

目录

前言	1
第一章 Emacs 的基本概念	17
Emacs 简介	17
理解文件与编辑缓冲区	19
编辑模式	20
启动 Emacs	22
Emacs 的编辑画面	23
Emacs 命令	26
打开一个文件	29
保存文件	33
退出 Emacs	34
获取帮助	34
小结	37
第二章 文件编辑	40
光标的移动	42
文本的删除	48

文本块及其编辑操作	53
段落重排	59
编辑技巧和快捷键	61
命令的中止和修改的撤销	63
对 Emacs 进行定制	67
第三章 查找和替换操作	73
查找操作	73
查找和替换操作	80
拼写检查	89
单词简写模式	99
第四章 使用编辑缓冲区和窗口	107
文件、编辑缓冲区和窗口	107
同时使用多个编辑缓冲区进行工作	109
使用窗口进行编辑	115
在文档中使用书签	130
临时性地挂起 Emacs	135
使用多个 X 窗口进行编辑	137
第五章 Emacs 工作环境	143
在 shell 编辑缓冲区里执行 UNIX 命令	143
文件和目录操作	155
Emacs 中的打印操作	171
用 Emacs 查阅 UNIX 的在线文档	173
时间管理工具的使用	173
用好 Emacs 工作环境	183

第六章 电子邮件和 Usenet 新闻	184
Emacs 的电子邮件功能	184
用 Emacs 发送邮件	185
用 Emacs 读取邮件	195
用 Gnus 读取 Usenet 新闻	216
第七章 Emacs 的因特网工具箱	240
Emacs 的 Telnet 模式	241
Emacs 的 Ange-ftp 模式	245
用 W3 模式浏览 Web 主页	249
第八章 简单的文字排版和特效编辑	261
文本的缩进	262
文本的居中	272
插入分页符	274
矩形编辑	275
绘制简单的图形	281
Emacs 的大纲模式	291
第九章 用 Emacs 设置排版标记	300
设置 troff 和 nroff 排版标记	302
设置 TeX 和 L ^A T _E X 排版标记	309
编写 HTML 文档	316
Emacs 的 Html-helper 模式	317
第十章 Emacs 中的宏	334
什么是宏	334
定义宏	335
向现有的宏里增加编辑命令	342
命名并保存宏	343

执行一个已命名的宏	344
建立复杂的宏	345
LISP 函数 —— 宏的补充	350
第十一章 对 Emacs 进行定制	352
键盘的定制	353
终端支持	360
Emacs 变量	364
Emacs 的 LISP 程序包	366
自动模式的定制	366
第十二章 程序员的 Emacs	369
语言编辑模式	370
C 和 C++ 模式	377
LISP 模式	391
FORTRAN 模式	399
对程序进行编译	405
第十三章 用 LISP 语言对 Emacs 做进一步开发	408
LISP 语言简介	409
LISP 语言的基础函数	420
Emacs 的内部函数	425
主编辑模式程序设计实例	441
对现有编辑模式进行定制	451
建立自己的 LISP 开发库	458
第十四章 Emacs 编辑器和 X 窗口系统	461
Emacs 的 X 界面	461
让 Emacs 使用 X 字体和颜色	465
定制 Emacs 在 X 环境中的显示情况	467

通过 .Xdefaults 文件进行定制	467
属性、窗格、菜单和鼠标事件	469
与 X 服务器进行通信	479
良好的 X 程序设计风格	480
第十五章 Emacs 下的版本控制	482
版本控制的用途	482
版本控制的有关概念	483
VC 对基本操作的辅助作用	485
修改注释的编辑	486
VC 命令汇总	487
VC 模式的标志	488
使用哪一种版本控制系统	488
VC 命令细说	489
对 VC 进行定制	495
对 VC 进行扩展	497
VC 的不足之处	498
有效地使用 VC	499
第十六章 在线帮助	500
Emacs 的自动补足功能	500
帮助命令	504
针对复杂 Emacs 命令的帮助功能	512
附录一 如何获得 Emacs 软件	513
附录二 解除他人对 Emacs 的定制设置	521
附录三 Emacs 变量	523

附录四 Emacs LISP 程序包	534
附录五 软件漏洞及其修补	541
附录六 Emacs 的版权文件	543
附录七 请支持自由软件基金会	555
附录八 Emacs 编辑命令速查表	556
词汇表	575



前言

Emacs 是迄今为止功能最为强大的文本编辑器。它与其他大多数编辑器（特别是 UNIX 操作系统的标准编辑器 vi）的不同之处在于 Emacs 是一个完备的工作环境。不管你做什么，都可以在清晨启动 Emacs，然后一整天都用它来工作：可以用它对文件进行编辑、重命名、删除和复制等操作；可以对程序进行编译；可以与 UNIX 操作系统的 shell 进行交互式操作；可以阅读和组织电子邮件；可以访问因特网等等。在 X 等窗口系统流行之前，人们通常把 Emacs 单独当做一个完备的窗口化系统来使用。只要有一台终端，就可以在 Emacs 环境里永不停息地工作。Emacs 还具备无穷的灵活性：你可以编写自己的命令，能够更改与 Emacs 命令关联的按键；如果愿意花时间，可以用它做任何你想做的事情。

学习本书的目的

因为 Emacs 能够完成如此多的工作，所以人们普遍认为它是一种极其复杂的编辑软件。到目前为止，已经出版的 Emacs 书籍大都是一些综合性的参考手册，不太适合作为 Emacs 初学者的入门教材。这正是我们编写本书的原因：教大家如何从最基本的东西开始去熟悉和掌握 Emacs。本书首先介绍 Emacs 的基础，然后过渡到一些更高级的功能。

我们努力使本书最大限度地满足不同读者的学习需要：也许你是一位只想用 Emacs 来写电子邮件消息和办公室留言的企业管理人员；也许是一位需要用 Emacs 来撰写

充满版式代码的复杂文档的职业作家；还可能是一位希望用 Emacs 来对源代码进行排版的高级程序员。不管你想用 Emacs 做些什么，你都会发现它其实很容易学习和掌握的。只要有一两次实际使用经验，你就能够了解基本的文件编辑方法。

在掌握了 Emacs 的基本用法之后，大家可以继续学到更高级的技巧。只有掌握了它们，你才能真正体会到 Emacs 的妙处。这些高级技巧包括：

- 使用多个窗口和编辑缓冲区，以同时操作多个文件。
- 对键盘命令进行定制。
- 使用变量设计 Emacs，使它适应用户个人的工作习惯。
- 使 Emacs 成为能够完成各种日常任务的工作环境，比如阅读电子邮件、编译程序和执行 shell 命令等。
- 创建宏编辑命令以快速完成重复性的操作任务。
- 用 Emacs 支持多种程序设计语言（其中包括 C、C++、LISP 和 FORTRAN）的编程工作。
- 利用各种标记语言（markup language）对文件进行排版。
- 利用 Emacs 的单词缩略功能，避免拼写过长的短语或者纠正常见的拼写错误。
- 利用 Emacs 访问因特网资源。

当然，这么多的论题不见得都适用于你的具体情况。有些专题可以在掌握了 Emacs 的基本用法之后再深入学习，第一次阅读本书时可以先跳过去。在这篇前言的结尾，我们给出了一些学习本书的不同方法，请大家根据自己的兴趣和经验作出具体的选择。

阅读本书需要满足一些最基本的要求。我们假定大家对 UNIX 操作系统都有一个基本的了解（或者，如果你不了解 UNIX，那么至少应该对正在使用的操作系统有一定的了解）。具体说来，你必须知道文件和目录指的是什么，怎样给它们起名字，对它们都能进行哪些基本操作（如复制、删除和重命名等操作；在 UNIX 操作系统里，这些操作要分别用 `cp`、`rm` 和 `mv` 命令来完成）。如果你对 UNIX 完全不了解，我们建议你先去学习一下 Grace Todino、John Strang 和 Jerry Peek 等人合著的《Learning the UNIX Operating System》（《UNIX 操作系统》）一书，这本书也是由 O'Reilly & Associates 出版公司出版的。

Emacs 的版本

Emacs 有许多不同的版本，提供了丰富多彩的功能。单就 UNIX 操作系统来说，Emacs 最重要的版本包括 GNU Emacs（也就是我们将在本书里讨论的）、Unipress、JOVE、MicroEmacs、Freemacs、MG、Epsilon 和 CCA。某些版本，包括 GNU Emacs 在内，可以在 MS-DOS 下的个人电脑上运行；其他一些版本只提供了有限的功能集，但优点在于占用的内存和 CPU 周期要相对节省。“主流”的 Emacs 版本要求计算机必须有比较好的配置，多个用户共享一台中央计算机的时候更是如此。有些类 Emacs 的编辑器（比如 Epoch 和 xemacs）与 X 窗口系统结合得非常紧密。有些版本是“free”（有自由、免费等几种含义）软件，另外一些则可能是相当昂贵的商业化产品。还有一些软件程序（比如 FrameMaker）虽然不在 Emacs 的家族中，但却有与之相似的键盘命令集。

作为一个编辑器家族，不同的 Emacs 编辑器之间会有一定的差别，我想这一点大家能够有所了解。但这些编辑器之间还是有很多相似之处的，比如说：它们都可以自由地把文本和编辑命令混在一起使用；大多数编辑器都支持多窗口操作；都有一些供编辑特殊类型文件（比如 C 语言程序、TeX 文件等等）用的支持功能；它们中的大多数都允许使用者进行大量的定制，而这些定制通常都是通过某些 LISP 程序设计实现的。

既然有这么多让人迷惑的差异，我们到底介绍哪一种版本好呢？这本书介绍的是 GNU Emacs 编辑器。自从 GNU Emacs 编辑器问世以来，它已经成为 UNIX 界最流行的 Emacs 实现版本，而这一状况目前还没有改变的迹象。Emacs 家族本身已经是异常强大和灵活的了，而 GNU Emacs 又可以说是其中之最。只要你掌握了 GNU Emacs，就可以毫不费力地掌握其他 Emacs 实现版本；反之，却不见得有这么简单。基于以上几点，我们惟一合乎逻辑的选择就是把这本书的重点放在 GNU Emacs 上。

不过，这本书的读者群却并非仅限于 GNU Emacs 用户。因为不同的 Emacs 实现版本之间有很多相似之处，所以本书完全可以帮你学习任何一种 Emacs 编辑器。一种 Emacs 编辑器之间所使用的键盘命令与另一种编辑器所使用的几乎没什么差别——你会看到“C-n”（即 **CTRL-n** 组合键）几乎总是代表“移动到下一行”。各种 Emacs 编辑器之间的差别主要体现在它们的高级命令和特色功能方面；不过，如果你准备利用某些高级功能而使用的却不是 GNU Emacs，那最好还是换成 GNU Emacs。

我们将在这本书里讨论的 GNU Emacs 编辑器是它的第 19 版。准确地说，是它的 19.30 版。本书的第一版是以 GNU Emacs 的第 18 版为重点的。在第 19 版里，GNU Emacs 的操作界面有了很大变化，除增加了一些字体和颜色之外，在以下几方面也有了升级：菜单、卷屏条、增强的 X 窗口系统支持和大多数标准按键（比如 **PAGE UP**、**PAGE DOWN**、**HOME** 和 **END** 等按键）的功能绑定等。有些过去需要由用户自行安装的 LISP 包，现在已经包括在标准的发行版本中，其中两个值得注意的是 Gnus 工具和 ange-ftp 模式，前者是一个 Usenet 新闻阅读器，后者是 FTP (file transfer protocol) 的一个透明接口。如果没有特殊说明，我们在这本书里介绍的命令、菜单和功能都将以 GNU Emacs 19.30 版为准（注 1）。

GNU Emacs 和自由软件基金会

使用 Emacs 编辑器并不意味着你必须了解它的发展史，但它的起源确实是近代计算机史中不能不提的故事。负责维护和发行 GNU Emacs 的自由软件基金会（Free Software Foundation，简称 FSF）已经成为计算机文化中一个非常重要的角色。

很久以前（1975 年），Richard Stallman 在麻省理工学院编写出了第一个 Emacs 编辑器。据说，最初的 Emacs 编辑器是 TECO 的一组宏编辑命令。这是一种很难学习的行编辑器，目前已经退出了历史舞台。而“Emacs”这个名字的意思就是“Editing Macros（编辑用宏命令）”。另外一种说法是“Emacs”起源于 Richard Stallman 特别喜欢的某个冰激凌店的名字。1975 年之后发生了许多事情，TECO 逐步退出了人们的视野，而 Emacs 则被重新编写为一个独立的软件程序。Emacs 曾经有过几个商业化版本，其中最重要的是 Unipress Emacs 和 CCA Emacs。曾有一段时间，在学术界以外你最有可能遇见的 Emacs 编辑器就是这些商业化的实现版本。

注 1：大家必须了解，Emacs 19 从第 19.12 版到第 19.30 版已经有了很大的改进，特别是第 19.12 版和第 19.22 版之间在功能方面有巨大的跃进。第 19.30 版为文本界面提供了菜单选项。此外，第 19.30 版对 Gnus 包做了升级，使它成为一个与早期版本中的 Gnus 包有很多不同之处的新版本。同时，菜单选项的安排也更加合理化了。如果你习惯于使用 Emacs 的菜单，不管是通过 X 界面还是通过文本界面，可能会发现以前熟悉的菜单选项移到另外一个不同的菜单里去了。这些小变化通常不会造成什么麻烦，但如果你按照我们的指导去进行操作却发现结果稍有差别，现在就应该知道是什么原因了。欢迎大家把自己的疑问（或者意见、建议）寄到电子邮件地址 deb@ora.com。

Stallman 的 Emacs 在自由软件基金会 (FSF) 和 GNU 项目出现之后逐步占据了主导地位。GNU 是 “GNU's Not UNIX (GNU 不是 UNIX)” 的首字母缩写，它指的是一个由 Stallman 及其伙伴们共同努力构建的完备的类 UNIX 操作系统。Stallman 创建自由软件基金会的目的就是为了保证某些软件能够永保自由 (free)。这里所说的“自由”并不意味着廉价。你可能必须支付一笔软件发行成本方面的费用，它的确切含义是指人们从软件使用许可方面的限制条款中解放出来。

为了理解“自由”这个词的含义，我们有必要先来看看软件通常是如何发行的。大多数商业软件在发行时都带有一个限制性很强的许可证。你必须先支付费用才能使用某个程序；你可能必须为运行此程序的每台计算机分别支付一定的费用；而为了能够继续使用此程序，你可能还得每年再交一些费用，在某些情况下，你甚至可能不得不按分钟计算来支付使用费。这些许可证几乎百分之百地不允许你把有关程序赠送给朋友，而且你几乎永远也见不到程序的源代码（除非你非常有钱）。如果某个商业化的程序出了问题或者它不具备你需要的功能，就只能等卖给你程序的那家公司的怜悯了，而那些公司极有可能根本不理睬你。

而 GNU Emacs 则没有这样的限制条款。只要你能找到一个愿意把它送给你的人（通常总能找到这样的人），就可以完全免费地得到它。它在许多开放性的软件仓库站点（包括一些匿名 FTP 站点）上都可以找到（我们将在附录一中讨论这个问题）。由于 GNU Emacs 在发行时永远带着源代码，因此，如果你是一位程序员，就能给它添加自己的功能，或者自行修补你发现的程序漏洞（注 2）。你可以把副本赠送给你的朋友，永远不会有人要求你支付使用费用。你唯一不能做的事情就是不得自行增加 GNU Emacs 在使用许可方面的限制条款。也就是说，如果你向他人赠送了副本或者对它进行了改进，你不能因此而开始收取所谓的许可费。GNU Emacs 是自由的，而这种自由将永远保持下去。作为一名使用者，你的权利和责任都已经在通用公共许可证（General Public License，GPL）里描述得非常全面和清楚了，该许可证的具体内容请参考附录六。

创建自由软件基金会来发行软件程序的目的，是为了鼓励大家去共享而不是占有软件。制定通用公共许可证的目的，是为了防止出现一种自私而又常见的行为，即某

注 2： 幸运的是，GNU Emacs 中的程序漏洞是非常少的。事实上，与大多数谋求暴利的公司相比，FSF 开发的代码要好得多，而且他们还非常重视程序漏洞方面的报告。不过，如果你真的发现了一个程序漏洞却又等不及让他们去修补，你完全可以自己动手。

个公司以公开软件代码为基础，在做了若干改进和纠错之后就宣称拥有改进版本的版权而谋取利益。一旦有公司这样做了，从本质上讲，那个程序就将成为一种私有财产而不再属于公用范畴。正是出于对这种行为的厌恶，Stallman 才创建了自由软件基金会这一组织。正如他在 GNU 宣言里所说的：“我不会在理智的情况下签署一项保密合约，或者一份软件许可证合约……因此，为了让自己能够不在不名誉的情况下继续使用计算机，我必须做出‘把足够多的自由软件集合在一起’的决定。这样，我就能够不依赖任何不自由的软件而继续生活和工作下去。”在这份宣言中，Stallman 还把软件的共享称为“程序员之间最基本的友谊行为”。程序员编写出来的软件是自由的，因为它们可以共享并将永远是共享的，而且这种共享不应该有任何附加的限制（注 3）。FSF 软件不受限制性版权法律的约束，那是 Stallman 从根本上反对的做法。事实上，他专门造了一个词“copyleft”来描述 FSF 的可共享软件在版权方面的基本观点。

自从 GNU Emacs 问世之后，GNU 操作环境许多其他的组成部分也逐步到位，其中包括：C 和 C++ 编译器（*gcc* 和 *g++*）、功能强大的调试器（*gdb*）、词法分析器 *lex* 和语法分析器 *yacc* 的替代品（分别叫做 *flex* 和 *bison*），一个 UNIX 操作系统的 shell（*bash*，“Bourne-Again Shell”的缩写），以及许多其他程序和库。许多已经存在的软件工具，比如源代码控制系统 RCS 等，也纳入了 FSF 的 copyleft 版权规定中。FSF 基金会还发行一个 Linux 版本（即 Debian Linux）。有了 Linux 和 GNU 软件工具，拥有一个能够完全体现 FSF 价值观的完备的操作环境就成为可能。

学习 Emacs 的方法

就像 Emacs 有许多版本一样，它的用户群也各不相同。这本书的写作目的是帮助大家（不论是有经验的计算机用户还是新手）尽快进入 Emacs 的大门。开篇的头两章内容向大家介绍应该了解的基本概念，其余各章都是建立在这些基本概念之上的。在学习完前两章之后，大家不必一定按顺序学习其余各章的内容，你可以直接跳到自己感兴趣的的主题上去。此外，这本书尽量做到深入浅出，你既可以仔细研读每一个细节，也可以快速查阅有关的命令清单和应用示例。

注 3：Emacs 等 FSF 程序经常会随商业化系统一起发行。但即使在这样的情况下，通用公共许可证也能保证你自由传播 FSF 程序的权利不受任何制约。当然，这个许可证对与 GNU 软件工具一起发行的其他专有软件是没有效力的。

你可以参考下面列出的阅读顺序进行学习：

如果	请阅读
你是一名系统管理员用户	前言、第一~三章、第十六章
你是一名非专业用户	前言、第一~三章、第十六章
你是一位程序员	前言、第一~五章、第十~十二章
你是一位作家或专业人士	前言、第一~四章、第八~九章、第十六章
你想对 Emacs 进行定制	第十一章，也许还要再学习第十三章
你想在 Emacs 里使用电子邮件	第六章
你想在 Emacs 里使用 UNIX 命令	第五章
你想从 Emacs 里访问因特网	第六~七章和第九章

上表给出的阅读顺序仅供大家参考。Emacs 是一个庞大而又功能丰富的编辑器。我们已经对它进行了分解以方便大家的学习和掌握，所以，你完全不必因它庞大和丰富的功能而丧失信心。学习 Emacs 的最佳办法是蚂蚁啃骨头：先学会最基本的编辑功能，其他功能可以等到你对它们有了兴趣、或者等到你想做什么事却又不知该如何在 Emacs 里做的时候再去进一步学习。Emacs 很可能已经具备你需要的功能，即使没有，你也可以通过编写 LISP 函数把它添加到 Emacs 里去（具体细节请参考第十三章）。GNU Emacs 的在线帮助系统是一个快速学习掌握新功能的好地方。我们在第一章里对在线帮助的具体使用方法做了介绍，在第十六章里又对它做了更深入的讨论。

下面列出了一些大家可能打算在闲暇时学习的功能：

- 单词简写模式（第三章）
- 如何使用宏编辑命令（第十章）
- 如何把键盘上的功能键映射为 Emacs 命令（第十一章）
- 如何发出（和编辑）shell 命令（第五章）
- 如何使用多个窗口（第四章）
- 如何在图形模式（picture mode）里绘制简单的图形（第八章）
- 如何访问因特网（第七章）



