行模式了。比起命令行模式的诸多操作命令，底行模式的操作命令就少多了，如表 3.6 所示。

表 3.6 底行模式下的常用命令

命令	操作说明
q	退出 vi 程序，如果文件修改过，则必须先保存文件
q!	强制退出 vi 而不保存文件
x	（exit）保存文件并退出 vi
x!	强制保存文件并退出 vi
w	（write）保存文件，但不退出 vi
w!	对于只读文件，强制保存修改的内容，但不退出 vi
wq	保存文件并退出 vi，同 x

E	在 vi 中创建新的文件，并可为文件命名
N	在本 vi 窗口中打开新的文件
w filename	另存为 filename 文件，不退出 vi
w! filename	强制另存为 filename 文件，不退出 vi
r filename	（read）读入 filename 指定的文件内容插入到光标位置
set nu	在 vi 的每行开头处显示行号
s/pattern1/pattern2/g	将光标当前行的字符串 pattern1 替换为 pattern2
%s/pattern1/pattern2/g	将所有行的字符串 pattern1 替换为 pattern2
g/parttern1/s//parttern2	将所有行的字符串 pattern1 替换为 pattern2
num1,num2 s/pattern1/pattern2/g	将行 num1 到 num2 的字符串 partten1 替换为 partten2
/	查找匹配字符串功能。用“/ 字符串”的命令模式，系统便会自动查找，并突出显示所有找到的字符串，然后转到找到的第一个字符串。如果想继续向下查找，可以按“F”键；向前继续查找则按“N”键
?	也可以使用“？ 字符串”查找特定字符串，它的使用与“/ 字符串”相似，但它是向前查找字符串

说明

为便于读者理解，对于表 3.6 中几个常用的字符串替换命令进行如下说明：

百分号（%）表示所有行。

g 为可选标志，带这个标志表示替换将针对行中每个匹配的串进行，否则只替换行中第一个匹配串。

s 表示其后是一个替换命令。

3.2 vi 使用实例

讲到这里，读者可能已经被 vi 编辑器的多个工作模式，以及不同模式下那些复杂烦琐的命令弄得晕头转向了，下面以一个具体的实例向读者演示 vi 编辑器的使用。

我们以第 2 章中的程序 2.12（见 2.9.2 节）为例，在 vi 中输入程序 2.12 所示的源代码，并保存文件名为 vi_test.c，这个程序的功能是判断并输出用户输入的两个数中较大的一个数。通过这个例子向读者演示了 vi 下设置程序代码显示行号、复制与粘贴文本、删除、字符替换等常见操作。

我们将操作步骤详细描述如下：

（1）在 Linux 命令行下输入 vi 命令，启动 vi。

（2）按“i”键，进入 vi 的插入模式，输入程序 2.12 所示的源代码，如图 3.3 所示。

```
#include <stdio.h>
int max(int a, int b);
main()
{
   int a,b;
   printf("Enter a b:");
   scanf("%d %d",&a,&b);
   printf("max = %d\n",max(a,b));
}
int max(int a, int b)
{
   int p;
   p = a>b?a:b;
   return (p);
}
~
~
~
~
~
~
~
-- 插入 --                                    15,2          全部
```

图 3.3 插入模式下输入源程序代码

（3）按“Esc”键，回到 vi 的命令行模式。

（4）输入“:”，进入 vi 的底行模式，然后在底行输入命令“w vi_test.c”，将当前的编辑内容保存为 vi_test.c 文件，但不退出 vi。

（5）再次键入“:”，然后在底行输入命令“set nu”，使程序代码显示行号，如图 3.4 所示。

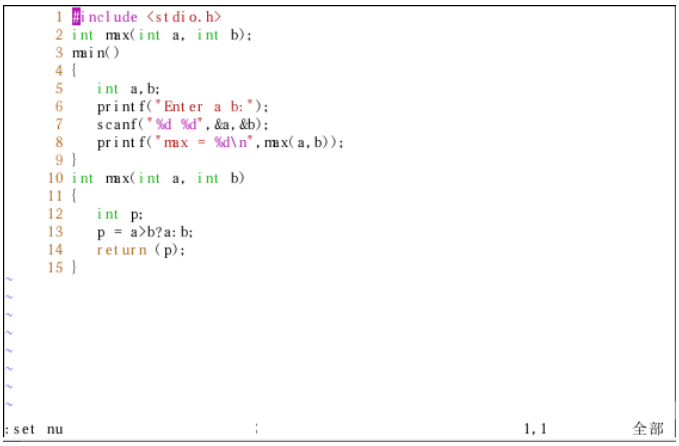

图 3.4 程序显示行号

（6）按“Esc”键，回到 vi 的命令行模式。将光标移至第 5 行，键入“4yy”复制第 5～8 行的内容；然后将光标移至第 8 行行首，键入“p”将复制的内容粘贴在此，如图 3.5 所示。

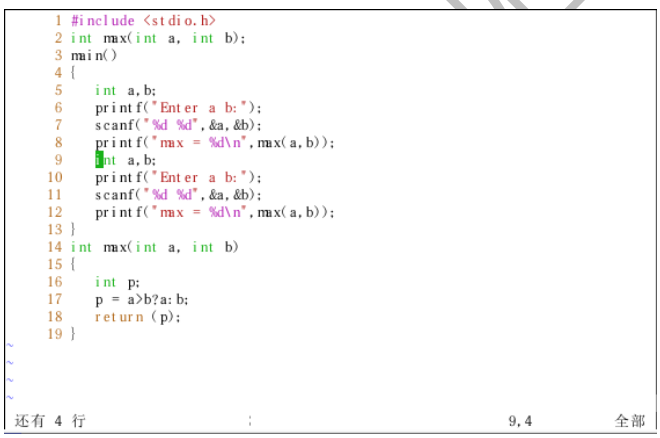

图 3.5 复制与粘贴文本

（7）此时光标在第 9 行行首，直接键入“dd”删除光标所在行（即第 9 行的内容），如图 3.6 所示。

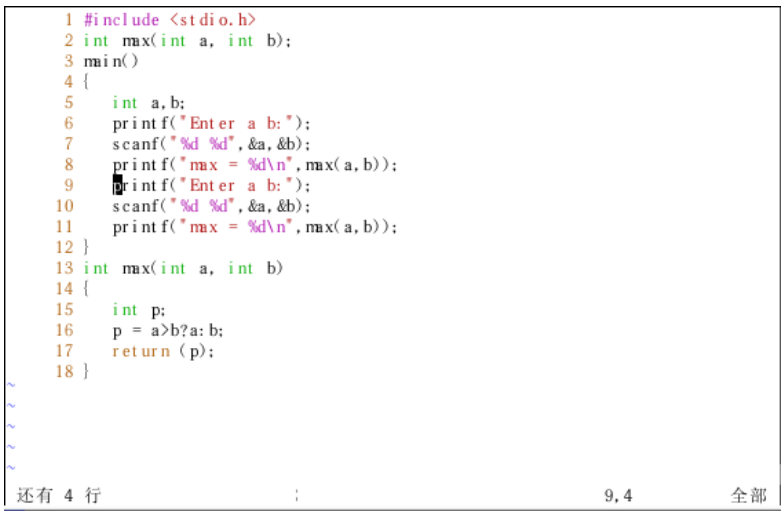

图 3.6 删除文本

（8）键入“:”，进入 vi 的底行模式，然后在底行输入命令“%s/i/n”，将文件中各行的第一个字符“i”替换为字符“n”，键入“Enter”运行该命令，结果如图 3.7 所示。