- 如何发送电子邮件和阅读 Usenet 新闻（第六章）

最后，如果你坚持要从头到尾通读本书，那么请参阅下面对各章内容的简单介绍：

第一章，Emacs 的基本概念：介绍怎样启动 Emacs 和怎样对文件进行操作的方法。这一章还对在线帮助系统做了一个简单的介绍。

第二章，文件编辑：介绍 Emacs 的编辑操作，包括光标移动命令、文本的复制和粘贴命令、撤销修改命令等。这一章还介绍了几种比较初级的定制方法，这将使 Emacs 按照你设定的方式去完成工作。

第三章，查找和替换操作：介绍了更多的编辑功能，包括查找和替换、单词的简写模式、拼写检查等功能。

第四章，使用编辑缓冲区和窗口：介绍多个编辑缓冲区、Emacs 窗口和 X 窗口系统的使用方法。这一章还介绍了在文件里插入书签以便日后检索定位的有关操作。

第五章，Emacs 工作环境：介绍能够在 Emacs 中的 shell 提示符下进行的各种操作。比如，发出 shell 命令，对文件和目录进行操作，或者使用一些基本的时间管理工具等。

第六章，电子邮件和 Usenet 新闻：介绍用 Emacs 发送、阅读和管理电子邮件的方法。Gnus 新闻阅读器使你能够在 Emacs 环境中完成对 Usenet 新闻组的访问。

第七章，Emacs 的因特网工具箱：介绍利用 Emacs 编辑器远程访问其他计算机、用其 FTP 功能检索文件及浏览万维网的方法。

第八章，简单的文本排版和特效编辑：介绍 Emacs 中基本的文本排版操作（例如段落缩进和居中等）和某些使用较少的专业化编辑功能（如图形模式和大纲模式等）。

第九章，用 Emacs 设置排版标记：介绍 Emacs 对 troff（及其相关软件）、TeX、L^AT_EX 和 HTML 等标记语言的支持功能。

第十章，Emacs 中的宏：介绍利用宏编辑命令简化重复性工作的方法。

第十一章，对 Emacs 进行定制：介绍根据个人喜好对 Emacs 进行定制的方法：定制屏幕画面、定制键盘命令和编辑环境、加载 Emacs 扩展包以实现特殊功能等。

第十二章，程序员的 Emacs：介绍 Emacs 在程序设计环境方面的有关功能，包括对 C、LISP、FORTRAN 和其他一些程序设计语言的编辑支持，还介绍了对编译器和 UNIX 操作系统的 make 工具的接口。

第十三章，Emacs LISP 程序设计：介绍 Emacs LISP 的基本概念，这是一种能够对 Emacs 做进一步定制的程序设计语言。

第十四章，Emacs 编辑器和 X 窗口系统：介绍 Emacs 与 X 窗口系统的接口。如果你使用的是一个图形工作站，这个接口将使你能够通过鼠标和弹出菜单来进行操作。

第十五章，Emacs 下的版本控制：介绍对文件版本进行控制的 VC 模式。如果你维护的某些程序或文档需要附带一个修订方面的历史记录，Emacs 的版本控制功能将大大简化这类操作。

第十六章，在线帮助：介绍 Emacs 丰富而又易于使用的在线帮助功能。

附录一，如何获得 Emacs 软件：介绍获得 GNU Emacs 和其他几种 Emacs 版本的方法。

附录二，解除他人对 Emacs 的定制设置：告诉大家如何解除他人对自己的 Emacs 进行的定制设置，使它能够按这本书里所描述的那样运行和工作。

附录三，Emacs 变量：列出了许多重要的 Emacs 变量，包括本书涉及到的全部变量。

附录四，Emacs LISP 扩展包：列出了 Emacs 自带的几个最有用的 LISP 开发包。

附录五，软件漏洞及其修补：介绍怎样（以及何时）提交在 Emacs 中发现的程序漏洞。

附录六，Emacs 的版权文件：给出通用公共许可证的完整内容，GNU Emacs 就是在这些规则下发行的。

附录七，请支持自由软件基金会：为了更多地推出高质量的软件，自由软件基金会在不懈地奋斗着，而你也可以为此尽一份力量。请支持他们的工作。

附录八，Emacs 编辑命令速查表：对本书介绍的各种 Emacs 重要命令的汇总。

词汇表：对 Emacs 术语的解释。

GNU Emacs 速查卡：这张可以撕下来的卡片对附录八中列出的 Emacs 命令做了进一步的筛选，列在这张卡片里的都是最基本的 Emacs 编辑命令。

本书没有讨论的 Emacs 功能

GNU Emacs 是一个功能丰富而且强大的编辑器。限于篇幅，我们在这本书里对它的介绍和讨论并不完整。作为一本针对初学者的教材，Emacs 的许多功能都没有包括进来，而新的功能还会不断地添加进来。下面是一些没有包括在这本书里的功能：

- 各种兼容模式（compatibility modes）：GNU Emacs 为 UNIX 操作系统下的 vi 编辑器和 VAX/VMS 操作系统下的 EDT 编辑器分别提供相应的兼容模式。本书没有讨论这些模式。就我们的经验而言，这些兼容模式只是对真实情况的拙劣仿真。如果读者确实想使用 vi 或 EDT，尽管去尝试好了。我们认为既然大家学习的是 Emacs，那最好还是用 Emacs，而不是把它假装成其他的东西。
- 多种程序设计语言模式：我们在这本书里讨论了 C++、LISP 和 FORTRAN 语言的编辑模式。其实 Emacs 里还有许多针对其他程序设计语言的模式；有些工作做得还相当不错，而且新的模式还会定期地添加到其中去。可惜我们没有办法做到面面俱到。
- 高级 LISP 程序设计：GNU Emacs 内含一个完备的 LISP 解释器。这本书只简要地介绍了 Emacs LISP 中那些最基本的东西；第十三章虽然足够让你入门，但也只能算是涉及到皮毛。自由软件基金会出版了一本完整的《Emacs LISP 参考手册》，O'Reilly & Associates 出版公司很快也将出版一本这方面的书，其作者是本书的一位技术顾问，Bob Glickstein。
- 移植、调试和安装：这本书没有涉及把 GNU Emacs 移植到其他系统上时可能会遇到的问题。我们相信人们需要一本对此问题进行讨论的书籍，可惜的是它超出了本书的讨论范围。如果你认为这样一本书会对自己有所帮助，请告诉我们，我们将考虑编写这方面的书。
- 游戏和娱乐：GNU Emacs 附带很多游戏和娱乐项目。比如，它能够随机抽取《Zippy the Pinhead》（《死脑筋 Zippy》）一书中的精彩语句，作为著名搞笑心理医生“Eliza”的诊断语。（你可以在闲暇时输入“**ESC x psychoanalyze-**”

pinhead RETURN”命令，看看会发什么有趣的事情。) 不过，这些“非核心”的东西只能放到其他书里介绍了。

本书的编写体例

下面介绍这本书所使用的编写体例。

Emacs 命令

Emacs 命令是由一个修饰符，比如 CTRL (CONTROL) 或 ESC (ESCAPE)，后面再跟一或两个字符组成的。这本书里的命令把 CTRL 简写为大写字母 “C”，比如：

C-g 按住 **CTRL** 键并按下 **g** 键。

为了输入一条命令，你可能还需要按下回车键，如下所示：

RETURN 按下回车键（这个键在你的键盘上可能为 **ENTER**）。

除 **CTRL** 键外，大多数 Emacs 手册还会提到一个 **META** 键。因为大部分键盘上都没有 **META** 键，所以本书将用 **ESC** 键来代替 **META** 键，如下所示：

ESC x 按下 **ESC** 键，放开它（注 4），再按下 **x** 键。

有的读者的键盘上可能真的有 **META** 键。比如，在 Sun 工作站上，紧靠空格键左右两端的就是 **META** 键。在许多其他型号的键盘上，**ALT** 键的功能也相当于 **META** 键（注 5）。如果你的键盘上有 **META** 键，那它的用法就像刚才介绍的 **CTRL** 键一样，即按住 **META** 键再按下所需要的按键，比如 **g**。因为可以在按住 **META** 键的同时多次按下其他的按键，所以 **META** 键有一定的优越性；而 **ESC** 键在需要多次

注 4： 我们在此强调放开 **ESC** 键的原因是，在 Emacs 19 的某些版本里，两次按下它 (**ESC** **ESC**) 或者因按住它超过大约一秒钟，而使它产生自动重复会导致一条错误信息。如果你遇到了这个问题，请参考第二章“重新绑定键盘按键”小节里给出的解决方案。

注 5： 如果你是以远程方式来使用 Emacs 的，按下 **ALT** 键可能会切断连接。虽然这种情况很少见，但我们认为还是应该让大家知道有这种可能性。

按下其他按键的时候就不那么方便了。一般来说，如果你的键盘上有一个**META**键，那你肯定会更喜欢用它而不是**ESC**键。

有几个鼠标命令（仅用在X窗口系统里，参见第十四章中的讨论）需要**SHIFT**键作为修饰符，通常还要与**CTRL**键联合使用。我们把这种情况简写为：

S-right 按住**SHIFT**键的同时按下鼠标右键。

C-S-right 按住**SHIFT**键和**CTRL**键的同时按下鼠标右键。

全部Emacs命令都有一个完整的名字，即使是最简单的也不例外。比如，使用**forward-word**命令相当于按下“**ESC f**”组合键；而使用**forward-char**命令相当于按下“**C-f**”组合键。有些命令只有全名而没有对应的按键。

在介绍一条命令的时候，我们将给出它的全名和组合键（如果有的话），按下那个组合键就等于发出了对应的命令。我们假设你使用的是默认的按键绑定（即按键动作与Emacs命令的默认映射关系）。

表格

如果你想快速查找一组命令，请在有关章节里查看相应的命令汇总表。这些表格的格式如下所示：

键盘操作	命令名称	动作
C-n	next-line	移动到下一行
C-x C-f <i>Files→Open File</i>	find-file	打开一个指定的文件
(无)	yow	把minibuffer缓冲区里保存的游戏人物 Pinhead的笑话打印出来

表中第一列是命令的默认按键绑定，第二列是命令的全名，第三列是命令的功能。比如，按下“**C-n**”（即执行**next-line**命令）将把光标移动到文件的下一行。有些命令，比如“**C-x C-f**”，还可以通过菜单来执行。如果有一个与命令对应的菜单选项，我们就用斜体字把它列在该命令的组合键的下面。比如，想执行**find-file**命令，则既可以在键盘上按下“**C-x C-f**”，也可以从“**Files**”菜单里选择“**Open File**”选

项。有时候，键盘操作栏里给出的是“(无)”，这并不代表你无法使用这个命令，它的意思是这个命令没有对应的键盘操作。要想使用没有对应的键盘操作命令，必须先按下“**ESC x**”，然后输入命令的全名，再按下**RETURN**键（有时你可以输入“**ESC x yow RETURN**”试试）。

示例

在本书中，经常会看到下面这样的情况：先是需要进行键盘操作，然后是一个给出操作结果的 Emacs 屏幕画面。如下所示：

输入：**emacs myfile**



在 UNIX 提示符处输入 **emacs** 和一个文件名，开始一次 Emacs 编辑任务。

用粗体字给出的“**emacs**”表示必须精确输入的内容；而用斜体字给出的“*myfile*”表示可以把它替换为任意一个文件名，用不着精确地按照书里给出的样式。Emacs 屏幕画面中的光标（即你当前正在进行编辑的字符位置）是以反显字体给出的，就像画面底部的 Emacs 模式条（mode line）那样。

在本书的后半部分，当我们讲到各种程序设计模式、定制和 LISP 程序设计的时候，反复给出雷同的 Emacs 屏幕画面没有什么实际的意义。所以到那时，我们将不再显示上面这样的画面，我们只给出一两行文本，如果有必要，我们会以反显字体给出光标的位置，就像下面这样：

```
/* This is a C comment */
```

字体体例

本书使用了以下几种字体体例：

- UNIX 命令、Emacs 键盘操作、命令的名字、菜单选项和变量都用粗体字给出。如 “**emacs**”。
- 文件名用斜体字给出，如 “*myfile*”。
- 编辑缓冲区的名字、LISP 代码、C 语言代码、Emacs 消息以及各种程序节选段落都用等宽字体给出，如 “`constant width`”。
- 将被实际值替换掉的示范性参数用斜体字给出（如果它们出现在一段程序里，将用等宽斜体字给出，如 “`constant width italic`”）。

建议与评论

本书的内容都经过测试，尽管我们尽了最大的努力，但错误和疏忽仍然是在所难免的。如果你发现有什么错误，或者是对将来的版本有什么建议，请通过下面的地址告诉我们：

美国：

O'Reilly & Associates, Inc.
101 Morris Street
Sebastopol, CA 95472

中国：

100080 北京市海淀区知春路 49 号希格玛公寓 B 座 809 室
奥莱理软件（北京）有限公司

如果你在学习 Emacs 的过程中遇到了问题，或者有针对本书的建议，请按电子邮件地址 `deb@ora.com` 发送给我们。如果你能够访问 Usenet 新闻，那还可以到 `gnu.emacs.help` 新闻组去寻求帮助。

你也可以发电子邮件。要想参加 O'Reilly 的邮件列表或者要一些目录，请给 `info@oreilly.com` 发邮件。要询问技术问题或者对本书做评论，请发邮件给 `bookquestions@oreilly.com`。

我们有一个关于本书的网站，上面有信息、勘误表和关于将来版本的一些计划。你可以访问：



<http://www.oreilly.com/catalog/gnu2>

最后，你可以在 WWW 上找到我们：

<http://www.oreilly.com>

<http://www.oreilly.com.cn>

致谢

Debra Cameron: 本书的第二版是在许多人的帮助下完成的。有许多热心的读者给我们发来了良好的建议。Emacs 本身也发生了变化，从第 18 版升级为第 19 版。但到目前为止，还没有一本书能够把 Emacs 讲得面面俱到。Emacs 远远超出了一个编辑器的概念，它本身就是一个不断被开发的世界。它就像是你刚搬进去的一套新居，你先按照自己的想法把它变成自己的家；再不断地添添改改，让它更能折射出你工作和生活的方式。

我要感谢 Bill Rosenblatt，他曾经帮我安装了 Emacs 的两个版本并让它们运行起来，一步一步地带我走过整个安装过程。在我安装 Emacs 19 的第三个版本的时候，Lar Kaufman 和 Matt Welsh 又帮我做了大量的录入工作。Linux 和自由软件基金会在软件的安装方面做了大量的工作，使我们只需花费几个小时去 FTP 和编译 Emacs 就能顺利地把它安装起来。我还要感谢 Lenny Muellner，他耐心地帮我完成了本书的校对工作。

我还要特别感谢我丈夫 Jim 对我的支持和鼓励；还有我的两个孩子——Meg 和 David——容忍我把自己的时间和精力用在这本“Emacs 书”的写作上。

Bill Rosenblatt: 我要向以下诸位表示感谢：普林斯顿大学古典艺术系的 Richard Martin 教授，是他在我的心中播下的种子使我能够把枯燥的写作当做一种艺术的享受；Intermetrics 公司，这家公司只给我分配了很少的工作，使我能够有空在工作时间里钻研 GNU Emacs；Hal Stern，是他带我进入 Emacs 的大门；Sandy Wise，他帮我完成了 X 窗口系统的有关章节；Jessica Lustig，感谢她的爱和支持；最后还要感谢研究生学院里那些睡在我同宿舍的兄弟，我整日整夜地占用着电话线，而他们对此却毫无怨言。



Eric Raymond: 首先，我要感谢“无法无天”的黑客社团，他们缔造了丰富的Emacs LISP程序设计传统，而正是这些传统把Emacs的定制工作从一种纯粹理论意义上的可能性，转变为现实生活中的软件工具。我有很多知识都是通过阅读别人的程序代码而掌握的，这些“前辈”包括：Olin Shivers、Jamie Zawinski、Kyle Jones、Barry Warsaw、Roland McGrath、Richard Stallman本人（当然得有他）和许多其他程序员。其次，我要向Catherine C. Olanich献上我的感谢和热爱，当我埋头写作由我负责的有关章节时，她在许多方面给了我支持。最后，我要感谢O'Reilly出版公司里那些热心、专业而又负责的人们。他们不仅知道如何把一本好书带给读者，还知道如何善待每位作者。摆在大家面前的这本书就是明证。

我们这三位作者必须向本书的技术审校人员表示感谢，他们在紧迫的时间压力下体现出了工作效率和聪明才智。这本书因Ellen Siever娴熟的审校而更紧凑和实用，我们在此向他表示感谢。Bob Glickstein以一位LISP专家的眼光对本书进行了认真的审核，并督促我们在遣词造句方面精益求精。

我们还要向我们的编辑Mike Loukides表示感谢，他使本书第二版的写作和出版工作能够按预定计划顺利完成。他的指导意见和采编功力对我们帮助良多。我们还要感谢他介绍Eric Raymond成为本书的第三作者，Eric Raymond对本书的部分章节做了认真的修订，并使本书新增加了一章关于版本控制问题的重要内容。Eric的才华进一步充实了这次改版。我们还要特别感谢Gigi Estabrook，她在紧迫的时间压力下接手了本书出版的工程，并协调远在各地的作者、编辑和技术审校人员圆满地完成了任务。

最后，我们要向O'Reilly & Associates出版公司负责装帧和印刷的员工们表示感谢。Mary Anne Weeks Mayo是本书的生产经理。Robine Andrau负责本书稿的编辑工作。Clairemarie Fisher O'Leary、Kismet McDonough-Chan和Sheryl Avruch对本书的印刷质量进行了把关。Seth Maislin编写了本书的索引并在Cynthia Grabke的帮助下完成了有关的出版工作。这本书在编辑方面融进了Erik Ray、Ellen Siever和Lenny Muellner等人多年的经验，在印刷方面又得到了Michael Deutsch和John Files等人的大力协助。本书的插图由Chris Reiley负责，内部设计由Nancy Priest和Mary Jane Walsh负责，Hanna Dyer设计了本书的封底，Edie Freedman设计了本书的封面。

第一章

Emacs 的基本概念

本章内容：

- Emacs 简介
- 理解文件与编辑缓冲区
- 编辑模式
- 启动 Emacs
- Emacs 的编辑画面
- Emacs 命令
- 打开一个文件
- 保存文件
- 退出 Emacs
- 获取帮助
- 小结

有些读者可能已经迫不及待地把手放在键盘上开始打字了。我们不打算制止大家这样做，翻到“启动 Emacs”那一小节开始行动吧。不过，等你稍后准备停下来的时候，最好还是从头开始学习一下本章的内容。如果你对 Emacs 所涉及到的各种基本概念有了比较清晰的理解，学习起来就更容易了。我们将在以下各个小节里对 Emacs 的基本概念进行讨论。

Emacs 简介

GNU Emacs 是目前 UNIX 世界里最为常用的文本编辑器。与 vi (UNIX 操作系统的标准编辑器) 或者其他内置在各种现代窗口系统里的编辑器相比，许多用户更喜欢使用 GNU Emacs。那么，它为什么这么流行呢？Emacs 并不是市面上最新的编辑工具，它也肯定不是最好的。但它却将是你可以找到的最有用的编辑工具。我们希望这一章能够让大家对 Emacs 的基本概念有清楚的认识，从而帮助大家有效地利用 Emacs 来完成自己的工作。本书是针对 Emacs 用户而编写的一本指南。在编写它的时侯，我们就已经把“满足尽可能多的读者群的学习需要”当做我们的写作目的之一，从需要撰写各种备忘录和报告的系统管理员及企业管理人员，到需要使用多种程序设计语言编写源代码的程序员都在我们考虑的读者范围内。

但也正是因为如此，我们势必无法把 Emacs 的一切功能都面面俱到地介绍给大家。Emacs 有许多功能和命令没有在这本书里出现。可我们并不认为这是一个问题，因为 Emacs 本身带有一个详尽的在线帮助功能，完全能够帮助大家做进一步的学习。而我们则把注意力集中在介绍 Emacs 是如何完成各种实用性工作方面。本书的前三章主要是介绍 Emacs 的基本编辑功能，在学习完这三章之后，我们将向大家介绍怎样才能把 Emacs 当做一个完备的工作环境来使用，怎样才能发送和接收电子邮件，怎样才能在不离开这个编辑器的前提下执行 UNIX 命令，怎样使用各种特殊的文本编辑模式，怎样利用 Emacs 来编辑各种特殊类型的文件（比如 troff、TeX 和各种程序设计语言的源代码文件等）等高级论题。我们将把最重要的命令和最重要的文本编辑模式介绍给大家。但大家千万不要忘记这条原则：Emacs 的确可以把许多事情都做得很好，不过人们并不是因为这一点才说它重要的。Emacs 的重要性体现在“它能把你想要做的许多事情都集成到一起来”这一点上。