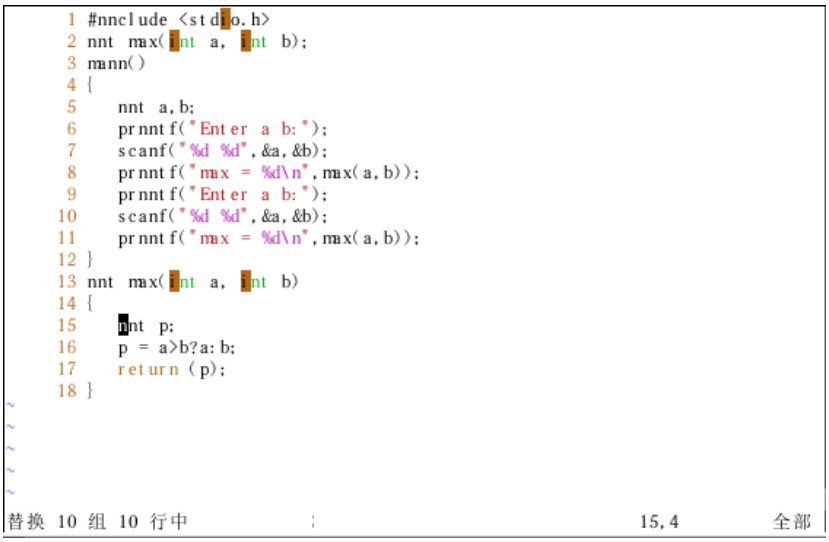

图 3.7 字符替换

说 明

从图 3.7 中可以看到，由于我们使用的命令“%s/i/n”并没有带可选参数“g”，所以只替换了文件各行中的第一个字符“i”，而不是文件中所有的字符“i”（参考表 3.6 及其下面的说明文字）。读者不防试一试使用“%s/i/n/g”命令的结果。

（9）按“Esc”键，回到 vi 的命令行模式。键入“u”撤销上一次的操作。

（10）键入“:”，进入 vi 的底行模式，然后在底行输入命令“wq”，保存当前的修改，并退出 vi。

3.3 Emacs 的使用

Emacs 是 Linux 下一个的功能强大的图形化文本编辑器软件，可以用来编写 C 源程序。Emacs，即 Editor Macros（编辑器宏）的缩写，与 vi 相比，它的一个显著特点是可以使用鼠标进行大部分的操作，对于习惯使用 Windows 系统的用户来说，Emacs 是一个不错的选择。

Emacs 不仅仅是一个文本编辑器，它更是一个整合环境，或称之为集成开发环境。Emacs 是目前世界上最具可移植性的重要软件之一，能够在当前大多数操作系统上运行，包括类 UNIX 系统（GNU/Linux、各种 BSD、Solaris、AIX、IRIX、Mac OS X 等）、MS-DOS、Microsoft Windows 及 OpenVMS 等。

Emacs 既可以在文本终端，也可以在图形用户界面（GUI）环境下运行。使用 GUI 环境下的 Emacs 能够提供菜单（Menubar）、工具栏（toolbar）、滚动条（scrollbar）及上下文菜单（context menu）等交互方式。

Emacs 可以用来编辑文档、收发电子邮件、玩游戏、计算器、浏览网站、查看日历、个人信息管理等。此外，Emacs 支持 Linux 的 Shell 模式，用户可以在 Emacs 中运行 Shell 终端，并在该终端下运行 Shell 命令。也可以直接在 Emacs 的 Shell 模式下运行 shell 命令。Emacs 还支持对多种编程语言的编译、调试功能，包括 C/C++、Java、Perl、Python、Lisp 等语言。

3.3.1 启动与退出 Emacs

在 Linux 终端命令提示符下使用“emacs”或“emacs filename”命令，即可启动 Emacs 编辑器。也可以在 X-Window 下通过选择“开始”→“编程”→“Emacs”命令进入 Emacs，进入 Emacs 的初始界面如图 3.8 所示。

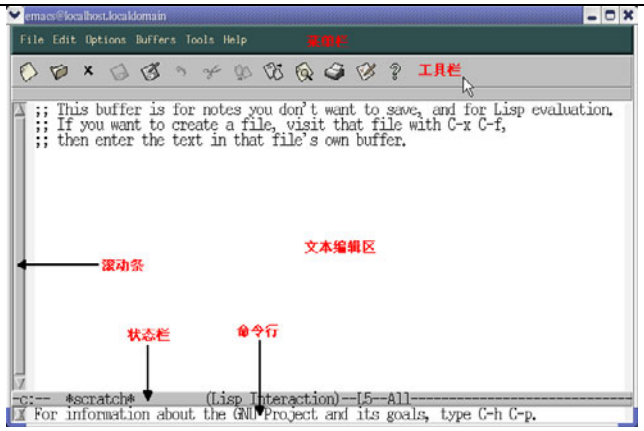

图 3.8　Emacs 的初始界面

要退出 Emacs，直接键入 C-x C-c 即可。

在图 3.8 中，Emacs 的状态栏显示了当前文本编辑区加载（运行）的状态。命令行是用户输入命令（不是快捷键!）的区域，比如键入“C-x C-f”后，在这里输入想要打开文件的文件名，按回车键后便在文本编辑区打开该文件了。