那么，所谓“集成”的意思又是什么呢？下面这个简单的例子就很能说明问题。假设有人给你发来一封电子邮件，向你介绍了一条访问新打印机的特殊命令。那么，你可以先用 Emacs 来阅读这封电子邮件，然后试试这条新命令：从 Emacs 里启动一个 UNIX shell 把新命令复制过去，再直接执行它。如果新命令很好用，你可以编辑自己的 .cshrc 文件，给这条新命令创建一个假名（或者缩写）。在做这些事情的时候，你根本用不着离开 Emacs 编辑器，也根本用不着重复输入那条命令。这就是人们说 Emacs 功能强大的原因。它远不仅是一个文本编辑器，它是一个能够改变你工作方式的完备的操作环境。

再提前给大家一个忠告：许多人认为 Emacs 是一种非常难以掌握的编辑器，但我们不明白这有什么道理。Emacs 的功能确实是非常之多，但这并不意味着你都必须学会和使用这么多的功能。就任何一种文本编辑器而言，不管它是多么简单或者多么复杂，其基本功能都应该是相同的。只要你能学会一种，就完全能够学会它们当中的任何一种。虽说我们为帮助大家记住 Emacs 的各种命令而给出了一些教条化的说明（比如“C-p 代表把光标上移一行”），可我们并不认为这是必不可少的。当然，这些助记说明确实能帮助大家越过初学阶段的拦路虎，但从长远看并不会造成什么区别。学习使用编辑器其实就是学习手指的习惯性动作：学会在想把光标移动到上一行去的时候应该把手指放到什么地方。只要大家肯在 Emacs 里做练习，用不了多长时间你就能很快适应手指的这些习惯性动作了。而一旦你掌握了这些习惯性动作，就永远也不会忘记它们，就像你永远不会忘记如何骑自行车一样。在使用 Emacs 一



两天之后，我们就再也用不着去想什么“**C-p**代表把光标上移一行”之类的东西了，手指自己就知道该放到什么位置上去。到了这一阶段，你就算入门了。也正是从这一阶段起，你将能更有创造性地使用 Emacs 来进行工作。接下来就该考虑怎样才能让 Emacs 的高级功能为自己服务的事情了。由于 Emacs 里有很多扩展菜单，所以新的鼠标点击式操作界面将使 Emacs 的使用更简便。不过，即使你有鼠标，我们也建议你学习一些最常用命令的键盘操作。良好的手指习惯动作无疑会使你成为一个打字快手；而把手指从键盘移到鼠标上去肯定会降低你的打字速度、特别是在写作的时候。

学习手指习惯动作这种办法还暗示着本书一种不同的阅读方式。毫无疑问，读一遍书肯定会学到很多东西，但你每天能够形成的新习惯却不可能太多，除非它们都是些坏习惯。第二章将把大多数常用的基本编辑技巧介绍给大家。大家可能需要多读几遍，每次学习都略有侧重。比如，Emacs 有许多办法能够让你把光标向前移动：你可以一次移动一个字符、一个单词、一行文本、一句话、一个段落和一张打印页等等。这些技巧都将在第二章里进行介绍。先从前、后移动开始学起，再逐步增加更复杂的命令。类似地，Emacs 还提供了很多在文件里进行文本搜索的办法，这比我们在其他编辑器里见过的要多很多。这些搜索操作都将在第三章里讨论。你用不着一次把它们都学会，先学会几个，多做些练习，再去学习后面的章节。即使你花了好几遍的功夫才能熟练掌握本书前三章所介绍的内容，也没有人会笑话你。在培养好习惯方面多花点时间是值得的。

理解文件与编辑缓冲区

编辑器并不是对某个文件本身进行编辑。事实上，它们会先把文件的内容放到一个临时性的缓冲区里，然后再对缓冲区里的东西进行编辑。在通知编辑器保存缓冲区的内容之前，存放在磁盘上的原始文件是不会发生任何改变的。记住：虽然缓冲区看起来与文件非常相像，但它只是一个临时性的工作区域，里面可能包含的是文件的一份副本。

Emacs 的编辑缓冲区和文件一样也有名字。缓冲区的名字通常就是正在编辑的文件的名字，但也有例外的情况。有些缓冲区没有与它们关联的文件，比如说，***scratch*** 就只是一个临时性的辅助性缓冲区，它的作用有点像草稿簿；而帮助功能会把帮助信息显示在一个名为 ***Help*** 的缓冲区里，它也是一个与任何文件都没

有关联的缓冲区。不过我们此时还用不着为此操心。就眼前来说，只要记住 Emacs 会在开始编辑一个文件的时候，把该文件复制到一个缓冲区里去就行了。编辑文件的时候，修改的是缓冲区而不是文件本身；可以等到把文本编辑得比较满意时再去保存它们，而文件本身只有在你明确地选择了存盘操作时才会发生变化。如果对自己的文本编辑工作不满意，可以在退出 Emacs 时选择不保存文件，这样就不会影响到原始的文件了。

编辑模式

Emacs 有各种各样功能略有差异的编辑模式，而它灵活多能的声誉也部分来源于此。“模式”一词听起来技术味很浓，好像还挺复杂，其实它真正的含义不过是 Emacs 对当前的文本编辑工作更“敏感”而已。当你在输入长篇大论的时候，通常需要字换行（word wrap）等功能，这样你就不必在每一行的末尾按回车键了。而当你进行程序设计的时候，就必须遵守程序设计语言在语句格式方面的规定。对写作来说，Emacs 有文本模式（text mode）；对程序设计来说，Emacs 与各种程序设计语言对应的编辑模式，比如 C 语言模式（C mode）。也就是说，编辑模式将使 Emacs 成为能满足你不同工作任务要求的“专用”编辑器。

文本模式和 C 语言模式都是主模式（major mode）。一个编辑缓冲区每次只能处于一种主模式中，退出一种主模式的办法是进入另一个主模式。表 1-1 列出了部分 Emacs 的主模式、它们的作用，以及所在的有关章节。

表 1-1：Emacs 编辑器的主模式

模式	功能
基本模式（fundamental mode）	默认模式，无特殊行为
文本模式（text mode）	书写文字材料（第二章）
邮件模式（mail mode）	书写电子邮件消息（第六章）
RMAIL 模式（RMAIL mode）	阅读和组织电子邮件（第六章）
只读模式（view mode）	查看文件，但不进行编辑（第五章）
shell 模式（shell mode）	在 Emacs 里运行一个 UNIX shell（第五章）
FTP 模式（ange-ftp mode）	下载或者查看远程系统上的文件（第七章）
Telnet 模式（telnet mode）	登录到远程系统（第七章）



表 1-1: Emacs 编辑器的主模式 (续)

模式	功能
大纲模式 (outline mode)	书写大纲 (第八章)
缩进文本模式 (indented text mode)	自动缩进文本 (第八章)
图形模式 (picture mode)	绘制简单的线条图形 (第八章)
nroff 模式 (nroff mode)	按 nroff 的要求对文件进行排版 (第九章)
T _E X 模式 (T _E X mode)	按 T _E X 的要求对文件进行排版 (第九章)
L ^A T _E X 模式 (L ^A T _E X mode)	按 L ^A T _E X 的要求对文件进行排版 (第九章)
C 模式 (C mode)	书写 C 语言程序 (第十二章)
C++ 模式 (C++ mode)	书写 C++ 程序 (第十二章)
FORTRAN 模式 (FORTRAN mode)	书写 FORTRAN 程序 (第十二章)
Emacs LISP 模式 (Emacs LISP mode)	书写 Emacs LISP 函数 (第十二章)
LISP 模式 (LISP mode)	书写 LISP 程序 (第十二章)
LISP 互动模式 (LISP interaction mode)	书写和求值 LISP 表达式 (第十二章)

当编辑一个文件的时候, Emacs 会根据正在进行的编辑工作尝试进入正确的主模式。如果编辑一个以 “.c” 结尾的文件, 它会转入 C 语言模式; 如果编辑一个以 “.el” 结尾的文件, 它会转入 LISP 语言模式。有时候, Emacs 会根据文件的内容而不仅仅是文件名来做出判断。如果编辑一个按 T_EX 格式排版的文件, 它会转入 T_EX 模式。如果它判断不出应该放到哪个编辑模式里, 就会转入基本编辑模式 (fundamental mode), 也就是最普通的编辑模式。

在主模式之外还有一些副模式 (minor mode)。副模式定义的是 Emacs 某些特定的行为, 可以在某个主模式里打开或者关闭。比如, 自动换行模式 (auto-fill mode) 表示 Emacs 将对文本自动换行; 当你输入一个长句子的时候, 它会在适当的位置自动插入一个换行符。表 1-2 列出了一些副模式和它们的作用, 以及所在的有关章节。

表 1-2: Emacs 编辑器的副模式

模式	功能
自动换行模式 (auto-fill mode)	开启字换行 (word wrap) 功能 (第二章)
改写模式 (overwrite mode)	打字时替换而不是插入字符 (第二章)

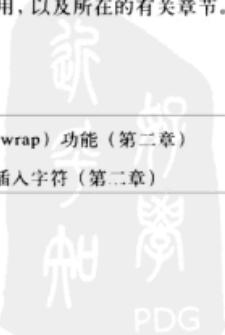


表 1-2: Emacs 编辑器的副模式（续）

模式	功能
自动保存模式 (auto-save mode)	把文件按一定周期自动保存到一个特殊的临时文件里（第二章）
行号模式 (line number mode)	在状态行上显示当前文本行的编号（第二章）
临时标记模式 (transient mark mode)	对被选取的文本区做高亮反显（第二章）
缩略语词模式 (abbrev mode)	允许使用单词的简写形式（第三章）
大纲模式 (outline mode)	书写大纲（第八章）
VC 模式 (VC mode)	在 Emacs 下使用各种版本控制系统（第十五章）

大家可能已经注意到，大纲模式既是一个主模式，又是一个副模式。这表示它既可以作为一种主模式单独使用，也可以作为一种副模式用在其他主模式里。

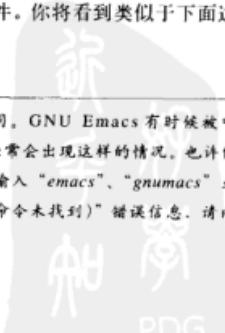
Emacs 还有其他一些没有列出的编辑模式，比如一些不常见但很有意思的程序设计语言（如 **Scheme** 等）所对应的编辑模式等。还有一些模式是 Emacs 自己使用的，比如对应于目录编辑功能的 **Dired** 模式（这个模式在第五章里介绍）。

最后，如果你擅长 LISP 程序设计，还可以自行增加新的编辑模式。Emacs 的可扩展性似乎是无穷无尽的。

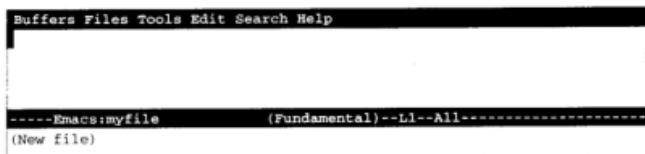
启动 Emacs

启动 Emacs 的办法很简单，输入 “**emacs**” 再加上要编辑（注 1）的文件名就行了。如果给出的文件名不存在，Emacs 将创建一个新的文件。你将看到类似于下面这样的画面：

注 1：这个命令的名字在不同的计算机上可能会有所不同。GNU Emacs 有时候被叫做 **gnumacs** 或 **gmacs**，在安装有多种编辑器的站点上经常会出现这样的情况。也许你还得修改自己 UNIX 操作系统的搜索路径。如果你在输入 “**emacs**”、“**gnumacs**” 或者 “**gmacs**” 的时候出现一条 “Command not Found (命令未找到)” 错误信息，请向你的系统管理员求助。



```
% emacs myfile
```

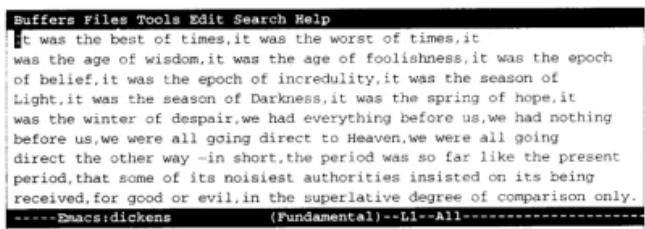


The screenshot shows the Emacs interface with a menu bar at the top labeled "Buffers Files Tools Edit Search Help". Below the menu is a large empty workspace. At the bottom, a status bar displays the text "----Emacs:myfile (Fundamental)--L1--All----" followed by "(New file)".

在 UNIX 操作系统的提示符处输入 **emacs** 和一个文件名来开始一次 Emacs 会话。

如果给出的文件已经存在，Emacs 将读入文件并把它显示在屏幕上，如下所示：

```
% emacs dickens
```

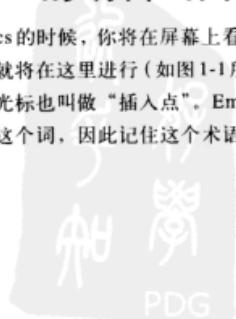


The screenshot shows the Emacs interface with a menu bar at the top labeled "Buffers Files Tools Edit Search Help". The workspace contains the text from Charles Dickens' "A Tale of Two Cities": "It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way - in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or evil, in the superlative degree of comparison only." At the bottom, a status bar displays the text "----Emacs:dickens (Fundamental)--L1--All----".

也可以省略文件名。如果只输入了“**emacs**”，屏幕上将出现你所运行的 Emacs 的版本信息、如何启动在线帮助系统和其他一些提示性信息。这些信息会在开始输入第一个字符的时候消失；而 Emacs 将把输入内容放到一个名为 *scratch* 的空缓冲区里去，这是个试做各种练习的好地方。

Emacs 的编辑画面

进入 Emacs 的时候，你将在屏幕上看到一个很大的工作区（一般是 20 多行），你的编辑工作就将在这里进行（如图 1-1 所示）。一个光标将标识出你在文件里的前后位置。这个光标也叫做“插入点”。Emacs 老手们或者 Emacs 的在线帮助系统比较习惯于使用这个词，因此记住这个术语将会很有用。



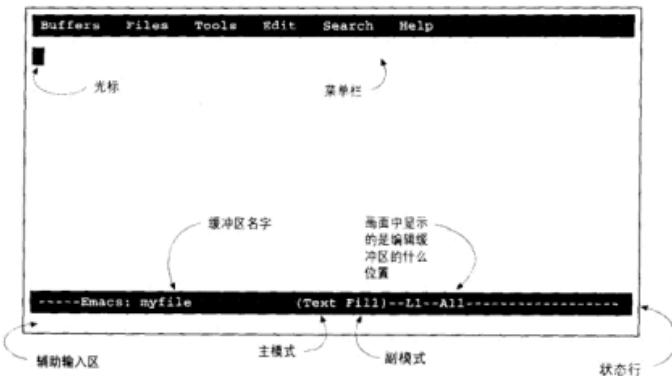


图 1-1: Emacs 的编辑画面

开始工作不需要做什么特殊的操作，在键盘上直接打字就行了。只要输入的是字母/数字字符和标点符号，Emacs 就会把它们插入到文本里去。光标指示 Emacs 将插入新字符的位置，它会随着打字动作而移动。Emacs 并没有为插入文本和输入命令分别准备编辑模式，这是它与许多编辑器（特别是 vi）的一个不同之处。现在就来输入些东西试试，你将发现 Emacs 是很容易使用的。（万一遇到了麻烦，请按 C-g 组合键。）

屏幕的顶部有一个菜单栏，上面排列着 **Buffers**（缓冲区）、**Files**（文件）、**Tools**（工具）、**Edit**（编辑）、**Search**（搜索）和 **Help**（帮助）等选项；这些选项将导向其他菜单。菜单栏的用法我们稍后再做介绍。

在屏幕的底部（倒数第 2 行），Emacs 会给出一大堆关于当前工作情况的信息。这一行叫做“状态行”。状态行靠左边的地方可能会有两个星号 (**）。这两个星号的作用是表明正在编辑的文件在上次存盘之后又被修改过。如果没有做过任何修改，那里就不会出现两个星号。接下来，Emacs 显示 “Emacs:” 和正在编辑的缓冲区的名字 (myfile)。接下来的圆括号里给出当前所处的编辑模式（编辑模式在本章前面刚刚讲过）。在它的后面，Emacs 给出在文件的前后位置：哪一行（图中的“L1”表示是第 1 行）和相对文件其余部分所处的位置。如果是文件的开头，Emacs 给出单



词“Top”；如果是文件的末尾，Emacs给出“Bot”；如果是文件中间，它会给出一个百分数（比如，“50%”表示现在看到的是文件中部的内容）；如果整个文件都显示在屏幕上，Emacs将给出单词“ALL”。

一个熟练的 Emacs 用户经常会同时打开多个缓冲区进行工作。如果是这种情况，则每个缓冲区都将有自己的一行状态行。我们现在还用操心这些事情，只要知道每个缓冲区都有一个描述其工作情况的状态行就行了。

屏幕画面最底部、状态行的下面是辅助输入区 (*minibuffer*)。Emacs 这个辅助性的输入区有许多用途，比如，Emacs 会把发出命令的执行结果回显在这里，在此输入文件名让 Emacs 去查找，搜索和替换所使用的值也要输入在这里等等。Emacs 的出错信息也将显示在这个辅助输入区里。如果你困在辅助输入区里出不来了，请再次按下 **C-g** 组合键。

X 技巧：X 窗口系统下的 Emacs 画面

X 窗口系统下的 Emacs 屏显画面与它在字符终端上的显示效果看起来稍微有些不同。每个 Emacs 窗口都有一个标题，它通常是“*emacs@systemname*”。但这并不是一成不变的：如果在一次 Emacs 会话里使用了多个 X 窗口，窗口的标题就会出现一些变化，具体情况我们将在第四章里介绍。在画面的右边是一个卷屏条，它能直观地反映出在文件中的位置，其作用与状态条上的百分比差不多。还可以利用卷屏条在文件中前后移动。图 1-2 给出了 Emacs 一个示范性的 X 窗口。

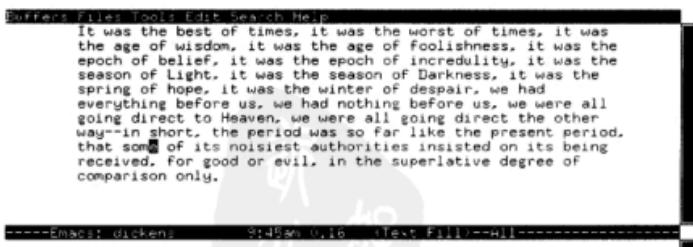


图 1-2：X 窗口系统下的 Emacs 编辑画面

Emacs 命令

我们马上就要开始学习一些 Emacs 命令了，所以，我们现在先来对它们做一个基本介绍。怎样才能发出命令呢？每个命令都有一个正式的名字，它们（如果你喜欢刨根问底儿的话）实际上是 Emacs 内部 LISP 例程的名字。这些名字一般都比较长，大多数人都不太喜欢输入完整的名称。所以，我们需要一些能够简化命令输入的办法。

Emacs 把一个命令名与一个以 **CTRL** 或 **ESC** 打头的组合键关联起来。命令与组合键之间的这种联系被称为“绑定”。在 X 窗口系统环境里，Emacs 会把一些命令与鼠标动作（单独按鼠标键或者先按下 **SHIFT** 或 **CTRL** 键再按鼠标键）和菜单里的某些选项绑定在一起。

Emacs 的创作者们已经尽量把最常用的命令，与手指最容易触到的组合键绑定在了一起。大家将会遇到的各种组合键如下所示：

- 最常用的命令（比如那些光标移动命令）都被绑定为“**C-n**”（*n* 可以是任意字符）的形式。“**C-n**”组合键的输入方法是：按住 **CTRL** 键，再按下 “*n*” 键，然后释放这两个键。
- 次常用的命令被绑定为“**ESC n**”的形式，而 *n* 可以是任意字符。“**ESC n**”的输入方法是：按下 **ESC** 键，释放它，再按下 “*n*” 键（注 2）。
- 其他常用命令被绑定为“**C-x something**”（即 **C-x** 后面再加上一些东西——可能是一个或者多个字符，也可能是另外一个控制组合）的形式。在大家将要学习到的各种命令中，文件操作类命令通常被绑定为“**C-x something**”的形式。
- 某些特殊命令被绑定为“**C-c something**”的形式。这类命令通常都与某些特殊的编辑模式有关——比如图形模式或邮件模式等。对这类命令的介绍将出现在本书比较靠后的部分。

注 2： Emacs 文档和在线帮助功能里说的都是 **META** 键，它的简化形式是大写的字母“M”。就各种实际应用目的而言，**META** 键与 **ESC** 键是完全等价的。这个按键在大多数键盘上都不存在（或者被隐藏起来了），所以本书用 **ESC** 键来代替它。如果你的键盘上确实有一个 **META** 键，那么它与 **ESC** 键还是有一点很重要的区别的。如果你准备发出一连串的 **ESC** 命令，就必须在每一个命令的前面按下 **ESC** 键；而如果你有个 **META** 键，那么你就可以按住 **META** 键来输入那一连串的命令。从这方面看，**META** 键与 **Ctrl** 键有一定的相似之处。在 Sun 工作站上，空格键左右两端的按键就是 **META** 键。在某些键盘上，**Alt** 键与 **META** 键作用相同。