Emacs 的默认工作目录是当前 Linux 用户的主目录，比如在命令行输入“hello”，是指打开主目录（根用户的主目录是 root）下的 hello 文件。

另外，使用 Emacs 编辑器，用户必须了解 Emacs 缓冲区的概念。当用户使用 Emacs 打开或编辑一个文件时，Emacs 将会自动创建一个缓冲区（Buffer），一个文件对应一个缓冲区。用户在窗口中进行编辑操作，输入的字符将会被暂存在缓冲区，当用户执行保存操作时，Emacs 会自动将缓冲区中的内容保存到当前打开文件中。Emacs 允许用户一次打开多个文件，这样就使用了多个缓冲区。

说 明

掌握 Emacs 的快捷键可以说是 Emacs 爱好者的基本功，也是提高编辑速度和质量所必备的，但是初学者可能记不住那么多的快捷键，必要时可以查阅帮助文档或相关书籍，最常用的快捷键数量也就十几个。Emacs 的快捷键都是绑定于“Ctrl”键和“Alt”（或称 Meta）上的，例如“C-x”即“Ctrl+x”，“M-x”即“Alt+x”。当然，所有的按键都可以自定义。

3.3.2　Emacs 下的基本操作

本小节中将遇到较多的快捷键的操作，在此有必要先说明键盘操作符号的意义。

C-x：同时按“Ctrl 键”和“x 键”。
C x：先按“Ctrl 键”，然后释放它，再按“x 键”。
M-x：同时按“Alt 键”和“x 键”。
M x：先按“Alt 键”，然后释放它，再按“x 键”。

1. 文件操作

选择 Emacs 菜单栏中的“File”命令，在下拉菜单中是一些与文件相关的操作。表 3.7 列出了这些操作中主要的快捷键功能说明。

表 3.7　文件操作相关的快捷键

快捷键	操作说明
C-x C-f	打开 Emacs 默认目录（用户主目录）下的某个文件
C-x d	打开文件路径，将查看某个文件的属性信息，并在这个文件上进行编辑操作
C-x i	将某个文件的内容插入到当前的缓冲区

C-x C-v	打开一个文件，取代当前缓冲区
C-x C-s	保存文件
C-x C-w	将当前缓冲区另存为新的文件
C-x C-q	切换为只读或者读写模式
C-x C-c	退出 Emacs

2. 编辑操作

选择 Emacs 菜单栏中的“Edit”命令，在下拉菜单中可以看到与文本编辑相关的操作。如表 3.8 所示是这些操作中主要的快捷键功能说明。当然，如果我们不想花时间去记忆这些烦琐的快捷键，则完全可以使用键盘上的按键，也可以使用下拉菜单中的命令。就像 Windows 下的 Word 软件，我们记忆的快捷键也只有区区几个而已。

表 3.8　编辑操作相关的快捷键

快捷键	操作说明	快捷键	操作说明
C-f	光标前进一个字符	M->	光标移动到文件尾部
C-b	光标后退一个字符	C-M-f	向前匹配括号
M-f	光标前进一个字	C-M-b	向后匹配括号
M-b	光标后退一个字	C-i	将光标所在位置居中
C-a	光标移动到行首	M-n or C-u n	重复操作随后的命令 n 次
C-e	光标移动到行尾	C-u	重复操作随后的命令 4 次
M-a	光标移动到句首（第一个非空字符）	C-u C-u	重复操作随后的命令 8 次
M-e	光标移动到句尾（最后一个非空字符）	C-x ESC ESC	执行历史命令记录，M-p 选择上一条命令，M-n 选择下一条命令
C-p	光标移动到上一行	C-d	删除一个字符
C-n	光标移动到下一行	M-d	删除一个字
C-v	向下翻页	C-k	删除一行
M-v	向上翻页	M-k	删除一句
M-<	光标移动到文件头部	C-_	撤销操作

3. 窗口操作

窗口就是指 Emacs 的文本编辑区，用户可以使用多个窗口来对同一个缓冲区的不同部分进行操作，比如可以使用“C-x 2”快捷操作使当前编辑区垂直均分为两个窗口；也可以对不同的缓冲区进行操作。如图 3.9 所示为将编辑区水平均分为两个窗口。

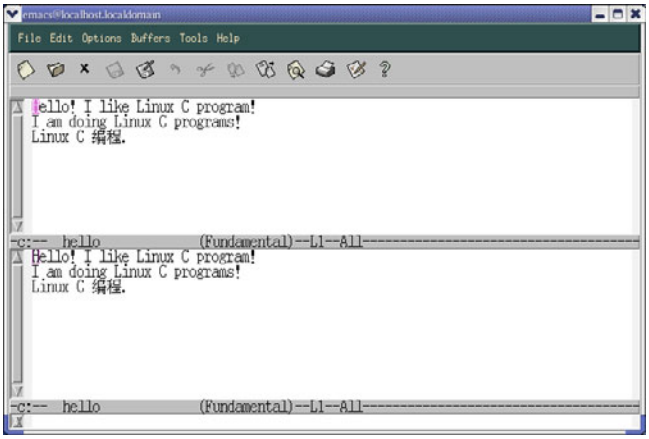

图 3.9　编辑区水平均分为两个窗口

选择 Emacs 菜单栏中的“Buffers”命令，可以看到在其下拉菜单中列出了当前打开的所有缓冲区（文件），用户可以点击不同的文件名，将当前的工作窗口切换至该缓冲区下。当然，也可以使用一些快捷键来对窗口进行操作，它们的说明如表 3.9 所示。

表 3.9　窗口操作相关的快捷键

快捷键	操作说明	快捷键	操作说明
C-x 0	关闭当前窗口	C-x s	保存所有窗口缓冲
C-x 1	只留下一个窗口	C-x b	选择当前窗口的缓冲区
C-x 2	垂直均分窗口	C-x ^	纵向扩大窗口
C-x 3	水平均分窗口	C-x }	横向扩大窗口
C-x o	切换到其他窗口		

4. 缓冲区列表操作

选择“Buffers”下拉菜单中的“List All Buffers”选项（或使用 C-x C-b），将在当前窗口打开所有的缓冲区列表。上下移动光标选中不同的行，键入“Enter”后便可对相应的文件（缓冲区）进行操作。在缓冲区列表中的操作快捷方式如表 3.10 所示。

表 3.10　缓冲区列表的操作

快捷键	操作说明	快捷键	操作说明
C-x C-b	打开缓冲区列表	u	取消标记
d or k	标记为删除	x	执行标记的操作
~	标记为未修改状态	f	在当前窗口打开该缓冲区
%	标记为只读	o	在其他窗口打开该缓冲区
s	保存缓冲		

5. 程序编译

Emacs 可以支持多种编程语言的编辑模式，比如 C、C++、Java 等语言。用户可以通过键入“M-x [language]-mode”命令（M-x 是快捷键操作，[language]-mode 是命令行输入的命令）来选择各种不同语言模式的编辑环境，[language]表示不同的编程语言。

比如我们首先键入“M-x”快捷方式，进入 Emacs 的程序编译模式，然后在命令行区域输入“c-mode”，按回车键后将出现如图 3.10 所示的界面。

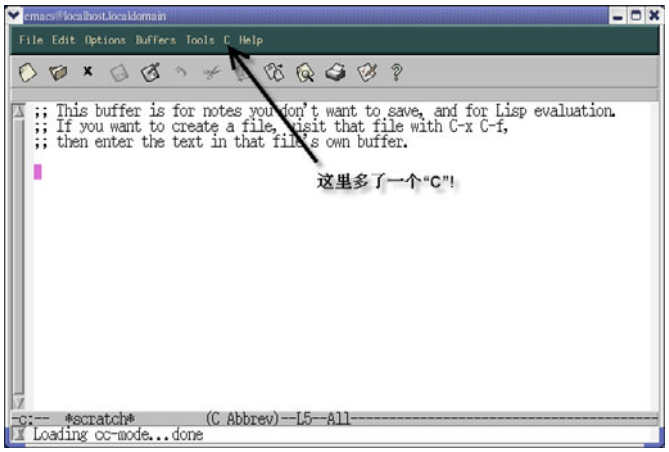

图 3.10　Emacs 的 C 编辑模式

可以看到，在图 3.10 中，编辑器的菜单栏中多了一个“C”选项，表明 Emacs 编辑区进入了 C 语言的编辑模式。读者可以试试输入“c++-mode”或者“Java-mode”命令时，编辑器的菜单栏将会发生怎样的变化。

在程序编译模式下，当用户输入代码时，Emacs 支持自动缩进显示。事实上，Emacs 可以支持多种缩进风格，在 C 模式下，用户可以通过使用“M-x c-set-style”命令来选择自己需要的缩进风格。