- 上面这些规则依然没有顾及到所有的可能性，有些命令无法绑定为上述几种形式。这类命令的输入方法是“**ESC x long-command-name RETURN**”。这种方法其实适用于全部的命令，但组合键通常比较容易学习。

Emacs 还允许用户自己定义组合键。如果总在使用一些长格式的命令，那么这个功能就将非常方便。我们将在第十一章介绍更多关于自定义组合键的内容。

X 技巧：使用下拉菜单

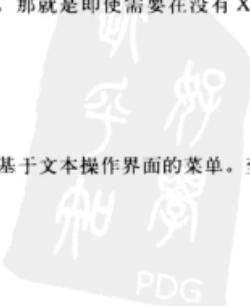
X 窗口系统的用户可以通过下拉菜单来访问常用的命令。如果想看看 **Files**（文件）菜单里都有些什么，请先把“L”形鼠标指针移到屏幕顶部的单词“**Files**”上，再按下鼠标左键（除非我们另有说明，否则都是指“一直按住它”）。屏幕上将出现**Files** 菜单，而鼠标光标的形状则变为一个箭头。这一章将介绍这个菜单里的四个选项，它们是“**Open File**（打开文件）”、“**Save Buffer**（保存缓冲区）”、“**Save Buffer As**（将缓冲区另存为）”和“**Exit Emacs**（退出 Emacs）”。如果某个菜单项不可用，就呈灰色，就像此时 **Files** 菜单里的“**Delete Frame**（删除窗格）”选项那样。

按住鼠标左键的同时在菜单里上下移动光标。在菜单里移动光标的时候，光标下的菜单项将呈高亮反显状态。如果想选中某个选项，在该选项上释放鼠标左键即可。（知道为什么要一直按住鼠标按键了吧。如果随随便便地松开鼠标按键，天知道会选中哪一个选项！）现在，既然我们只是随便试试而不是真的要进行什么操作，那么请按住鼠标按键的同时把光标移到 Emacs 画面的主窗口里，离开菜单，再松开鼠标按键。

菜单里的每个选项都可以通过键盘命令来访问。在菜单里，与各个菜单项对应的组合键就列在该菜单项的旁边。同时学习鼠标操作和键盘操作是个很不错的主意。有时，键盘操作的效率要高一些（比如在写人文稿、不想把手移到鼠标上去的时候）。学会键盘命令还有一个好处，那就是即使需要在没有 X 窗口系统的情况下使用 Emacs，你也能够应付自如。

使用基于文本的菜单

Emacs 在第 19.30 版里增加了基于文本操作界面的菜单。至少有两大理由支持增加这些菜单：



- 不必再去记忆组合键，现在可以简单地通过选择一个菜单选项来执行有关的命令。
- “鼠标手”的患者不用再移动手指去够 **CTRL** 键，这个动作已经被研究证实会加重病症。

举个例子：假设你想使用“**find-file**（查找文件）”选项（具体含义马上就要介绍）。按下 **F10** 键或“**ESC `**”（单反引号键，通常位于键盘的左上角，紧靠在数字“1”旁边）将打开主菜单（注 3）。屏幕画面显示如下：

按下：**F10**

```

Buffers Files Tools Edit Search Minibuf Help
It was the best of times,it was the worst of times,it was the age of
wisdom,it was the age of foolishness,it was the epoch of belief,it
was the epoch of incredulity,it was the season of Light,it was the
season of Darkness,it was the spring of hope,it was the winter of
despair,we had everything before us,we had nothing before us,we
were all going direct to Heaven,we were all going direct the other
way—in short,the period was so far like the present period,that
some of its noisiest authorities insisted on its being received,for
good or evil,in the superlative degree of comparison only.
-----Emacs:dickens      (Text Fill)--L1--All-----
Press PageUp Key to reach this buffer from the minibuffer.
Alternatively,you can use Up/Down keys (or your History keys)to change
the item in the minibuffer,and press RET when you are done,or press the
marked letters to pick up your choice.Type C-g or ESC ESC ESC to cancel.
In this buffer,type RET to select the completion near point.

Possible completions are:
B==>Buffers          F==>Files
T==>Tools            E==>Edit
S==>Search           H==>Help
-----Emacs:*Completions*,   (Completion List)--L1--All-
(up/down to change,PgUp to menu):B==>Buffers

```

Emacs 打开一个列出菜单选项的编辑缓冲区。

选择菜单选项有 3 种方法：

- 可以使用 **PgUp** 键切换到 *Completions* 缓冲区里去，然后使用方向键移动到想执行的选项上，再按下回车键。
- 可以直接按下回车键选中出现在辅助输入区里的默认选项。如果想要的是另一

注 3： 虽然功能键在大多数情况下都能正常使用，但如果是通过 Telnet 或者终端仿真软件来使用 Emacs，它们就不会起作用。因此，在这种情况下，必须使用“**ESC `**”组合键。

个选项，按动上、下方向键直到想要的选项出现在辅助输入区里，然后再按下回车键选中它。

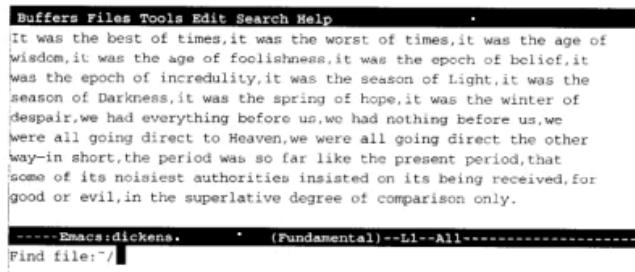
- 可以输入 *Completions* 缓冲区里列出的各选项前面的字母。比如说，按下“F”键等于选中了“Files”选项。

在选择了一个菜单项之后，该菜单的各种选项就会出现在屏显画面里。重复上述过程直到你找到你想要的选项为止。

打开一个文件

在启动 Emacs 的时候，可以给出一个文件名以打开一个文件（就像我们刚才做的那样），也可以按下“C-x C-f”组合键（与此对应的长格式命令名是“**find-file**”）。“C-x C-f”的作用是创建一个新的编辑缓冲区，它的名字与文件的名字相同。

输入：C-x C-f



```
Buffers Files Tools Edit Search Help
----- Emacs:dickens.      (Fundamental)--L1--All-----
Find file: /
```

Emacs 提示输入一个文件名。请输入一个文件名，然后按下回车键。

输入：newfile RETURN



```
Buffers Files Tools Edit Search Help
----- Emacs:newfile      (Fundamental)--L1--All-----
(New file)
```

Emacs 创建一个新的缓冲区，新文件的内容将显示在这个缓冲区里。

使用“**C-x C-f**”组合键的方法是：先按住**CTRL**键，再依次按下“x”键和“f”键，然后松开**CTRL**键。这个操作过程听起来挺复杂，不过尝试几次之后你就会很熟练了。

按下“**C-x C-f**”组合键之后，Emacs会通过辅助输入区提示输入文件名。注意：只要是Emacs需要进一步提供信息，它就会把光标自动放到辅助输入区里。完成在辅助输入区里的输入之后，需要按下回车键来确认输入了一条命令。在普通编辑命令（比如，使用**CTRL**和**ESC**键的命令）后面是用不着按下回车键的。

如果对同一个文件尝试进行两次读入操作会发生什么样的事情呢？Emacs不会创建一个新的缓冲区，它会进入该文件所在的编辑缓冲区里去。

如果你还没有在Emacs中与输入过任何东西，现在正是试着输入一些东西的好机会。你很快就会发现自己需要多学些光标移动和编辑方面的命令，这再正常不过了。你完全可以跳过本章后面的内容去开始学习第二章的内容，但我们建议你还是先读完本章的“保存文件”和“退出Emacs”小节比较好。为了方便大家今后的参考，我们在这一章的末尾给出了一个命令速查表。如果你愿意多学一些文件操作方面的知识和有帮助作用的快捷操作方法，请随我们一起去完成这一章的学习。

如果读入了错误的文件

如果错误地读入了另外一个文件（比如，因为在另外一个目录里或者出现了输入错误），找到正确文件最简便的办法是按下“**C-x C-v**”组合键（对应于“**find-alternate-file**”命令）。这个命令的意思是“读取另一个文件来代替刚才读入的那个”。按下“**C-x C-v**”组合键之后，Emacs会把当前文件的名字放到辅助输入区里去；对输入错误或不正确的文件路径（它们是读错文件最常见的两个原因）进行修改。输入正确的文件名，再按下回车键。Emacs会用新打开的文件替换编辑缓冲区里的内容。

Emacs 的名称自动补足功能

Emacs有一个很有用的功能叫做“自动完成(completion)”。如果想打开一个已经存在的文件，只需输入该文件名的头几个字母，以构成一个唯一识别的文件名即可；然后按下**TAB**键，Emacs会自动补足文件名的剩余部分。请看下面的例子，假设要打开一个已经存在的文件，它的文件名是 *dickens*。

输入: C-x C-f di

The screenshot shows the Emacs interface with the minibuffer at the bottom containing the text "Find file: /di". The menu bar above it includes "Buffers", "Files", "Tools", "Edit", "Search", "Minibuf", and "Help". The status bar at the bottom displays the message "----Emacs:newfile (Fundamental)--L1--All----".

按下“C-x C-f”之后, Emacs 提示输入文件名。输入前几个字母。

按下: TAB

The screenshot shows the Emacs interface with the minibuffer at the bottom containing the text "Find file: /dickens". The menu bar above it includes "Buffers", "Files", "Tools", "Edit", "Search", "Minibuf", and "Help". The status bar at the bottom displays the message "----Emacs:newfile (Fundamental)--L1--All----".

按下 TAB 键时, Emacs 将自动补足文件名的剩余部分。

按下: RETURN

The screenshot shows the Emacs interface. The buffer area contains the text of Charles Dickens' "A Tale of Two Cities". The minibuffer at the bottom still shows "Find file: /dickens". The menu bar above it includes "Buffers", "Files", "Tools", "Edit", "Search", and "Help". The status bar at the bottom displays the message "----Emacs:dickens (Fundamental)--L1--All----".

Emacs 读入 *dickens* 文件后, 可以开始对其进行编辑。

如果有不止一个文件的名字以“di”开头, 则 Emacs 将打开一个窗口, 把以这个字符串开头的文件都列出来。再多输入几个字符(足够唯一地确定文件即可)并再次按下 TAB 键, 就可以从中挑选出想要编辑的文件了。也可以在自动补足窗口里移动到想要的条目上, 然后按下回车键, 这样也能从文件清单里挑选出想要的文件。(X 窗口系统的用户可以利用鼠标中键来实现自动补足功能。)



自动补足功能也能用在需要输入长格式命令名的场合。Emacs这个了不起的功能能够节省不少的时间。它不仅启发了Korn shell和tcsh的开发者，还被内置在自由软件基金会的shell bash里。关于自动补足功能的进一步讨论请参考第十六章。

注意：Emacs不允许在“**C-X C-f**”操作里使用UNIX通配符（*、?等）。（但有一个简单的办法可以绕过这条限制：如果你需要对一组文件进行编辑，可以在shell提示符下使用通配符来启动Emacs。比如，如果想对以“*projectx*”开头的全体文件进行编辑，就可以在shell提示符处输入“**emacs projectx***”。）可以使用波浪符（~）作为主目录的简写形式。

插入和追加文件

如果想把一个文件插入另外一个文件，只需移动到文件的适当位置，再按下“**C-x i**”即可。（是的，我们还没有向大家介绍如何在文件里前后移动。这些操作将在下一章里讨论。）如果想追加一个文件，移动到文件的结尾（**ESC >**）（注4）、然后按下“**C-x i**”。与“**C-x C-f**”的情况相类似，Emacs会提示用户在辅助输入区里输入文件名。

Emacs 如何确定默认目录

使用任何一个需要进一步给出文件名的命令时（比如“**C-x C-f**”），Emacs会在辅助输入区里给出一个默认的目录，然后由用户来输入文件名的其余部分。那么，Emacs如何确定默认目录呢？默认目录是根据光标当时所在的编辑缓冲区确定的。按下“**C-x C-f**”组合键的时候，如果正在编辑一个位于主目录里的文件，Emacs会假定用户要编辑所登录目录里的另外一个文件；如果正在编辑的文件的文件名是*/sources/macros/troff.txt*，Emacs会把默认子目录设置为*/source/macros*。如果想找的是另外一个目录里的某个文件，就需要对Emacs提供的默认目录进行修改，或者是删除它再重新输入一个新目录。

出个难题考考大家：如果你正在编辑的那个缓冲区与文件没有联系，那么会出现什么样的情况？比如，如果你在启动Emacs时没有给出文件名，你就将在“***scratch***”

注4：“**ESC >**”的使用方法是：按下**ESC**键，释放后，再按下“>”键。

缓冲区里进行编辑，此时的默认目录又该如何设置呢？这当然是有规定的，但你其实不必为此费心——你只需按下“**C-x C-f**”组合键，看看 Emacs 在辅助输入区里给出的目录是什么，如果不合心意，把它改过来就是了。在猜测用户想设置的默认目录方面，Emacs 还是相当有门道的。

保存文件

如果想保存正在编辑的文件，请按下“**C-x C-s**”组合键或者在“**Files**”菜单里选择“**Save Buffer**”（保存缓冲区）选项。Emacs 会把文件存盘。为了让用户了解文件已被正确地保存，它会在辅助输入区 minibuffer 里显示一条“*wrote filename*”（文件已存盘）信息。如果没有对文件进行任何修改，Emacs 会在辅助输入区里显示一条“(No changes need to be saved)（没有需要保存的修改）”信息。

如果想用“**C-x C-s**”组合键把在“*scratch*”缓冲区中输入的内容保存起来，Emacs 会提示你为它指定一个文件名。给出文件名之后，Emacs 会修改状态行以反应你设置的文件名——把“*scratch*”替换为给出的新文件名。

如果按下“**C-x C-s**”组合键时发生了死机现象（再输入什么都没有反应），那么这时就需要将“**C-s**”和“**C-q**”当做流控制字符来使用了。这里，**C-s**的意思将是“停止接收输入”；而**C-q**则相当于重新启动这次会话。一个名为“**enable-flow-control**”（激活流控制）的命令（最早出现在 Emacs 19 里）可以快速解决这一问题。先输入“**ESC x enable-flow-control RETURN**”。然后，可以用“**C-v**”代替“**C-s**”，用“**C-^**”代替“**C-q**”。不过这条命令只能解决本次会话中的这个问题。永久性的解决办法请参考第二章末尾“对 Emacs 进行定制”一节里的内容。对流控制问题的进一步讨论请参考第十一章。

如果在使用“**C-s**”时遇到了麻烦（甚至即使没有遇到麻烦），你可能更愿意用**write-file**（写文件）命令（对应于“**C-x C-w**”组合键）来保存自己的文件。**write-file** 命令与**save-buffer** 命令在做法上稍微有些差异。**save-buffer** 命令假设你不想改变文件的名字；而**write-file** 命令却认为你想修改文件名，它会让你在辅助输入区 minibuffer 里输入一个新的文件名。不过，如果你直接按下回车键而不是输入一个新的文件名，**write-file** 命令会按原来名字对文件进行存盘——就像“**C-x C-s**”做的那样。

write-file 命令可以用来编辑没有修改权限的文件。先用 **find-file** 命令把想查看或者编辑的文件读入一个缓冲区，再通过 **write-file** 命令用另外一个名字（也许还要用另外—个路径）把它存为私用的版本。这个办法能够把本来无权修改的文件复制为自己—个文件；然后就可以对它进行编辑了。当然，原始文件是不会受到影响的。

退出 Emacs

如果想结束一次 Emacs 会话，可以按下“**C-x C-c**”组合键或者在 **Files** 菜单里选择“**Exit Emacs**（退出 Emacs）”选项。如果对文件进行了编辑却没有保存，Emacs 会问你是否想保存那些修改。如果回答是“**y**”，Emacs 将在对文件存盘后退出；如果回答是“**n**”，Emacs 会再次提问是否真的想放弃所做的修改并退出，这一次必须输入完整的“**yes**”或“**no**”作为回答。如果回答是“**no**”，则这次 Emacs 会话将持续下去，就像你根本没有按下过“**C-x C-c**”组合键一样。如果回答是“**yes**”，就将退出 Emacs，在这次 Emacs 会话过程中所做的修改也就都不会保留下来。如果不打算把所做的修改保留下来，那么不存盘地退出就是最合理的做法。

顺便说一句，Emacs 对所回答的是“**y**”还是“**yes**”是很挑剔的。它有时要求这样回答，有时又要求那样回答。如果它预期的回答是“**y**”，那么输入“**yes**”往往也能奏效；但反过来就未必能行。如果听见报警声并看到“Please answer yes or no”（请回答 **yes** 或 **no**），就说明你没有完整地输入这两个单词之一而它却要求这样做。在 19.29 之前的版本里，它甚至对字母的大小写都很挑剔；如果使用的是早期的版本，请根据它的要求输入小写的“**y**”或“**n**”，或者小写的“**yes**”或“**no**”作为回答。

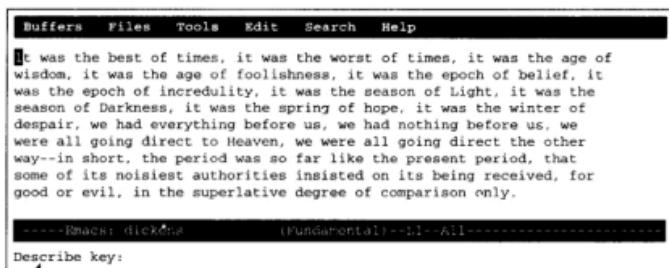
获取帮助

GNU Emacs 的在线帮助功能非常丰富，具体细节我们将在第十六章里做详细介绍。进入帮助功能的方法是敲入“**C-h**”组合键或者从 **Help**（帮助）（注 5）菜单里选择一个选项。按下“**C-h**”组合键，会出现一个选项清单。按下“**C-h t**”组合键将启动 Emacs 教程，这是一个非常好的 Emacs 入门介绍。

注 5： 在某些键盘上，用来进入帮助功能的按键是 **F1**。

如果想了解按键组合的含义,请按下对应于“**describe-key**(按键释义)”命令的“C-h k”组合键。比如,如果按下“C-h k C-x i”,Emacs 将给出一个关于 **insert-file**(插入文件)命令的用法说明——这条命令被绑定在“C-x i”组合键上。而按下“C-h f”(命令名是 **describe-function**)则是让 Emacs 对某个函数(实际就是某个命令的完整名字,比如“**find-file**”)进行解释。从理论上讲,“C-h k”和“C-h f”给出的信息是完全一样的。它们之间的区别是:“C-h k”后面要跟着一个组合键,并让 Emacs 对这个组合键的功用做出解释;而“C-h f”后面要跟着一个命令名,并要求 Emacs 对这个函数的功用做出解释。

如果想了解“C-x i”组合键的作用。那么首先要输入“C-h k”(如图 1-3 所示):



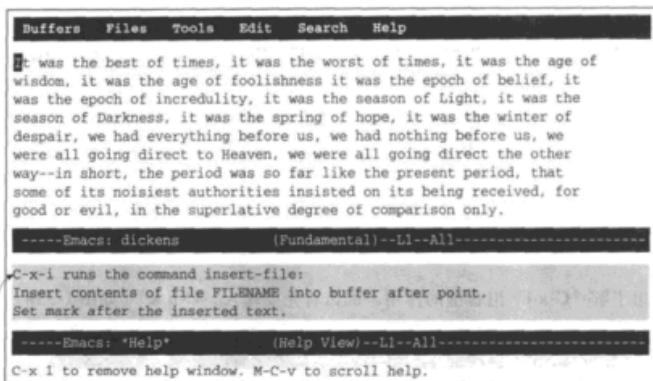
你现在准备查询关于某个键盘命令的帮助信息

图 1-3: 查询关于某个键盘命令的帮助信息

然后按下“C-x i”组合键查出关于 **insert-file** 命令的帮助信息(如图 1-4 所示)。

注意几个问题:屏显画面现在被分为两个部分,这是因为用户正在查看两个不同的编辑缓冲区。每个缓冲区都有它自己的状态条。下半部分的缓冲区是*Help*缓冲区。它包含着关于 **insert-file** 命令的帮助信息,也就是想查找的东西。Emacs 把编辑光标留在 **dickens** 缓冲区里,因为它知道一般情况下不会有人想修改*Help*缓冲区里的内容。如果想让*Help*缓冲区从屏幕上消失,需要输入“C-x !”命令(可以用“返回1号缓冲区”来帮助记忆这条命令)(注6),或者从**Files**菜单里选中“**One Window**”选项。其他帮助命令的行为都与此差不多。

注 6: 这实际上是等于跳到了第四章,但不介绍这条命令而让*Help*缓冲区总占据着你屏显画面的一半好像不太公平。



Emacs给出关于插入文件命令的帮助信息

图 1-4: Emacs 的帮助画面

大家可能还会注意到：在对命令进行解释的时候，Emacs 把光标称为“*point*（插入点）”。GNU Emacs 的各种文档里——包括在线文档和 GNU Emacs 的使用手册都使用了这种称呼。