Emacs 不仅仅是一个编辑器，更是一个集成开发环境，可以使用它来进行 C 程序（当然也可以是其他的程序设计语言）的编译和调试。用户在“Tools”菜单中找到“Compile”选项，或者直接键入“M-x compile”，就可以在 Emacs 的命令行输入编译命令。

如果有 Makefile 文件（将在第 5 章向读者介绍），就接受默认设置，使用“make -k”命令来编译程序。编译出现错误和警告时，程序员可以单击鼠标来定位这些警告和错误。

出现了警告和错误信息，就离不开调试这一重要步骤。Emacs 还支持程序的调试功能，用户可以使用“M-x gdb”命令来调用 Linux 下的 gdb 调试器，或者在 Tools 菜单中选择 gdb 选项，然后即可输入调试命令。

表 3.11 列出了在程序编译模式下的常用快捷操作。

表 3.11　程序编译模式下的操作

快捷键	操作说明	快捷键	操作说明
M-x compile	执行编译操作	M-x xdb	调用 xdb 调试器
M-x gdb	调用 gdb 调试器	M-x sdb	调用 sdb 调试器
M-x dbx	调用 dbx 调试器		

在表 3.11 中，gdb、dbx、xdb 和 sdb 是 Linux 下的各种程序调试工具（将在第 4 章中向读者介绍），当用户输入不同的命令时，Emacs 便会自动调用 Linux 下的这些调试器来对当前缓冲区中的程序进行调试。所以，与其说 Emacs 是一个文本编辑区，不如说它更像是一个功能强大的集成开发环境。

6. 搜索模式

Emacs 支持对当前窗口文件中的字符搜索功能，这无疑会使 Emacs 下的文本编辑工作变得更加方便、适用。Emacs 的字符搜索相关的快捷键操作如表 3.12 所示。

表 3.12　字符搜索操作

快捷键	操作说明
C-s 字符	向前搜索字符，查找到的字符以蓝色字体显示
ENTER	停止搜索
C-r 字符	向后搜索字符，查找到的字符以蓝色字体显示
C-s C-w	以光标所在位置的字为关键字进行搜索
C-s C-s	重复上一次搜索
C-r C-r	重复上一次反向搜索
C-s ENTER C-w	进入单词搜索模式，搜索完毕后，光标停留在查找到的第一个单词的后面
C-r ENTER C-w	进入反向单词搜索模式
C-r	在进入查找/替换模式后，该命令进入迭代编辑模式
C-M-x	退出迭代编辑模式，返回到查找/替换模式

7. Shell 模式

Emacs 编辑器最显著的特点之一是它支持 Linux 的 Shell 模式，用户可以在 Emacs 的文本编辑区运行 Shell 终端，并在该终端下运行 Shell 命令。比如键入“M-x”快捷键后，在 Emacs 的命令行输入“shell”，按回车键后，Emacs 便会在当前窗口打开一个 Shell 终端。我们在 Shell 终端中运行命令，如图 3.11 所示。

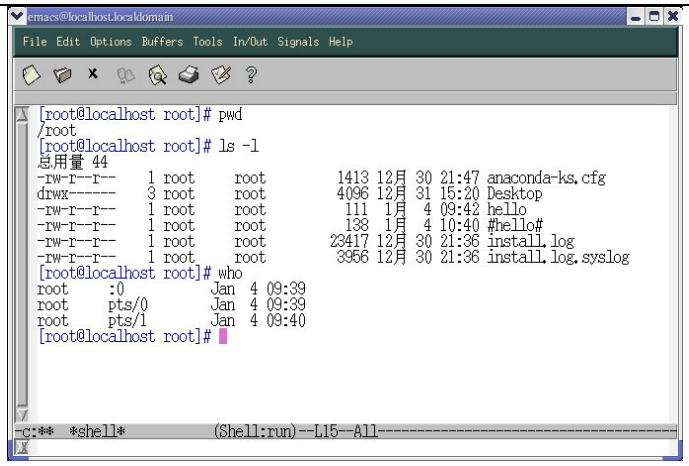

图 3.11　Emacs 下运行 Shell

另外，也可以直接在 Emacs 的命令行中执行 Linux Shell 的任何命令，并将执行结果输出在文本编辑区。选择菜单栏中“Tools”选项下的“Shell command”，或键入“M-!”，即可进入 Shell 模式。

例如，键入“M-!”后，Emacs 进入 Shell 模式，此时在 Emacs 的命令行输入“ls -l”命令，按回车键后编辑器的界面如图 3.12 所示。

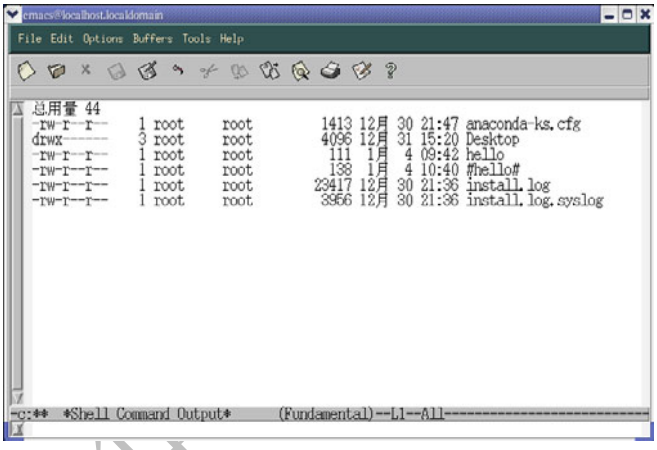

图 3.12　Emacs 中执行 Shell 命令

在图 3.12 中可以看到，“ls -l”命令的执行结果显示在 Emacs 的文本编辑区。由于 Shell 命令的输出是在一个编辑缓冲区里，因此我们可以对它进行编辑、保存等操作。所以，在想要保存 Linux 某一个 Shell 命令的执行结果的场合中，Emacs 的这种模式就显得十分适用。

表 3.13 列出了在 Emacs 的 Shell 模式下常用的快捷键操作说明。

表 3.13　执行 shell 命令的快捷键

快捷键	操作说明
M-x shell	打开 Shell
M-!	执行 Shell 命令（Shell-command）
M-1 M-!	执行 Shell 命令，命令的输出插入在光标当前位置，而不打开新的输出窗口
M-|	针对某一特定区域执行 shell 命令（Shell-command-on-region）
M-! M-p	执行前一条 shell 命令，同 M-!+向上方向键
M-! M-n	执行下一条 shell 命令，同 M-!+向下方向键

此外，Emacs 还具有很多其他的功能，比如收发电子邮件、玩游戏、计算器、浏览网站、查看日历、个人信息管理等，鉴于篇幅和本书的介绍范围，在此不一一列举，用户也可以查看 Emacs 的帮助手册来获得更多的信息。

3.4 Emacs 使用实例

本节以 3.2 节中使用 vi 编辑好的 vi_test.c 文件为例，在 Emacs 中编译并运行该程序，并将程序的运行结果保存到文本文件 vi_test_result 中，主要向读者演示了 Emacs 的 C 程序编译模式与 Shell 模式下的常见操作。我们将操作步骤详细描述如下：

（1）在 Linux 命令行下输入“emacs vi_test.c”命令，使用 Emacs 编辑器打开 vi_test.c 文件。

（2）键入“M-x c-mode”命令，进入 Emacs 的 C 程序编译模式，如图 3.13 所示。

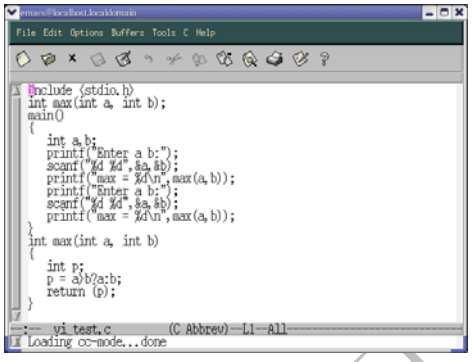

图 3.13　vi_test.c 的编译模式

（3）键入“M-x compile”，此时便可以在 Emacs 的命令行输入 C 程序编译命令，这里输入编译命令“gcc -o vi_test vi_test.c”，按回车键后程序的编译结果如图 3.14 所示。