Help 菜单

也可以通过 **Help**（帮助）菜单快速访问帮助命令。在这个菜单上，大家可以看到刚才介绍过的几个选项：“**Tutorial**”（教程，键盘命令组合是“**C-h t**”）、“**Describe Key**”（按键释义，键盘命令组合是“**C-h k**”）和“**Describe Function**”（函数释义，键盘命令组合是“**C-h f**”）等。此外，还有其他一些选项可供使用，比如获取按键绑定对应表的简单办法、访问 UNIX 命令的使用手册页（manpages）和 Emacs 常见问题答疑（frequently asked questions，简称 FAQ）文件的办法和对当前编辑模式的一个介绍等。还有一个可以用 **info** 命令来查阅 Emacs 在线文档的接口。选中“**Browse Manuals**（浏览使用手册）”就能启动 **Info** 命令。

本小节对进入 Emacs 帮助功能的几种途径进行了介绍。帮助功能还有很多种，第十六章对它们做详细的讨论。现在介绍的这些帮助功能应该足以满足大家开始使用 Emacs 的需要了。如果你还有兴趣多学点东西，可以直接跳到第十六章。

小结

这一章对启动、退出 Emacs 以及简单的文件操作进行了介绍。第二章将以此为基础继续介绍一些在 Emacs 里进行编辑工作所需要的基本命令。表 1-3 对本章涉及到的各种命令进行了汇总，**Files** 和 **Help** 菜单里的部分选项也包括在其中。

表 1-3：与文件操作有关的命令

键盘操作	命令名称	动作
C-x C-f <i>Files→Open File</i>	find-file	查找文件并在一个新缓冲区里打开它
C-x C-v	find-alternate-file	读入另外一个文件替换掉用“ C-x C-f ”读入的文件
C-x i <i>Files→Insert File</i>	insert-file	把文件插入到光标的当前位置
C-x C-s	save-buffer	保存文件
C-x C-w <i>Files→Save Buffer As</i>	write-file	把缓冲区内容写入一个文件
C-x C-c <i>Files→Exit Emacs</i>	save-buffers-kill-emacs	退出 Emacs
C-h	help-command	进入 Emacs 的在线帮助系统
C-h f <i>Help→Describe Function</i>	describe-function	给出某个给定命令名的在线帮助信息
C-h k <i>Help→Describe Key</i>	describe-key	给出某个给定键序列的帮助信息
C-h t <i>Help→Emacs Tutorial</i>	help-with-tutorial	启动 Emacs 教程
C-h i <i>Help→Browse Manuals</i>	info-goto-emacs-command-node	启动 Info 文档阅读器

疑难解答

- 在试图进入在线帮助系统时意外退出。按下了“**C-h**”组合键（或者映射到“**C-h**”的其他键），而这个组合键被绑定为退格操作的 ASCII 控制序列。在 Emacs

里，“**C-h**”是求助键，按下“**C-g**”组合键将退出帮助系统。如何把求助键从“**C-h**”组合键改为其他的绑定，比如“**C-x ?**”？具体设置办法请参考第二章中“对 Emacs 进行定制”一节。（如果只是想在文件中后退一个字符，可以通过按下“**C-b**”组合键来完成，这是下一章介绍的一个光标移动命令。）

- 无法进入在线帮助系统。“**C-h**”组合键可能被映射为键盘上的**DEL**键了。试试**F1**键管不管用。如果**F1**键不管用，请参考第二章“对 Emacs 进行定制”一节。
- 死机。你可能无意中按下了“**C-s**”组合键，而它将产生一个暂停数据流的流控制字符。有的工作站一般情况下允许使用“**C-s**”组合键，但如果是通过**rlogin**、Telnet 或者调制解调器来使用 Emacs 的，也往往会出现死机现象。要想重新恢复工作，请按下“**C-q**”组合键。如果问题还没有解决，请通过键盘输入“**ESC x enable-flow-control RETURN**”。可以用“**C-**”代替“**C-s**”，用“**C-^**”代替“**C-q**”。
- Emacs的工作情况与本书所介绍的不一样。请输入“**ESC x version RETURN**”以确认自己运行的确实是 GNU Emacs 19（而不是其他版本的 Emacs 或者 GNU Emacs 的早期版本）。如果确定版本是 GNU Emacs 19，请输入“**emacs -q**”退出 Emacs 并再次进入。第二章的“对 Emacs 进行定制”一节给出了一个比较彻底的解决方案。
- 启动 Emacs 或者查找文件时屏幕上出现乱码。问题的原因可能是正在编辑的文件是一个二进制文件或者其他某种特殊的文件。这种情况经常出现在拼错了某个文件名的时候。请输入“**C-x C-v filename RETURN**”重新查找出正确的文件。如果什么事情都不对头——根本分辨不出是否有一个缓冲区、一个状态条或者一个辅助输入区 minibuffer，那么可能是终端设置不适合 Emacs，这时需要系统管理员或者专家的帮助。按下“**C-l**”（“L”的小写字母）也可能会有所帮助。
- 看到出错信息“**This terminal is not powerful enough to run Emacs**（本终端功能不足，无法运行 Emacs）”。终端设置有问题，或者（只是可能是）使用的终端太旧，无法支持 Emacs 的运行。不过，这样的终端现在已经非常少见了。这条消息通常意味着环境变量 **TERM** 的设置不正确。如果不知道如何对环境变量 **TERM** 进行设置，请向系统管理员求助。

- 在 X 窗口系统下，状态条或者辅助输入区没有出现在屏幕上。Emacs 窗口超出了显示器屏幕。请参考第十四章，对名为 **geometry** 的 X 变量进行调整，以使 Emacs 在启动时有一个合理的窗口尺寸。作为一种临时措施，重新调整窗口尺寸也可以解决这一问题。
- 画面中一个菜单也没有。是否能够看到菜单取决于正在运行的 Emacs 版本和是否运行 X 窗口系统。在 19.30 版本之前，菜单只有在 X 窗口系统下才能工作。如果有 X 窗口系统却看不到任何菜单，可以输入 “**ESC x menu-bar-mode**” 把它们显示出来。就 19.30 版本而言，任何用户都应该能够看到菜单。请输入 “**ESC x version**” 命令来检查运行的 Emacs 版本。
- Emacs 报告 “Please enter y or n (请输入 y 或 n)”，而你已经这样做了。问题的原因可能是输入的是大写字母，而 Emacs 要求的是小写字母。请检查是否按过键盘上的 “**CAPS LOCK**” 键。



第二章

文件编辑

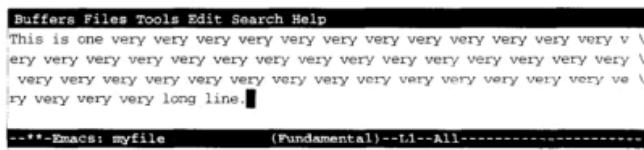
本章内容：

- 光标的移动
- 文本的删除
- 文本块及其编辑操作
- 篡落重排
- 编辑技巧和快捷键
- 命令的中止和修改的撤销
- 对 Emacs 进行定制

现在，大家已经了解了如何进入、退出 Emacs 以及如何对文件进行基本的操作。我们接下来将学习如何在文件里前后移动并进行编辑。Emacs 为我们准备了许多在文件里前后移动的方法。但正是因为完成同一件事情的方法太多，所以刚开始的时候大家可能很容易把它们弄混。不要着急，随着学习的深入，这种混乱会逐步地减少，而大家也将逐渐学会如何去欣赏 Emacs 那丰富的命令。学会的方法越多，到达文件中准备开始编辑工作的位置所需要的击键次数也就越少。

如果你打算在学习编辑命令的同时对它们加以练习——这是帮助你尽快掌握它们的好办法，可以从你手边随便什么东西开始输入一两页内容，报纸就很不错。这等于是替你自己准备了一段练习用的文本，可以用它们来练习自己在本章学到的各种编辑技巧。不要害怕出错，一口气输入下去。可以在掌握了本章所介绍的基本编辑技巧之后再回过头来改正那些错误。学习使用一种编辑器实际上是要形成某种习惯性的手指动作，而不是死板地记忆书上是怎样的说的。只有亲自动手开始打字，才能学会正确的手指习惯动作。

在打字的时候，如果到达了屏幕显示画面的右端，会有两个选择。可以按回车键转到下一行，也可以继续打下去。如果在输入一个长句子的时候没有按回车键，Emacs 会在到达显示画面右端的时候，自动在这一行的末尾加上一个反斜线 (\) 并转到下一行去。请看下面这个长句子：



当一行文本长于显示画面的宽度时，Emacs会自动在行尾加上一个反斜线。

这个反斜线字符不是文本的一部分；它们只是一种标记，提醒显示画面里的下一行其实是属于上一行的（注1）。

在每一行末尾都按一次回车键比较麻烦，而反斜线看起来又比较乱，Emacs 提供了一种比这两种做法都要好的方法——自动换行模式（auto-fill mode），这种副模式把在什么地方断行的工作交给 Emacs 去决定。Emacs 会在句子接近行尾的时候等待你输入一个空格，然后它会把下一个单词（有时候是好几个单词）转到下一行去。这种行为有时候也被叫做“字换行（word wrap）”功能，它在需要录入大段文字的时候是很有用的。出于这个理由，自动换行模式和文本模式（text mode）通常是形影不离。文本模式通常用来录入英文；但在编写程序的时候，人工添加换行符往往是人们更喜欢用的方法。

自动换行模式不会被默认地设置为on。请查看状态行。如果它上面有单词“Fill”，就说明已经在自动换行模式里了；有人（很可能是系统管理员）替用户把它设置为on。如果在状态行上没有看到这个单词，可以输入“**ESC x auto-fill-mode RETURN**”（注2），把这个编辑缓冲区的自动换行模式设置为on，但这只对这一个缓冲区有用。如果不想要自动换行模式，请再次输入“**ESC x auto-fill-mode RETURN**”。这个命令就像是一个信号灯开关，只不过它切换的是自动换行模式的开、关状态。

注1：在某些情况下（我们稍后会讲到），Emacs会在显示画面的右边界放上一个美元符号(\$)，这一行的其余部分就不显示出来了。

注 2：当我们说“输入 ‘**ESC x this-outrageously-long-string RETURN**’”的时候，别忘记 Emacs 的自动完成功能。在输入几个字符后按下 TAB 键，Emacs 通常就会自动填上单词的其余部分或者命令的其余部分。应该尽量把自己的工作转移一些给 Emacs。

用户可能（就像我们一样）希望在每次进行文件编辑的时候都自动进入自动换行模式。它的具体做法将在本章快要结束的时候进行介绍。

光标的移动

移动光标最简单的方法是点击鼠标左键（如果有鼠标可用的话）或者按动方向键。把光标向右移动一个字符位置的Emacs命令是“**C-f**”（“f”表示“forward”，向前）；而“**C-b**”将把光标向左移动一个字符位置。把光标向上移动一行，按下“**C-p**”（对应的命令名是 **previous-line**）；向下移动一行，按下“**C-n**”（对应的命令名是 **next-line**）。知道了这些字母所代表的单词，就可以很容易地记住这些命令了。图 2-1 给出了如何使用 Emacs 命令让光标做上、下、左、右方向的移动。

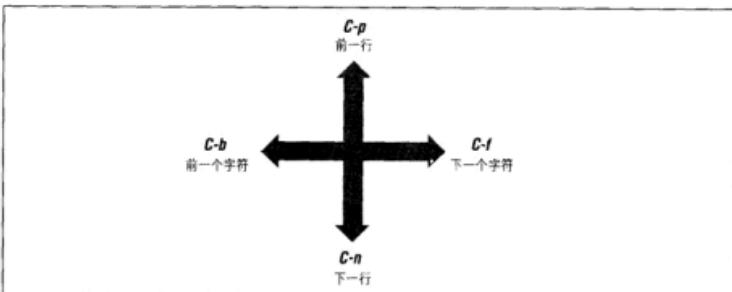


图 2-1：基本的光标移动操作

如果是在一行的末尾，“**C-f**”将把光标移到下一行的第一个字符；类似地，如果是在一行的开始，“**C-b**”将把光标移到上一行的最后一个字符。如果光标不能被移动，Emacs会鸣叫并显示出错信息“Beginning of buffer (缓冲区头)”或“End of buffer (缓冲区尾)”。这条规则有一个重要的例外：如果在缓冲区的最后一行按下“**C-n**”组合键，将把光标移动到下一行，即等于给缓冲区增加一个新行。此时（也只有在此时）的“**C-n**”不仅把光标移动到下一行，还创建了一个新行。别问我们为什么会这样。

移动光标的其他方法

现在我们来学习更高级的光标移动方法。一种常见的情况是把光标向前或者向后移过一个单词：“**ESC f**”把光标右移一个单词；“**ESC b**”把光标左移一个单词。还可以直接把光标移动到一行的开始或者结束。“**C-a**”把光标移到一行的开始（就像“*a*”是字母表的第一个字母一样），“**C-e**”把光标移到一行的结束。如果想把光标左移一个句子，按下“**ESC a**”；如果想把光标右移一个句子，按下“**ESC e**”。如果想把光标下移一个段落，按下“**ESC }**”；如果想把光标向上移一个段落，按下“**ESC {**”。如果光标处在一个句子或者一个段落的半中间，那么把光标向回移过一个句子或者段落将把光标移到这个句子或者段落的开始。

图 2-2 用维克多·雨果的小说《悲惨世界》里的几段话做例子，演示各种把光标一次移动一个以上字符的方法。

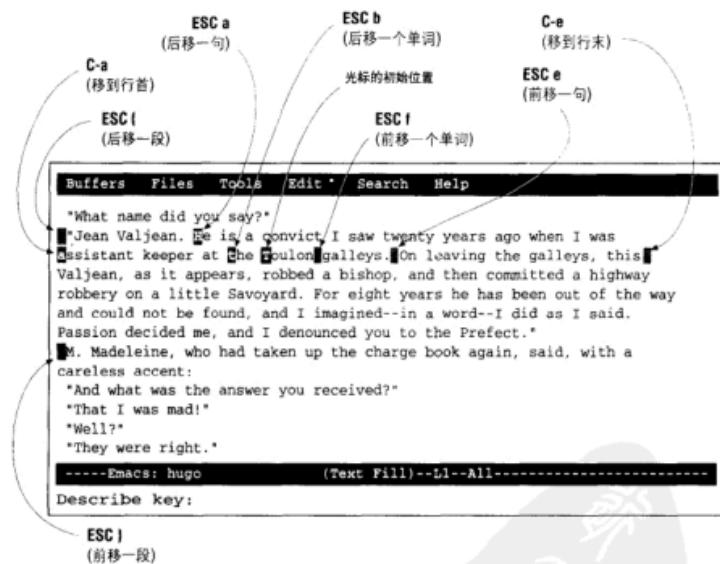


图 2-2：光标的快速大范围移动

下面是记忆这些基本编辑命令的重要提示。请注意以“**CTRL**”开头的命令与以“**ESC**”开头的命令之间的区别。以“**CTRL**”开头的命令的光标移动距离通常都要比对应的以“**ESC**”开头的命令移动距离短。比如，“**C-b**”把光标向左移动一个字符，而“**ESC b**”把光标向左移动一个单词。类似地，“**C-a**”把光标移动到文本行的开始，而“**ESC a**”把光标移动到一个句子的开始。

在以句子或者段落为单位来移动光标方面有一个限制性的前提：Emacs对句子的定义是很严格的。如果句子的结束位置不是一个文本行的结尾，那么它最后一个标点符号的后面必须有两个空格。如果只有一个空格，Emacs就无法分辨。类似地，把光标向前或者向后移过一个段落也涉及到Emacs对段落的定义。对Emacs（以及我们当中的大多数人）来说，段落通常要缩进一个制表位——至少要缩进一个空格，或者两个段落之间增加一个空白行（现代书信体）。用户可以对这些设定情况进行修改，但必须先掌握正则表达式的使用方法——我们将在第三章对正则表达式做简单的介绍，在第十三章做进一步的讨论。第十一章将涉及有关修改Emacs变量的内容。

如果文件里有分页符，可以通过敲入“**C-x]**”（命令名为**forward-page**）或“**C-x [**”（命令名为**backward-page**）组合键，来把光标移动到下一页或上一页。类似于段落和句子的光标移动方式，按页移动光标又将涉及到Emacs对页面的定义。Emacs所使用的分页符是由一个名为**page-delimiter**的变量定义的。如果文件里没有Emacs能够识别出来的分页符，Emacs就会把编辑缓冲区看做是一个非常长的页面。在这种情况下，**forward-page**命令会把光标移到缓冲区的末尾，而**backward-page**命令则会把光标移到缓冲区的开头。

文本模式里的分页符是一个进纸换页字符，它的作用是告诉打印机先进到下一个打印页（在连续纸上，这等于是让打印机走纸到下一个页面，这就是“*formfeed*”进纸这个术语的来历）再继续打印。如果想在文本模式下在文件里插入分页符，可以按下“**C-q C-l**”（小写的“L”字母）组合键。“**C-q**”叫做“引用”命令，它的作用是告诉Emacs要把“**C-l**”字符插入到文件里去，而不是把它解释为重新绘制屏显画面命令。“**C-l**”字符在文件里的样子就像是两个字符（^L），但实际上是一个。（大家用**DEL**键试着删除一个这样的字符就明白了。）

把光标一次移过一个（或者多个）屏显画面

如果想一次把屏幕上的文件上卷一页，可以使用键盘上的 **PgDn** 键或者按下“**C-v**”（命令名是 **scroll-up**）组合键。Emacs 会把文件的下一屏内容显示到屏幕上。它会把上一屏幕内容的最后几行留在屏幕顶部，好让用户掌握文件的上下文关系。类似地，按下“**ESC v**”（或者 **PgUp** 键）将把文件的前一屏内容显示在屏幕上。“**C-v**”和“**ESC v**”提供了快速翻看文件的简便方法。

如果输入的光标移动命令的目标位置在当前屏显文本内容以外，Emacs 会自动进行卷屏。比如说，如果在屏幕最后一行按下“**C-n**”组合键，Emacs 就会上卷一行。类似地，如果在屏幕最顶部按下“**C-p**”组合键，Emacs 就会下卷一行。

有时候，需要直接到达某个文件的开头或者结尾。在 Emacs 里，按下“**ESC >**”或者按下 **END** 键将移动到编辑缓冲区的末尾；而按下“**ESC <**”或者按下 **HOME** 键将移动到文件的开始。请记住：“>”指向缓冲区尾，而“<”指向缓冲区头。

另外还有两种很方便的光标移动方法。“**ESC x goto-line n RETURN**”把光标移动到文件的第 *n* 行。当然，Emacs 是从文件头开始计算行数的。类似地，“**ESC x goto-char n RETURN**”把光标移动到文件的第 *n* 个字符，从文件头开始计数。两个命令中的“*n*”都代表着一个数字。

这两个命令对程序员来说是非常有用的，这是因为许多编译器给出的出错信息都是“Syntax error on line 356（第 356 行语法错误）”这样的格式。这两个命令可以让用户迅速找到发生错误的位置。还有一些更复杂的方法可以把 Emacs 与编译器，或其他程序产生的出错报告联系起来；另外还有几个光标移动命令只能用在编辑程序代码的场合，这些内容都将在第十二章里讨论。

X 技巧：使用卷屏条

用鼠标来移动光标就更简单了。只要在屏显画面上的某个位置点击一下鼠标左键，光标就会跳到那个地方。使用卷屏条时的直观性要稍差一些。点击屏显画面右边的卷屏条可以在文件中移动。当把鼠标指针移动到卷屏条上的时候，它的形状会变成一个指向上下方向的双向箭头。此时，点击左键将使文件内容向上移动，点击右键将使文件内容向下移动。按住鼠标中键后可以上下拖动卷屏条中的滑块。把滑块拖

到卷屏条的顶部可看到缓冲区开始部分的内容；而把滑块一直向下拖动可看到缓冲区结尾部分的内容。也可以把滑块放到卷屏条的任何中间位置上；把滑块放到卷屏条的中部可到缓冲区中间部分的内容。

命令的重复执行

现在来学习几个提高操作效率的技巧。Emacs的任何命令都允许重复执行多次。第一种方法是在准备重复执行的命令前面加上“**ESC n**”，其中的“n”是准备重复执行的次数。这个命令被称为 **digit-argument** 命令。

如果准备重复执行一个命令多次，可以用“**ESC n**”指定一个很大的数字。举个例子：假设正在编辑一个大约有 1000 行的大文件。如果输入“**ESC 500 C-n**”，光标就将向下移动 500 行而到达文件的中部。在大文件里用这个命令来移动光标通常是最快的。如果在“**ESC n**”中给出的数字超出了命令所能执行的范围，它将执行尽可能多的次数之后停下来。

另外一个可以重复执行编辑的命令是“**C-u**”（命令名是 **universal-argument**）。可以像使用“**ESC n**”那样也在“**C-u**”后设定一个参数，“**ESC 5**”和“**C-u 5**”都会把随后的命令重复执行 5 次。但“**C-u**”与“**ESC n**”的不同之处在于前者即使没有参数也可以重复执行命令。如果不带参数，“**C-u**”将把随后的命令重复执行 4 次。如果按下的是“**C-u C-u**”，它将重复执行命令 16 次。也就是说，重复使用“**C-u**”可以按 4 的幂次来重复执行随后的命令：16 次、64 次、256 次，依此类推（注 3）。

重新绘制屏显画面

很多情况下需要用“**C-l**”（小写“L”字母）组合键来重新绘制屏显画面。比如，如果是在一个字符终端上而不是 X 窗口系统下来使用 Emacs 的，那么，随着时间的推移，屏显画面就可能会因为其他进程发送的各种消息而变得混乱不堪。比如，可能

注 3： 在大多数情况下，可以按我们这里的介绍来使用“**C-u**”。可它并不总是一个命令重复因子；有时候，“**C-u**”的作用会是改变某个命令的功能。大家将在本章的后面看到一个这样的例子。不过，只要正在做的事情与命令的重复执行有关，“**C-u**”就几乎总是能起到命令重复因子的作用。

会收到通知有电子邮件到达、有人想用 talk 聊天软件请你吃午饭，或者磁盘空间不足等各种各样的消息。如果有的是用调制解调器，屏幕上还会显示出许多“垃圾”字符来。Emacs 是不会被这些来来往往的字符弄糊涂的，不过它们却可能把用户给弄糊涂，因为屏幕上多出了一些不属于自己文件的文本，而光标也没有处于它应在位置上等等。如果遇到这种情况，按下“C-l”就可以解决问题：Emacs 会正确地重新绘制屏显画面。作为一般原则，只要屏幕有点乱，就可以试试“C-l”组合键。

“C-l”还有另外一个好处：它会把正在编辑的那一行放到屏显画面的中心位置。（准确地说，是放到“窗口的中心位置”，但我们还没有讲到窗口。我们将在第四章讨论它们。）当打字打到画面的底部或者顶部时，这个命令的好处就体现出来了。按下“C-l”会把用户最关心的文件内容放到屏显画面的中心位置，方便用户看到它的上下文。

表 2-1 列出了我们前面介绍的各种光标移动命令。

表 2-1：光标移动命令速查表

键盘操作	命令名称	动作
C-f	forward-char	光标前移一个字符（右）
C-b	backward-char	光标后移一个字符（左）
C-p	previous-line	光标前移一行（上）
C-n	next-line	光标后移一行（下）
ESC f	forward-word	光标前移一个单词
ESC b	backward-word	光标后移一个单词
C-a	beginning-of-line	光标移到行首
C-e	end-of-line	光标移到行尾
ESC e	forward-sentence	光标前移一个句子
ESC a	backward-sentence	光标后移一个句子
ESC }	forward-paragraph	光标前移一个段落
ESC {	backward-paragraph	光标后移一个段落
C-v	scroll-up	屏幕上卷一屏
ESC v	scroll-down	屏幕下卷一屏
C-x]	forward-page	光标前移一页
C-x [backward-page	光标后移一页

表 2-1：光标移动命令速查表（续）

键盘操作	命令名称	动作
ESC <	beginning-of-buffer	光标前移到文件头
ESC >	end-of-buffer	光标后移到文件尾
(无) ^a	goto-line	光标前进到文件的第 <i>n</i> 行
(无)	goto-char	光标前进到文件的第 <i>n</i> 个字符
C-l	recenter	重新绘制屏显画面，当前行放在画面中心处
ESC n	digit-argument	重复执行 <i>n</i> 次后续命令
C-u n	universal-argument	重复执行 <i>n</i> 次后续命令（省略 <i>n</i> 时重复 4 次）

a. 表中第一栏里的“(无)”表示如果想执行这个命令，就必须先按下“**ESC x**”，再输入该命令的全名，最后按下回车键。它们没有对应的默认组合键。

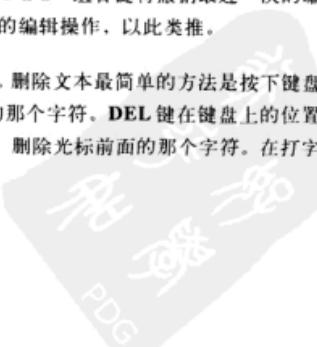
Emacs 命令与你的键盘

许多 Emacs 命令可以通过键盘上的标准按键来快速输入，比如 **PgDn**（前进到下一页）和 **Home**（跳到缓冲区的开头）等。图 2-3 给出了一个键盘布局样板和各有关按键的功能，各个用户按键可能与图中给出的位置稍有差别。不过，如果键盘上有同名或者名称相似的按键，它们就应该能用。“应该能用”的意思是这些按键在少数情况下会不能用，比如你在一台远程机器上使用 Emacs 的时候。建议大家记住这些标准的 Emacs 命令，因为它们在任何一种键盘上都能用；而且，一旦掌握了它们的用法，用手指够起来往往也比较容易。

文本的删除

在用文本删除命令做练习之前，我们想先把撤销操作（undo）命令介绍给大家，本章后面将对这个命令做详细的讨论。按下“**C-x u**”组合键将撤销最近一次的编辑操作；再次按下这个组合键会撤销再上一次的编辑操作，以此类推。

Emacs 为我们准备了很多种删除文本的方法。删除文本最简单的方法是按下键盘上的 **DEL** 键，它的作用是删除紧靠光标左侧的那个字符。**DEL** 键在键盘上的位置请参考图 2-3。**DEL** 键的功能是最容易定义的：删除光标前面的那个字符。在打字的



时候，如果想删除刚刚输入的那个字符，应该按下键盘上的哪个按键呢？在 Emacs 里，应该按下 **DEL** 键。

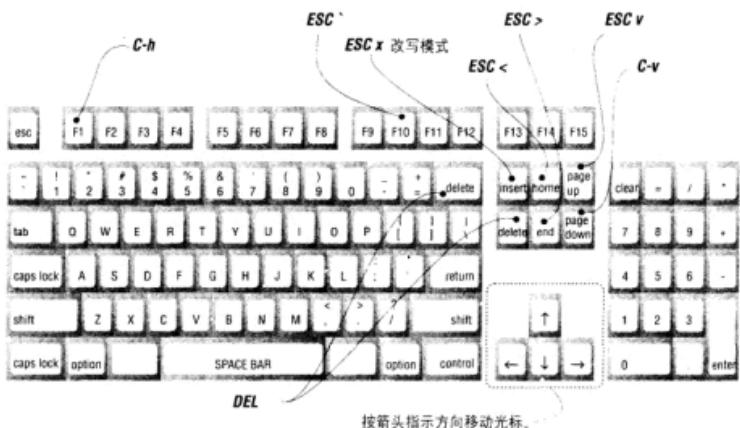


图 2-3: Emacs 命令与键盘

Emacs 提供了许多其他的删除命令，比如用来删除光标所在位置处字符的“**C-d**”命令（命令名是 **delete-character**）等——虽然命令很多，不过它们都是有存在价值的。删除下一个单词的命令是“**ESC d**”（命令名是 **kill-word**）。再次提醒大家注意 **ESC** 是如何扩大命令的作用范围的：“**C-d**”对字符进行操作，而“**ESC d**”对单词进行操作。

Emacs 提供了向前、向后删除单词、句子和段落的编辑命令。可以根据这些命令的名字猜出它们在光标处于单词、句子或者段落之间时会有什么样的操作结果。可是，如果光标位于一个单词、句子或者段落的中间，它们的操作结果就有些出人意料了：它们会删除当前那个单词、句子或者段落的一部分，具体是向前删还是向后删要看命令本身是向前操作还是向后操作。请看下面这些操作实例，“**ESC d**”会根据光标的不同位置采取不同的行动：

如果光标位置在：	“ESC d” 的操作结果：
It was the w <small>or</small> t of times	It was the w <small>o</small> f times
It was the <small>wor</small> t of times	It was the <small>w</small> of times
It was the wors <small>t</small> of times	It was the wors <small>t</small> of times

类似地，如果在某个单词的中间让 Emacs 去删除前一个单词（**ESC DEL**，命令名是**backward-kill-word**），它将从光标位置开始删除一直到那个单词的第一个字母。

如果想全部或者部分地删除一行文本，可以使用“**C-k**”（命令名是**kill-line**）命令。这个命令将把从光标位置到行尾之间的文本全删除。在空白行上按下“**C-k**”组合键将删除这一行本身。因此，一般要用两个“**C-k**”才能删除一行：一个用来删除文本，另一个用来删除剩下来的空白行。如果想删除的是从行首到光标位置之间的东西，可以试试复杂的组合命令“**ESC - C-k**”（即先按**ESC**键，后面跟一个短划线，然后是“**C-k**”）。

恢复已删除的文本

大家可能已经注意到 Emacs 中某些删除命令的名称里有个单词“*kill*”，如 **kill-region**, **kill-word** 等。在 Emacs 里，删除并不意味着永远消失。事实上，在许多情况下，已经被删掉的文本都可以再恢复。被删掉的文本并没有消失，它们被隐藏在一个被称为“删除环（kill ring）”的地方。删除环与大家熟悉的剪贴板有着异曲同工之处。要想恢复已经删除的东西，可以按下“**C-y**”（命令名是**yank**）组合键（注4），或者按下**SHIFT-INSERT**组合键，或者从**Edit**（编辑）菜单里选择“**Paste Most Recent**”选项。更方便的是，如果连续删除了多行文本，Emacs 会把它们收集起来做为一个整体放到删除环里去——这时用一个“**C-y**”命令就可以把它们都恢复回来。在下面的例子里，我们先用 4 个“**C-k**”删除小说《双城记》选段的前两行（记住：第一个“**C-k**”用来删除文本，第二个“**C-k**”用来删除剩下来的空白行），然后用一个“**C-y**”就把它们都恢复回来了。

注 4： 正在学习 Emacs 的 vi 用户请注意：vi 里也有一个术语叫做“yank”，但这两者的含义几乎正好相反，千万不要弄混了。

初始状态：

```
Buffers Files Tools Edit Search Help
It was the best of times, it was the worst of times, it was the age of
wisdom, it was the age of foolishness, it was the epoch of belief, it
was the epoch of incredulity, it was the season of Light, it was the
season of Darkness, it was the spring of hope, it was the winter of
despair, we had everything before us, we had nothing before us, we
were all going direct to Heaven, we were all going direct the other
way—in short, the period was so far like the present period, that
some of its noisiest authorities insisted on its being received, for
good or evil, in the superlative degree of comparison only.
-----Emacs:dickens (Text Fill)--L1--All-----
```

光标位于左上角。

按下：C-k C-k C-k C-k

```
Buffers Files Tools Edit Search Help
-----as the epoch of incredulity, it was the season of Light, it was the
season of Darkness, it was the spring of hope, it was the winter of
despair, we had everything before us, we had nothing before us, we
were all going direct to Heaven, we were all going direct the other
way—in short, the period was so far like the present period, that
some of its noisiest authorities insisted on its being received, for
good or evil, in the superlative degree of comparison only.
-----Emacs:dickens (Text Fill)--L1--All-----
```

用“C-k”删除了前两行。

按下：C-y

```
Buffers Files Tools Edit Search Help
It was the best of times, it was the worst of times, it was the age of
wisdom, it was the age of foolishness, it was the epoch of belief, it
was the epoch of incredulity, it was the season of Light, it was the
season of Darkness, it was the spring of hope, it was the winter of
despair, we had everything before us, we had nothing before us, we
were all going direct to Heaven, we were all going direct the other
way—in short, the period was so far like the present period, that
some of its noisiest authorities insisted on its being received, for
good or evil, in the superlative degree of comparison only.
-----Emacs:dickens (Text Fill)--L3--All-----
```

只用一个操作就把所有东西都恢复回来了。

到底有哪些东西会被放到删除环里去呢？进入删除环的内容包括用“C-k”命令删除的所有东西、用“C-w”命令删除的所有东西和用“ESC w”命令复制的所有东西（后两个命令马上就会学到）。用“ESC d”和“ESC DEL”命令及其变体删除

的单词、句子和段落也将进入删除环。此外，用“C-u”加DEL或“C-d”删除的文本也将进入删除环。不会被Emacs保存到删除环里去的内容大概只有用DEL或“C-d”删除的单个字符了。（如有必要，可以用undo撤销操作命令“C-x u”来恢复这类删除操作。）

Emacs对删除环中文本的处理也很聪明：它总是能够把一组删除操作所删除下来的东西，正确地拼凑成一个大段的文本。比如，可以先按下几个“**ESC d**”，再来几个“**ESC DEL**”，中间还可以再夹杂几个“**C-k**”。可当按下“**C-y**”组合键的时候，Emacs会把刚才删除的文本按正确的顺序都恢复回来。

但必须注意这样一个问题：如果给出了一个不属于“kill”类命令的命令，Emacs就将停止拼凑被删除文本的工作。比如，假设先按了一次“**C-k**”组合键，然后用“**C-d**”删除了一个字符，最后又按了一次“**C-k**”组合键。这就不能算是一组连续的删除操作了；实际进行的是两次互不相干的删除操作。稍后再向大家介绍如何把前面的删除操作恢复回来。

表2-2对文本的删除和恢复命令进行了汇总，Edit菜单里的有关选项也包括在其中。

表2-2：文本删除命令速查表

键盘操作	命令名称	动作
C-d	delete-char	删除光标位置上的字符
DEL	delete-backward-char	删除光标前面的字符
ESC d	kill-word	删除光标后面的单词
ESC DEL	backward-kill-word	删除光标前面的单词
C-k	kill-line	从光标位置删除到行尾
ESC k	kill-sentence	删除光标后面的句子
C-x DEL	backward-kill-sentence	删除光标前面的句子
C-y 或 SHIFT-INSERT <i>Edit→Paste Most Recent</i>	yank	恢复被删除的文本
C-w 或 SHIFT-DELETE <i>Edit→Cut</i>	kill-region	删除文本块
(无)	kill-paragraph	删除光标后面的段落
(无)	backward-kill-paragraph	删除光标前面的段落

文本块及其编辑操作

如果想删除一个短语该怎么办？段落的一部分呢？几个段落呢？可以组合使用各种删除命令来删除想去掉的东西，不过Emacs已经准备了一个更简单的方法：用标记文本的方法把打算删除的东西定义为一个区域。这个被标记出来的区域就叫做“文本块 (region)”。

为了对文本块进行定义，要用到一个称为“文本标记 (mark)”的辅助指针。某些 Emacs 版本的文本标记在屏幕上是可见的，可惜 GNU Emacs 的文本标记是不可见的。

把光标移到文本块的一端，按“C-@”或“C-SPACE”组合键设置一个文本标记；然后移动光标，文本标记和光标当前位置之间的文本就构成了一个文本块。(在提到光标的时候，人们也经常使用“插入点 (point)”这个词。不过，光标和插入点却有一个细小但很重要的区别。光标显示在一个字符上；而 Emacs 中的插入点实际是在光标位置处字符与其前一个字符之间的夹缝里。两者的区别确实很小，但在标记文本的时候就值得细加注意了。) 图 2-4 给出了插入点、文本标记和文本块之间的关系。

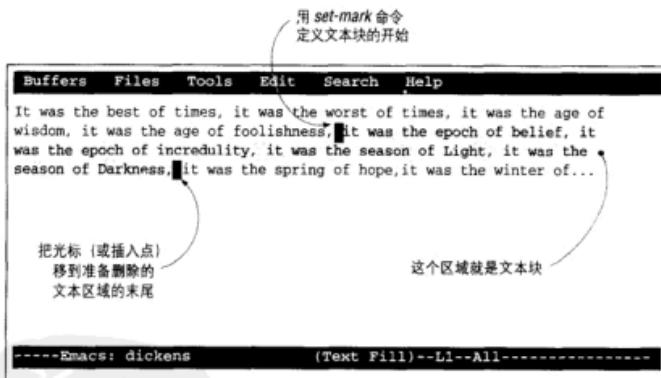


图 2-4：插入点、文本标记和文本块

我们来做一个标记文本块的示范。在下面的例子中，打算删除的是“it was the worst of times.”这句话。首先，把光标移到这段话的开始，设置文本标记；然后，把光标移动到这段话的结尾；最后执行删除操作。给被选取的文本块加上阴影以示醒目（但书里的阴影实际上是不会反映在屏幕上的，除非使用了X）。

把光标移到“it”的首字母，按下“C-@”组合键。

```

Buffers Files Tools Edit Search Help
It was the best of times, it was the worst of times, it was the age of
wisdom, it was the age of foolishness, it was the epoch of belief, it
was the epoch of incredulity, it was the season of Light, it was the
season of Darkness, it was the spring of hope, it was the winter of
despair, we had everything before us, we had nothing before us, we
were all going direct to Heaven, we were all going direct the other
way—in short, the period was so far like the present period, that
some of its noisiest authorities insisted on its being received, for
good or evil, in the superlative degree of comparison only.

-----Emacs:dickens      (Text Fill)--L1--All-----
Mark set

```

设置文本标记；辅助输入区里出现“Mark set”字样。

把光标移到“it was the age of wisdom.”中“it”的字母“i”处。因为插入点的精确位置在字母“i”的前面，所以它正是我们需要的位置。

```

Buffers Files Tools Edit Search Help
It was the best of times, it was the worst of times, it
was the age of wisdom, it was the age of foolishness, it was the epoch
of belief, it was the epoch of incredulity, it was the season of
Light, it was the season of Darkness, it was the spring of hope, it
was the winter of despair, we had everything before us, we had nothing
before us, we were all going direct to Heaven, we were all going
direct the other way—in short, the period was so far like the present
period, that some of its noisiest authorities insisted on its being
received, for good or evil, in the superlative degree of comparison
only.

-----Emacs:dickens      (Text Fill)--L1--All-----
Mark set

```

把光标移到准备标记的文本块的末尾。

现在，文本块就标记好了。在执行删除命令之前最好先检查一下，看文本块标记得是否正确。按下“C-x C-x”（命令名是 **exchange-point-and-mark**）组合键，它的作用是互换插入点和文本标记的位置。如果光标移到了应该有文本标记的地方，就说明文本块已经被正确地标记出来了。因为GNU Emacs中的文本标记是不可见的，

所以最好养成一个好习惯：在对文本块做删除操作之前先用“**C-x C-x**”检查一下它的位置。即使是一些已经使用 Emacs 很多年的人也会忘记检查文本标记，随手一删，不知道会删除哪些东西。（这时候就用得着撤销操作命令“**C-x u**”了。）

要想删除文本块，可以按下“**C-w**”（命令名是**kill-region**）组合键，或者按下**SHIFT-DELETE** 组合键，或者从**Edit** 菜单里选择执行 **Cut**（剪切）操作。

按下：**C-w**

```
Buffers Files Tools Edit Search Help
It was the best of times, it was the age of
wisdom, it was the age of foolishness, it was the epoch of belief, it
was the epoch of incredulity, it was the season of Light, it was the
season of Darkness, it was the spring of hope, it was the winter of
despair. we had everything before us, we had nothing before us, we
were all going direct to Heaven, we were all going direct the other
way—in short, the period was so far like the present period, that
some of its noisiest authorities insisted on its being received, for
good or evil, in the superlative degree of comparison only.

---Emacs:dickens          (Text Fill)--L1--All----
```

C-w 删除了这个文本块。

如果对删除的东西不放心，可以按下“**C-x u**”组合键来撤销这次操作。撤销回来的文本仍带有文本标记；如果愿意，可以再次用“**C-w**”命令删除它。如果想移动文本，可以先给它加上文本标记，按下“**C-w**”删除该文本块；再把光标移到打算插入这段文本的位置，按下“**C-y**”组合键。如果弄错了地方，马上用“**C-x u**”撤销这次操作，再移到正确的位置并按下“**C-y**”组合键即可。

在定义文本块的时候，比较常见的做法是先在该区域的一头设置文本标记，然后再把光标移动到该文本块的另一头。下面介绍几个对大多数情况都很有用的快捷键。假设要标记一个段落。当然可以采用“把文本标记放在段落的一头，再移动到另一头”的方法。可这里有个更简便的方法：按下“**ESC h**”（命令名是**mark-paragraph**）。它会自动地把文本标记放到段落的结束位置、把光标放到段落的开始位置。类似地，“**C-x h**”（命令名是**mark-whole-buffer**）会把整个缓冲区标记为一个文本块，光标在缓冲区的开始位置，文本标记在缓冲区的结束位置。最后，如果处在文本模式里，并已经把“**C-l**”字符定义为页面的分页符，就可以用“**C-x C-p**”组合键快速标记出当前页面。把段落、页面或者缓冲区标记为文本块通常都是后续操作，比如删除操作“**C-w**”等的准备工作。