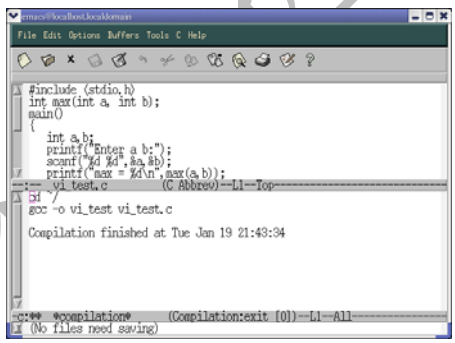

图 3.14　调用 gcc 编译 vi_test.c

（4）将光标定位在编辑区的下面一个窗口，然后键入“C-x 0”关闭当前窗口。

（5）键入“M-x shell”，按回车键后 Emacs 自动在当前窗口打开一个 Shell 终端。

（6）在该 Shell 终端中运行编译通过的可执行程序 vi_test，如图 3.15 所示。

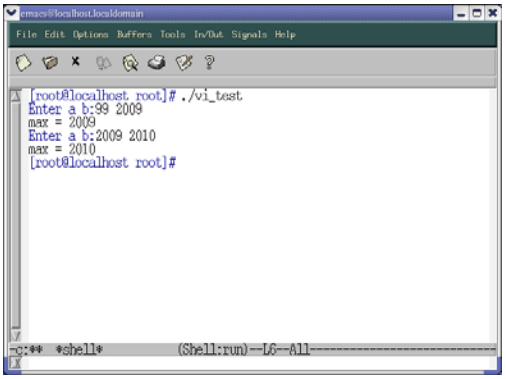

图 3.15　调用 shell 执行 vi_test 程序

（7）键入“C-x C-w”，将当前缓冲区的内容保存为新的文件，我们在命令行输入想要保存的文件名 vi_test_result，按回车键后便将 vi_test 程序的执行结果保存到 vi_test_result 文本文件中去了。

（8）键入“C-x C-c”，退出 Emacs。

3.5 本章小结

本章向读者介绍了 Linux 下最常见的两种编辑器 vi 和 Emacs，它们不仅是文本编辑器，也是进行程序开发的环境，因此熟练掌握和应用这两种编辑器，尤其是它们各种工作模式下的命令及快捷操作，是进行 Linux 下 C 程序开发的基本本领。

实战演练

1. 使用 vi 编辑器编辑下面这段文字：

```
Hello! I like Linux C program!
I am doing Linux C programs!
Linux C 编程.
```

保存为文件 hello.txt，然后退出 vi。

2. 验证 vi 编辑器 3 种工作模式间的切换命令。

3. 验证当使用“A”、“I”、“O”命令将 vi 从命令行模式切换至插入模式时，分别与“a”、“i”、“o”有何不同？

4. 试用 vi 打开/usr/src/linux-x.x.xx/kernel（x.x.xx 表示 Linux 的内核版本号，不同用户会有所不同）目录下的 fork.c 文件，在 vi 的命令行模式下复制某一行的内容，粘贴至下一行，然后再取消当前的操作，并强制退出。

5. 用 vi 打开/usr/src/linux-x.x.xx/kernel 目录下的 fork.c 文件，查找该文件中的字符串“fork”，使其以红色字体显示。

6. 在 Emacs 中打开一个 Shell 终端，执行 ls -l 命令，并将命令执行的结果保存到文件 ls_result.txt 中。

7. 用 Emacs 编写一个 Hello World 的程序，保存为 hello.c 文件，并编译运行该程序。程序如下：

```
#include <stdio.h>
main()
{
   printf(“Hello, Linux World!\n”);
}
```

8. 用 Emacs 打开 hello.c 文件，将当前窗口均分为 4 个工作窗口，在不同的窗口对同一文件的不同部分进行编辑。

9. 用 Emacs 打开/root 目录下的 install.log 文件，查找该文件中的字符“i”，使其以蓝色字体显示。

10. 在 Emacs 中执行 Shell 命令，比如 ls、pwd、who、uname 等。

联系方式

集团总部地址：北京市海淀区西三旗悦秀路北京明园大学校内 华清远见教育集团

北京地址：北京市海淀区西三旗悦秀路北京明园大学校区，电话：010-82600386/5

上海地址：上海市徐汇区漕溪路 250 号银海大厦 11 层 B 区，电话：021-54485127

深圳地址：深圳市龙华新区人民北路美丽 AAA 大厦 15 层，电话：0755-25590506

成都地址：成都市武侯区科华北路 99 号科华大厦 6 层，电话：028-85405115

南京地址：南京市白下区汉中路 185 号鸿运大厦 10 层，电话：025-86551900

武汉地址：武汉市工程大学卓刀泉校区科技孵化器大楼 8 层，电话：027-87804688

西安地址：西安市高新区高新一路 12 号创业大厦 D3 楼 5 层，电话：029-68785218

广州地址：广州市天河区中山大道 268 号天河广场 3 层，电话：020-28916067