X 技巧：用鼠标标记文本块

用鼠标标记准备删除、移动或者复制文本块是很直观的。按住鼠标左键并拖动鼠标，就可以选取文本块，被选取的文本将被高亮显示。可一旦松开了鼠标左键，高亮显示区域就会消失；这虽然不会影响对文本块进行的标记工作，但人们都愿意在松开鼠标键之后还能看到文本块被标记出来。如果想在松开鼠标之后还能看到高亮显示的文本块标记，请启用 Emacs 的“临时标记模式（transient mark mode）”。之所以把它称为“临时”标记模式是因为只要对缓冲区做了修改，它就会取消文本标记，好在它带来的不便并不是很大。在 X 窗口系统下，很多人愿意加上这个模式，好让文本块显得醒目一些。输入 “**ESC x transient-mark-mode RETURN**” 将启用这个模式（注 5）。现在，在文本块的开始处按下鼠标左键，然后按着鼠标左键移动到文本块的结尾并释放，两次动作之间的文本就被标记为一个文本块了。在标记文本的时候，你可以朝屏显画面边界外的方向移动；Emacs 会自动卷屏以便能够对多屏信息进行标记。高亮显示的文本都是已经被标记的。

用鼠标标记文本的另一个方法是：把光标移动到文本块的开始，按鼠标左键并释放，然后把光标移动到文本块的结尾，按鼠标右键一次。如果是在临时标记模式里，刚才选取的文本就都将呈高亮显示状态。如果不在临时标记模式里，现在该干什么了？对，用 “**C-x C-x**” 来检查文本块是否被正确标记。

如果只想标记一个单词或者一行文本，这里有个简单点的做法：在单词或文本行上多按几次鼠标左键。双击选取一个单词；三击选取一行文本。

文本的复制

文本的复制操作是这样的：先选取一个文本块，按下 “**ESC w**”（命令名是 **kill-ring-save**）组合键或者在从 **Edit** 菜单里选择 **Copy**（复制）操作；然后把光标移动到准备插入文本的位置，按下组合键 “**C-y**”。文本的复制与文本的删除基本相同，只不过复制操作中的 Emacs 不会删除选取的文本。被复制的文本将被放到删除环里去，所以可以用 “**C-y**” 命令对它做任意次操作。

注 5： 如果打开了临时标记模式，那么不管文本标记是用鼠标还是用键盘命令加上去的，所有的文本块都将呈高亮反显状态。如果想在自己每次进入 Emacs 的时候都让临时标记模式自动打开，请阅读本章后面“对 Emacs 进行定制”一节里的有关内容。

“**ESC w**”的一个优势是它可以对只读文件或只读缓冲区进行操作。比如，如果想自己创建个关于Emacs使用技巧方面的文件，就可以用“**ESC w**”从在线帮助里复制些文本到某个缓冲区里去。

下面是文本块各种常见操作的基本步骤。

标记一个文本块：

1. 把光标移动到准备标记的文本区域的开始位置。
2. 按下“**C-@**”或“**C-SPACE**”组合键。Emacs会显示“Mark set”。
3. 把光标移动到准备标记的文本块的结束位置。
4. 如有必要，按下“**C-x C-x**”互换插入点和文本标记的位置，检查文本块是否已标记正确。

删除一个文本块：

1. 标记好准备删除的文本块。
2. 按下“**C-w**”删除该文本块。Emacs将删除该文本块。

移动文本：

1. 按“标记一个文本块”和“删除一个文本块”中的操作步骤，删除准备移动的文本。
2. 把光标移动到准备插入文本的位置。
3. 按下“**C-y**”组合键。Emacs将把刚删除的文本插入到该位置。

复制文本：

1. 给准备复制的文本做好标记。
2. 按“**ESC w**”复制文本。
3. 把光标移动到准备插入复制文本的位置，按下“**C-y**”组合键。Emacs将把刚复制的文本插入到这里。

恢复早先的删除操作

我们前面对删除环做过简单的介绍，Emacs会把删除的东西保存在这个临时性的存储区域里。我们此前一直假定试图恢复的都是最新被删除的东西。其实，删除环的用处远不止此。事实上，删除环可以把最近30次删除的文本都保存起来。我们已经看到“**C-y**”可以把最新删除的文本恢复回来；继续按下“**ESC y**”组合键，Emacs会用删除环里保存的倒数第二次删除的文本把刚恢复的文本替换掉。

请看下面的例子。假设刚删除的文本是“most recent”。“**C-y**”将把这个单词从删除环里恢复回来。按下“**ESC y**”时，Emacs会用删除环里的下一个条目（例如“second-last”）替换“most recent”。如下所示：

键盘操作	动作
C-y	This was the most recent deletion.
ESC y	This was the second-last deletion.
ESC y	This was the third-last deletion.
ESC y	This was the fourth-last deletion.

可以继续按下“**ESC y**”组合键把更早先的删除内容依次恢复回来，直到到达删除环的末端（此时将循环回最新删除的文本；这也是它被称为“环”的原因）。

如果觉得保存最近30次删除操作还不够，可以自行加大删除环的尺寸。但很少有人需要扩大删除环，除非是有很特殊的理由或者是个非常有耐心的人。不过，如果真的想试试，可以使用下面这个命令：“**ESC x set-variable RETURN kill-ring-max RETURN new-value RETURN**”（*new-value* 是一个数字）。

文本块的选取和粘贴

可以通过菜单以更直观的方式访问删除环里的文本：从**Edit**菜单里选择执行“**Select and Paste**（选取与粘贴）”操作。选择执行“**Select and Paste**”操作的时候，屏幕上会出现一个列有删除文本的窗口，最近删除的排在最顶部。为了能够尽可能多地显示以前删除的文本，各次删除操作在这个窗口里都只能用一行文本来表示。这就是说，即使删除了一个大文本块，比如500行，在这个窗口里也只能看见它的第一行，后面是省略号。（用户可能需要多熟悉几次才能适应这种安排。）虽然

窗口显示仅为一行，但选取这一行就等于是选取了被删除的整个文本块。选好准备粘贴的文本块之后，弹出窗口消失，选取的文本块就会被粘贴到编辑缓冲区里的光标位置处。

表 2-3 对与文本块有关的编辑命令进行了汇总。

表 2-3：文本块操作命令速查表

键盘操作	命令名称	动作
C-@ 或 C-SPACE	set-mark-command	标记文本块的开始（或结束）位置
C-x C-x	exchange-point-and-mark	互换插入点和文本标记的位置
C-w 或 SHIFT-DELETE <i>Edit→Cut</i>	kill-region	删除文本块
C-y 或 SHIFT-INSERT <i>Edit→Paste Most Recent</i>	yank	粘贴最近删除或复制的文本
ESC w 或 C-INSERT <i>Edit→Copy</i>	kill-ring-save	复制文本块（以便用“C-y”命令来粘贴它）
ESC h	mark-paragraph	标记段落
C-x C-p	mark-page	标记页面
C-x h	mark-whole-buffer	标记整个缓冲区
ESC y <i>Edit→Select and Paste</i>	yank-pop	在用过“C-y”命令以后粘贴更早删除的文本

段落重排

如果是在自动换行模式里，Emacs 会自动在每行末尾添加一个换行符；这样，每一行的长度都差不多，文件看起来也就比较整齐美观。当然，这种效果只会出现在第一遍草稿中。随着编辑工作的进行，肯定会有一些语句被改得长长短短，整个文件的格式不可能再像当初那样整齐了。经过编辑后，有些文本行会变短，有些文本行会长得超出屏显画面的边界。



Emacs 不会主动重排文本。如果想让自己的文件重新恢复当初的整齐效果，必须给出一个匀空 (fill) 命令。重排文本最简单的方法是按下“**ESC q**”组合键以执行**fill-paragraph** 命令。Emacs 将重排那个段落，然后把光标放到该段落的末尾。

但这里有一个必须引起大家注意的问题。在文本模式里，段落或者必须缩进，或者其前后必须有一个空行。如果没有在文件里给段落前后加上空行，Emacs 就会把它整个看做是只有一个段落。如果不小心按下了“**ESC q**”组合键，Emacs 就会把文件中的全部文字当做一个段落进行重排，忽略所有的换行符，结果是把它重排为一个很长的段落。如果使用的是 **nroff**、**troff** 或者 **TeX** 等排版系统，这种结果当然不会是你所期望的。幸好还可以用“**C-x u**”（命令名是 **undo**）组合键来恢复文件。如果使用 **troff** 或 **TeX** 排版系统，或者（出于种种原因）需要创建段落前后不带空白行的文件，请按下面的建议操作：

- 对 **nroff** 或 **troff** 排版系统，请使用 Emacs 编辑器的 **nroff** 模式，该模式将在第九章介绍。这样，就可以用“**ESC q**”组合键来重排段落，而排版命令将不受影响。对 **TeX** 用户来说，请使用 Emacs 编辑器的 **TeX** 模式，该模式也在第九章里讨论。这些特殊的模式重新对段落进行了定义，使 **fill-paragraph** 命令能够正确地执行。除此之外，这两种模式与 Emacs 编辑器的文本模式是很相似的。
- 不对段落进行重排，而对文本块进行重排。先定义好准备重排的文本块，再按下“**ESC x fill-region RETURN**”或者从 **Edit** 菜单里选择执行 **Fill** 操作。这个命令对文本块进行处理，能够把文本块里各个段落的格式都重新排好。

再提醒大家一句：如果在“**ESC q**”的前面加上了“**C-u**”（或者“**ESC -**”），Emacs 会对匀空操作稍微做些变化。普通的匀空命令会使右页边参差不齐。加上这两个前缀中的任何一个之后，匀空命令将会用在文本行里插入额外的空格的方法来调整右页边。不过，平心而论，我们认为这样的居右调整会降低文档的可读性。

表 2-4 对段落重排方面的编辑命令进行了汇总。

表 2-4：段落重排命令速查表

键盘操作	命令名称	动作
ESC q	fill-paragraph	重排段落
(无)	fill-region	对某个文本块中的段落进行重排
<i>Edit</i> → <i>Fill</i>		

编辑技巧和快捷键

学会基本的编辑操作——把光标移动到正确的位置、删除/复制/移动文本后，我们再介绍一些简化编辑工作的技巧。

交换位置

最常见的输入错误之一是把前后两个字符的位置打错了，通常会在敲过键盘之后立刻发现这类错误。按下“C-t”组合键可以交换两个字符的位置，把它们修改为正确的顺序，如下所示：

按“C-t”之前	按“C-t”之后
the best of times, it	the best of timeſ, it

要想交换两个字符的位置，先要把光标放到准备交换的第二个字符上。按下“C-t”组合键。（如果经常弄错两个字符的前后顺序，请考虑用第三章介绍的单词简写模式（word abbreviation mode）来自动改正这类打字错误。）

还可以交换两个单词、文本行、段落或者句子的位置。交换两个单词的位置的方法是：把光标放在两个单词的中间，然后按下“**ESC t**”组合键。Emacs在完成操作后会把光标放到这两个单词的后面。如下所示：

按“ ESC t ”之前	按“ ESC t ”之后
one three two	one two three

交换两个文本行的方法是：把光标放在第二个文本行的任意位置上，按下“**C-x C-t**”组合键。Emacs会把第二个文本行放到第一个文本行的前面去。如下所示：

按“ C-x C-t ”之前	按“ C-x C-t ”之后
second line	first line
first line	second line
third line	third line

表 2-5 对交换位置类的编辑命令进行了汇总。

表 2-5：位置交换命令速查表

键盘操作	命令名称	动作
C-t	transpose-chars	交换两个字符的位置
ESC t	transpose-words	交换两个单词的位置
C-x C-t	transpose-lines	交换两个文本行的位置
(无)	transpose-sentences	交换两个句子的位置
(无)	transpose-paragraphs	交换两个段落的位置

改变字母的大小写

在字母的大小写方面也经常会弄出一些常见而又恼人的输入错误。Emacs专门为改正字母大小写错误而准备了一些命令。如果想把某个单词的第一个字母改为大写，请把光标放到第一个字母上再按下“**ESC c**”组合键；如果想把一个单词全部改为小写字母，请按下“**ESC l**”组合键；如果想把一个单词全部改为大写字母，请按下“**ESC u**”组合键。这些键盘命令可以这样来记忆：**ESC** 加 “**c**” 表示单词的首字母要改为大写；加 “**l**” 表示要把单词的所有字母都改为小写；加 “**u**” 表示要把单词的所有字母都改为大写。但要注意：如果把光标放在单词的半中间，Emacs 只对从光标位置开始到单词尾之间的字母进行处理；也就是说，可以用“**ESC l**”组合键方便地把单词的前半部分改为小写。

如果发现刚输入的单词有错误，可以在以上几个命令的前面加上一个“**ESC -**”（按 **ESC** 键，再按连字符键）来修改。这种方法不会改变光标的位置。如果光标放在某个单词的半中间，在命令前加上一个“**ESC -**”会只对单词的前半部分（光标位置之前的单词字母）进行改正，光标位置之后的单词字母不受影响。

请看下面的例子，我们从“abcd■fghij”开始进行操作：

如果你按下	你将得到
ESC u	abcdEFGHIJ■
ESC - ESC u	ABCD■fghij
ESC c	abcdEfghij■
ESC - ESC c	Abcd■fghij

表 2-6 对与字母大小写有关的编辑命令进行了汇总。

表 2-6：字母大小写编辑命令速查表

键盘操作	命令名称	动作
ESC c	capitalize-word	把单词的首字母改为大写
ESC u	upcase-word	把单词的字母全部改为大写
ESC l	downcase-word	把单词的字母全部改为小写
ESC - ESC c	negative-argument; capitalize-word	把前一个单词的首字母改为大写
ESC - ESC u	negative-argument; upcase-word	把前一个单词的字母全部改为大写
ESC - ESC l	negative-argument; downcase-word	把前一个单词的字母全部改为小写

文本的改写模式

大家可能习惯了用新输入的字符取代以前的内容而不是删除它们。用这种方法来删除不想要的旧文本确实是很方便的。在 Emacs 里你也可以这样做——只要进入名为改写模式 (overwrite mode) 的副模式即可。在进入改写模式之后，输入的新文本将取代光标下面的旧字符。如果不在改写模式里 (即在普通的 Emacs 环境里)，则输入的新文本将插入到光标位置，而光标后面的旧文本将被推到右边去。(其他软件把这种模式叫做“插入模式”；因为这是 GNU Emacs 的正常工作行为，所以 GNU Emacs 没有给它专门起个名字。)

进入改写模式的办法是按下键盘上的**INSERT**键。状态行上将出现“ovwrt”字样。如果不起作用 (或者键盘上没有 **INSERT** 键)，请输入“**ESC x overwrite-mode RETURN**”。再次输入“**ESC x overwrite-mode RETURN**”将退出改写模式。利用 Emacs 命令的自动补足功能(completion)，只要输入“**ESC x ov**”再按下**RETURN** (回车键) 即可。这已经足以让 Emacs 知道要切换为改写模式了。自动补足功能是 Emacs 最好的快捷操作方式之一，我们将在第十六章对它进行介绍。

命令的中止和修改的撤销

有时候，用户可能会不小心执行了一个错误的命令，或者当命令执行到中途的时候又不想让它完成了。别担心：在 Emacs 里，既可以在命令执行到中途的时候取消，也可以在它完成后撤销它做的编辑修改。

命令的中止

如果想在命令执行到中途的时候停止它，请按“**C-g**”组合键。命令区里将出现“**Quit**”字样。

修改的撤销

如果在编辑过程中出现错误怎么办？可以按下“**C-x u**”、“**C-_**”或者“**C-/**”（命令名是**undo**）等组合键，或者从**Edit**菜单里选择执行**Undo**（撤销）操作来撤销刚才的编辑修改。连续按下“**C-x u**”组合将逐步退回到出错之前的位置。虽然**undo**命令的功能很强，但定期存盘永远是最保险的方法。我们通常会在打字过程中停顿的时候把文件存一次盘——即使只停顿几秒钟。请训练手指在准备稍事休息的时候按下“**C-x C-s**”组合键，这是一个值得培养的好习惯。

如果在执行了**undo**命令之后又想再次执行一个命令该怎么办呢？Emacs没有提供“再次执行某个命令”的命令，但可以利用**undo**命令的特性来进行以下操作：随便移动一下光标，然后再次按下“**C-x u**”组合键。Emacs会把过去撤销的命令再执行一遍。可以用这种办法恢复执行前面撤销的命令。

虽然**undo**是一个重要的命令，但如果想撤销一次大规模的修改，那么它的执行速度也会变的很慢。表2-7对3种用来撤销编辑修改的方法，以及它们各自的适用场合进行了总结。

表2-7：撤销编辑修改的3种方法

如果你	请使用以下命令
不喜欢刚做的修改，想一个一个地撤销它们	C-x u （命令名是 undo ）
想撤销自上次对文件存盘之后的所有修改	ESC x revert-buffer RETURN
想回到该文件以前的版本（即这个文件在开始这次编辑工作之前的样子）	C-x C-f filename~ RETURN C-x C-w filename RETURN

我们已经讨论过使用**undo**命令撤销编辑修改的情况；接下来我们将介绍如何用某个文件，来取代当前编辑缓冲区里的内容，和如何返回到文件的某个早期版本。

用文件内容取代编辑缓冲区中的内容

如果 **undo** 命令不够用，还有其他方法可以把文件恢复到其早期的状态。如果想让文件回到存盘时的样子，请输入 “**ESC x revert-buffer RETURN**” 或者从 **Files** 菜单里选择执行 “**Revert Buffer** (回复缓冲区)” 操作。Emacs 会做如下所示的提问：

```
Revert buffer from file filename? (yes or no)
```

其中 **filename** 是原文件名。如果想重新调入那个文件，回答 “**yes**”；如果改变了主意，回答 “**no**”。Emacs 会从保存在磁盘上的文件里读出其内容，放到当前编辑缓冲区里去，自你上次存盘之后的一切改动都将不复存在。虽然这个命令的名字是 **revert-buffer**，但请注意它只能对与文件有关联的缓冲区进行恢复。

返回文件某个以前的版本：备份文件

Emacs 会在第一次保存某个文件的时候创建一个备份文件。如果发生了灾难性的意外，而其他用来撤销编辑改动的技巧又帮不上忙的时候，就可以求助于这个备份文件。备份文件的名字与你正在编辑的文件差不多，只是在最后加上了一个波浪符 (~)。比如说，如果你正在编辑的文件是 *text*，则其备份文件就是 *text~*。

Emacs 没有提供从备份文件恢复编辑缓冲区的专用命令。最简单的方法是直接编辑备份文件，然后再把它保存为一个正式的文件。假设正工作在一个名为 *text* 的文件上，需要如下操作：先用 “**C-x C-c**” 组合键退出 Emacs，然后用 “**emacs text~**” 重新启动 Emacs；在备份文件出现在屏幕上以后，用 “**C-x C-w text RETURN**” 把它保存为一个正式的文件。Emacs 会在覆盖原始文件之前提问：

```
File text exist; overwrite ? (y or n)
```

输入 “**y**”，Emacs 将用备份文件覆盖原始文件。

GNU Emacs 还有一种编号备份功能。如果启用了编号备份功能，Emacs 会在每次对文件进行存盘的时候都创建一个备份文件（后缀是 “*~n~*”）。“*n*” 是一个数字，每存一次盘，这个数字就加 “1”。如果文件的旧版本很重要，就应该考虑使用这项功能，只要愿意，文件的旧版本可以永远保存下去。不过，编号备份功能往往会浪费大量的磁盘空间，一个比较好的折中办法是让 Emacs 只保存最近的 *n* 个版本，这里

的 *n* 是要保存的版本个数。编号备份的控制变量将在附录 C 里介绍。如果对完备的版本控制有兴趣，可以参阅将在第十五章讨论的版本控制模式（VC mode）。表 2-8 对与“命令的中止执行和撤销修改”有关的命令进行了汇总。

表 2-8：命令的中止和撤销

键盘操作	命令名称	动作
C-g	keyboard-quit	放弃当前命令
C-x u	advertised-undo ^a	撤销上一次编辑（可以重复使用）
C-_ 或 C-/ Edit→Undo	undo	撤销上一次编辑
(无)	revert-buffer	把缓冲区恢复到上次对文件进行存盘（或者自动存盘）时的状态

a. **advertised-undo** 和 **undo** 命令之间并没有真正的不同。它们的工作情况都是一样的。

恢复丢失的编辑修改

我们已经介绍了如何撤销不想要的编辑修改；可找回丢失的修改又是另外一回事了。丢失编辑修改的情况可能会发生在以下几种场合：电源突然断电、计算机突然死机、不小心关掉了计算机的电源等。在非正常退出 Emacs 的时候，也可能会丢失所做的编辑修改。幸好 Emacs 想得非常周全，它可以定期把文件保存在“自动保存（auto-save）”文件里。如果用户足够细心，就会发现“Auto saving”消息会不时地闪现在辅助输入区里。有了自动保存文件，就可以找回大部分编辑工作——即使不是全部的话。自动保存文件的文件名和正在编辑的文件一样，只是在其前后分别加上了一个井字符（#）。比如，如果正在编辑的文件是 *text*，它的自动保存文件就将是 #*text*#。

从自动保存文件恢复文本的方法是输入“**ESC x recover-file RETURN**”。Emacs 会打开一个窗口，窗口里列出了文件和与它对应的自动保存文件以便对存盘时间、文件长度等进行比较；Emacs 会提出下面这个问题：

```
Recover auto-save file #text# ?(yes or no)
```

如果想把自动保存文件的内容复制到当前文件里去，回答“yes”；如果又改了主意，回答“no”。（如果不那么肯定，可以在使用 **recover-file** 命令之前先用“**C-x C-f**”



把自动保存文件#text#读到一个缓冲区里并对它做仔细的检查。如果想具体比较这两个版本之间的不同之处。请参考第四章“对比两个窗口中的文件”一节中。)

Emacs会在什么时候创建自动保存文件呢？Emacs会在每敲了几百次键盘之后或者每当电源被切断或Emacs非正常结束时创建一个自动保存文件（注6）。可以用修改`auto-save-interval`变量的方法，来改变Emacs对文件进行自动保存的频率。在默认的情况下，Emacs会每隔300次击键就自动存盘一次。修改变量值的具体操作请参考第十一章。

Emacs的自动保存文件有一个重要的注意事项：如果在文件里进行了一次大规模的删除操作，Emacs将停止自动保存这个文件并显示一条消息通知用户。要想让Emacs再次开始自动保存这个文件，请用“**C-x C-s**”组合键给这个文件存一次盘，或者输入“**ESC 1 ESC x auto-save RETURN**”（命令中的“1”是数字1）。

Emacs的文本编辑命令到这里就介绍得差不多了，它们已经足以完成大部分的编辑工作。我们下面来学习如何对Emacs的某些功能——比如启用自动换行模式(`auto-fill mode`)——进行设置，这样就不必在每次进入Emacs的时候去启用它们了。下一小节对Emacs的定制情况做了一个简单的介绍；这个问题在第十一章有更详细的讨论。

对Emacs进行定制

如果各位是捧着这本书一直读下来的，那么可能已经列出了一个每次进入Emacs时都要去做的工作清单。这些工作可能包括：

- 启用自动换行模式，让Emacs去处理字换行问题。
- 如果有X窗口系统，启用临时标记模式使Emacs高亮显示文本块。
- 如果在流控制方面遇到了问题（比如“**C-s**”会造成死机），一劳永逸地解决它们。

注6：准确地说，这句话应该是Emacs试图在这些场合这样做，但也有Emacs力所不及的时候。这可没有办法打保票。不过我们可以为Emacs的这个功能投一张信任票。在使用Emacs进行写作的时候，我们从来没有丢失过作品，对其他软件我们可不敢这么说。

我们将介绍怎样才能给Emacs列出一个初始化工作表，让Emacs在用户每次进入的时候自动完成有关的设置工作。这些选项都是在一个名为“*.emacs*”的初始化文件里定义的。初始化文件是一些能够自动运行的文件。有些初始化文件，比如“*.login*”，会在每开始一次UNIX会话时自动运行；而其他的初始化文件，比如“*.emacs*”，会在启动一个与之关联的软件程序时自动运行。总之，“*.emacs*”文件会在启动Emacs的时候自动运行，把该文件里定义的各种选项都设置好。Emacs没有这个文件也能运行；这个初始化文件的惟一作用是让Emacs能够如人所愿地去工作。

我们将在这本书里把你可能想添加到自己的“*.emacs*”文件里去的语句文本都给出来。若想把一行文本添加到你的“*.emacs*”文件里，请按以下步骤操作：

1. 进入 Emacs（如果还没有进入）。
2. 输入“**C-x C-f ~/.emacs**”，再按下回车键。
3. 按下“**ESC >**”组合键移动到文件的末尾。
4. 把本书给出的语句准确地输入到文件里去（注 7）。
5. 按下“**C-x C-s**”，保存“*.emacs*”文件。
6. 按下“**C-x C-c**”，退出 Emacs。
7. 输入“**emacs**”，重新启动 Emacs，让新添加的语句起作用。

如果在输入方面出了点小错误（比如漏掉了一个问号或者一个括号），就会在重新启动 Emacs 的时候看到一条“Error in initial file”（初始化文件有错）的出错信息。请重新编辑“*.emacs*”文件，把添加的语句与它们的出处（不管是本书还是其他用户的“*.emacs*”文件）进行对比。只要足够仔细，通常都能把错误找出来；如果自己找不出错误，请向有“*.emacs*”文件（最好还懂LISP语言）的其他人求助。改正错误，保存文件，然后重新启动 Emacs。

注 7： *.emacs* 文件包含 LISP 函数调用表达式，因此它对语法非常挑剔。（如果“LISP 函数调用表达式”使读者感到困惑，这并不意外。只要原样键入，它就会工作。现在无需深究。读完第十三章的详细说明，就会明白。）

自动启用文本模式和自动换行模式

如果想把文本模式设置为默认的主模式，并在进入 Emacs 的时候自动启用自动换行模式，请把下面这些语句添加到 “.emacs” 文件里：

```
(setq default-major-mode 'text-mode)
(add-hook 'text-mode-hook 'turn-on-auto-fill)
```

第一条语句告诉 Emacs 把文本模式设置为默认的主模式；等于在说“如果没有另行说明，请启用文本模式”。第二条语句的作用是只要在文本模式里，就启用自动换行模式。

自动启用临时标记模式

如果想自动启用临时标记模式，请把下面这个语句添加到 “.emacs” 文件里。它的作用是为 X 用户启用高亮显示功能，使文本块的标记工作更容易进行。

```
(setq-default transient-mark-mode t)
```

这条语句的作用是启用临时标记模式。语句结尾处的 “t” 表示 “true”；“t”的相反取值是 “nil”。

解决流控制问题

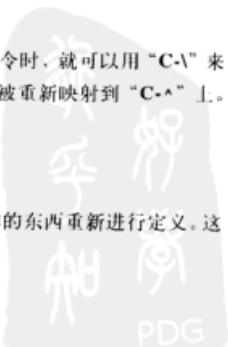
如果 “C-s” 组合键在系统上会引起死机，就说明它被用做一个流控制字符了。如果把下面这条语句添加到 “.emacs” 文件里，就可以用 “C-v” 来代替 “C-s”，用 “C-^” 来代替 “C-q”：

```
(enable-flow-control)
```

现在，再输入要求使用 “C-s” 组合键（却引起死机）的命令时，就可以用 “C-v” 来代替它。类似地，“C-q”（撤销 “C-s” 命令的操作结果）被重新映射到 “C-^” 上。

重新绑定键盘按键

“.emacs” 文件的另外一种主要用途是对 Emacs 中干扰工作的东西重新进行定义。这



里讨论的大多数键盘绑定方面的问题在 Emacs 19.29 都已经得到了解决。如果使用的是一个早期的版本，就可能需要改变这些绑定。即使使用的版本比较新，也可以根据这些例子体现的原则重新绑定其他按键。

一个比较常见而又烦人的错误是当按下 **ESC** 键的时候，系统会认为按了两次而不是一次。在许多键盘上，如果按键超过一秒钟，它就会自动重复。如果发生了这样的事情，就会撞进 **eval-expression** 命令，而这个命令通常只有 LISP 程序员才会用到。在默认的情况下，这个命令是被禁用的，Emacs 会显示一个窗口询问是否真的打算使用它。如果用户经常不小心输入 “**ESC ESC**”，就会无所适从。另外，一个经常出现意外的按键是 “**C-x C-u**”，它也有与 “**ESC ESC**” 同样的麻烦。（我们经常会在重复输入 **undo** 命令 “**C-x u**” 时错误地弄成 “**C-x C-u**”。）要想让 “**ESC ESC**” 和 “**C-x C-u**” 不再显示那些烦人的提示信息，请把下面这些语句添加到 “**.emacs**” 文件里去：

```
(global-unset-key "\e\e")
(global-unset-key "\C-x\C-u")
```

请注意击键动作在 “**.emacs**” 文件里的表示方法。**ESC** 是 “**\e**”，所以 “**ESC ESC**” 就是 “**\e\e**”，中间不能有空格。类似地，“**C-x**” 和 “**C-u**” 的前面要加上反斜线 (\)，“**\C-x**” 和 “**\C-u**” 之间也不能有空格。把这两条语句添加到 “**.emacs**” 文件里后，如果又错误地按下了这些组合键，就只会听到一声蜂鸣，但工作却不会被打断了。

在线帮助功能方面的问题

另外一个常见的错误是意外地进入了在线帮助功能。进入帮助系统的方法是按下 “**C-h**” 组合键。但在某些键盘上，“**C-h**” 组合键被映射为一个退格字符或者左箭头键，想退格的时候 Emacs 却认为是在求助。有时候，“**C-h**” 又被映射为 **DEL** 键，致使很难进入帮助功能。把下面这些语句添加到 “**.emacs**” 文件里可以解决这两个问题：

```
(define-key global-map "\C-x?" 'help-command)
(define-key global-map "\C-h" 'backward-char)
```

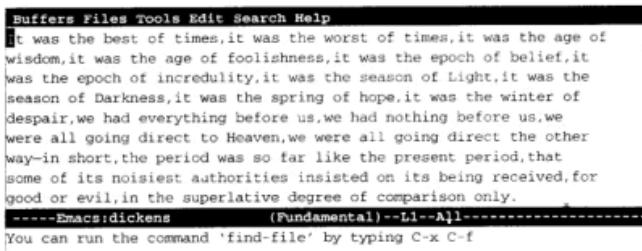
做完这些改动之后，你可以用 “**C-x ?**” 组合键进入帮助功能。



关闭按键提示功能

在 Emacs 19.30 里，如果某个长命令有一个对应的组合键，Emacs 就会在输入这个命令的时候在辅助输入区里给出一条提示符。设计这个功能的目的是帮助那些初学者，但时间一长，有的用户可能就想关掉这项功能了。请看下面这个例子。在输入“**ESC x find-file**”时请注意会发生什么事情。

输入： **ESC x find-file**



A screenshot of the Emacs interface. The menu bar at the top includes 'Buffers', 'Files', 'Tools', 'Edit', 'Search', and 'Help'. Below the menu bar, the main window displays a large block of text from Charles Dickens' 'A Tale of Two Cities': 'it was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way—in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or evil, in the superlative degree of comparison only.' A cursor is visible in the middle of the text. At the bottom of the screen, there is a status bar with the text '---- Emacs:dickens (Fundamental)--L1--All----' and below it, the instruction 'You can run the command 'find-file' by typing C-x C-f'.

Emacs 会提示这个命令的按键绑定。它会在一两秒钟之后消失，然后开始执行 **find-file** 命令。

如果不想要 Emacs 继续提示按键绑定，请把下面这些语句添加到 “**.emacs**” 文件里：

```
(setq suggest-key-bindings nil)
```

Emacs 的定制功能是很强的，用户可以让 Emacs 完全按照自己的意愿工作。更深入的定制设置将在第十一章讨论。此处这些简短的介绍一是为了引起大家的兴趣，二是为了让大家知道在必要的时候怎样往 “**.emacs**” 文件里添加简单的语句。

下一章的主题是 Emacs 提供的各种搜索功能，包括查询、替换、拼写检查、单词简写模式（word abbreviation mode，主要用在自动改正打字错误方面）等。如果你想学习这些功能，请继续阅读下一章。从这里开始，各位可以根据自己的情况有选择地来学习这本书，看哪些东西是自己最想掌握的，不必按部就班地阅读本书的其他章节。



疑难解答

- 出现错误信息“auto-save error (自动保存错误)”或“cannot create backup file (无法创建备份文件)”。这些错误信息往往让人摸不着头脑。它们的真实意思是用户正在一个没有写权限的子目录里进行编辑工作，因此 Emacs 无法创建一个自动保存文件或备份文件。如果这个子目录不存在，也会显示这条错误信息。Emacs 允许创建一个名字像“foo/bar”这样的编辑缓冲区，即使这个子目录实际并不存在，也可以开始工作。在 Emacs 试图自动保存文件之前，一切都会显得很正常。可以把正在编辑的用“C-x C-w”文件备份到另外一个子目录里去，也可以用 UNIX 命令改变这个子目录上的权限。
- 执行移动光标、插入或者删除等操作时，屏幕画面看起来不正常。比如画面上有反显的小区块、文本行尾部消失等等。终端或显示器的设置有问题。作为一种临时性的解决措施，可以试试重新绘制屏幕命令“C-l”（小写的“L”字母）以清除屏幕。然后让系统管理员把配置弄正确。
- 在修改过“.emacs”文件后重新启动 Emacs 时出现了一条“Error in initial file (初始化文件有错)”的信息。重新编辑“.emacs”文件，把添加的语句与它们的出处进行对比，看看有没有细微的输入错误。漏掉连字符或者括号这类小错都会引起这个问题。改正错误，保存文件，退出 Emacs，再重新进入。
- 在 X 下，用鼠标左键选取文本的时候，它们没有呈高亮显示状态。首先，输入“ESC x transient-mark-mode RETURN”进入临时标记模式。如果文本仍然不能在选取的时候呈高亮显示状态，那么原因可能是高亮显示颜色和窗口的背景颜色设置相同（Emacs 高亮显示文本块，可却无法分辨）。可以按第十四章介绍的方法对“.Xdefaults”文件中的显示颜色进行修改。



第三章

查找和替换操作

本章内容：

- 查找操作
- 查找和替换操作
- 拼写检查
- 单词简写模式

前两章里介绍的编辑命令带大家入门已经足够了，但离专业编辑水平的要求还有不小的差距。如果想用 Emacs 来编辑“宏篇巨作”，就应该掌握本章介绍的高级文档编辑技巧。我们将在这一章介绍在 Emacs 里查找和替换文本的各种方法。Emacs 不仅提供了能在任何一种编辑器里看到的传统的查找、查找/替换功能，还提供了几种重要的变体，其中包括递增查找、正则表达式查找和查询-替换等。Emacs 的拼写检查功能也会在这一章加以介绍，因为它也是一种替换操作（查找拼错的单词并用正确的替换之）。最后，我们将把单词简写模式（word abbreviation mode）介绍给大家，它也可以看做是一种自动替换功能，用好了可以节省大量的时间。

查找操作

在做文本编辑工作的时候，经常需要查找一些以前输入的东西。不用采取把文件从头到尾读一遍的办法来查找要找的东西，因为任何一种编辑器都提供有某种形式的查找功能，这类功能能够迅速地从文件里把特定的文本字符串找出来。Emacs 也不例外，它有自己的查找命令——事实上，它提供的查找命令极为丰富。下面是对 Emacs 的各种查找操作的一个简单介绍：

简单查找

给出一个文本字符串（叫做“查找字符串”），Emacs 会把该字符串在文件里的下次出现位置找出来。几乎所有的编辑器都有这项功能。

递增查找

Emacs的递增查找操作是这样工作的：只要输入了查找字符串的第一个字符，Emacs就开始进行查找。它会随着敲入的更多字符而继续进行查找。

单词查找

Emacs的单词查找操作与简单查找操作很相似，但前者只找出完整的单词或短语。比如，如果想查找的是单词“hat”，就不必担心找到的是单词“that”。想查找的短语跨越两个文本行时，单词查找也很有用。

正则表达式查找

正则表达式查找功能可以根据匹配模板，来查找稍有不同但又有一些共性的一组字符串。比如，如果想把文件里的“B1”和“B2”字样都找出来，就可以利用正则表达式“B[12]”来查找。有时候，正则表达式本身可能会相当复杂。我们这一章只对此做一个比较概括的介绍，对它的详细论述请参考第十三章内容。

递增正则表达式查找

把递增查找和正则表达式查找两种操作结合起来，就得到了递增正则表达式查找。

可以向前或者向后进行查找。查找既可以区分字母的大小写，也可以不区分字母的大小写。前者表示Emacs将认为大写字母和小写字母不同（比如，把单词“This”和“this”看做是不同的字符串）；后者表示Emacs不把大写单词和小写单词区别对待（即把“This”和“this”看做是一样的字符串）。在默认的情况下，Emacs的查找操作是不区分大小写的，也就是说，它认为大写字母和小写字母没有区别。一个例外：如果输入的单词里有一个以上的大写字母，Emacs就会按区分大小写的情况来对待整个查找字符串。因为Emacs会认为既然你多花了工夫去输入一个以上的大写字母，就应该是想查找与之精确匹配的东西。

替换操作与查找操作是紧密相关的。类似于查找操作的情况，Emacs也已经准备了好几 种替换操作，其中包括：

简单查找与替换

在这个操作里，Emacs将把文件里的一个字符串用另外一个字符串全部替换。但这往往不会是我们想要的结果，可以用查询-替换操作来有选择地进行替换。

查询 - 替换

在查询 - 替换操作中，Emacs 会有条件地把文件中的每个字符串替换掉。

Emacs 把查找字符串在文件里的出现位置都找出来，每找到一个，就会询问是否要进行替换。如果只打算部分地替换文件里的某个单词或短语而不是其全部，那么这种替换操作最有用。

正则表达式替换

正则表达式替换利用 UNIX 强大的“正则表达式”功能（请参考本章后面的内容）来找出替换字符串。

大家可以根据自己的需要有选择地学习这一章。千万不要被 Emacs 众多的查找操作吓倒。在实际工作中，可能用一个查找命令和一个替换命令就能应付百分之九十九的工作。比如，在编写本书的时候，大部分工作都是用递增查找和查询替换完成的；如果是作家，则单词查找应该最能满足需要；而如果是一位程序员，正则表达式查找将是得心应手的工具。如果刚开始学习 Emacs，那么最好先掌握递增查找，再继续学习本章的其他内容。总之，只有在知道都有哪些查找命令可用的情况下，才会在必要时想起它们来。

递增查找

递增查找从输入查找字符串的第一个字符起就开始行动了。对初学者而言，它并不是最容易使用的查找操作，可它有许多优点。很多用户都比较喜欢递增查找的效率；但如果在打字时经常出错的话，递增查找反而会影响工作和心情。如果真是这样，那么也许简单查找更适合一些。可话又说回来，Emacs 的递增查找确实有很多优点，从纯粹的使用角度讲也最简便，所以我们将从它开始来展开本章的讨论。

递增查找的操作方法是这样的：先敲入“C-s”组合键，再输入想查找的文本。Emacs 将临时性地进入 Isearch 模式（知不知道这一点其实无关紧要，但我们认为还是告诉大家比较好）。请看递增查找操作的工作情况：Emacs 会根据输入的每一个字符在文件里进行查找。假设想查找的单词是“meter”。在递增查找操作中，敲入字母“m”的时候，Emacs 会把光标位置以后的第一个“m”找出来；再输入字母“e”的时候又会把光标位置以后的第一个“me”找出来；当你再输入字母“t”的时候又会把光标位置以后的第一个“met”找出来；以此类推。最终，或者找到了想要找的东西，或者什么也没找到。如果找到了想要的东西，按下回车键退出这次查找，光标将停

留在文件的当前位置上。如果 Emacs 没有找到能够与查找字符串相匹配的东西，它就会在屏幕画面的底部显示一条“Search failed”（查找失败）信息，并蜂鸣报警。

下面是查找单词“*meter*”的情况；图中的数字指明了每在查找字符串里输入一个新字母后光标将会到达的位置。

输入：C-s ***meter*** RETURN

The screenshot shows an Emacs window with a menu bar at the top. The menu items are: Buffers, Files, Tools, Edit, Search, Help. Below the menu is a list of words in a buffer:

- made
- 1
- means
- 2
- method
- 3
- meteor
- 4
- meter*** (highlighted with a black rectangle)
- 5

At the bottom of the window, there is a status bar with the following text:

Emacs:rmwords (Text Fill Isearch)--L9--All--
I-search:meter

在这次递增查找操作中，Emacs 会随着输入查找字符串“*meter*”而把光标依次移动到位置 1、2、3 等等。另外，请注意状态条上出现的“l-search”字样。

如果找到了正在查找的字符串，可它的位置却不是你所希望的，该怎么办呢？比方，想查找的单词是“*eschatology*”并已经找到了一个，如果它不是想找的那个，再次按下“C-s”组合键，使 Emacs 继续查找下一个出现“*eschatology*”的地方。Emacs 会用当前查找字符串去进行查找，用不着重新输入。

找到想找的文本后，千万要记得按下回车键。忘记结束查找操作（即按下回车键或者输入其他光标移动命令）是一个很常见的错误——输入一些东西后，Emacs一下子跑到文件中的其他地方去了。出现这种情况的原因是 Emacs 认为还在进行查找，它会把刚才输入的字符看做查找字符串的一部分。

如果在输入查找字符串时打错了字，递增查找往往会让人心烦意乱。如果真的在查找字符串里错误地输入了另外一个字符，可以用 DEL 键加以改正；Emacs 会根据改正后的字符串返回到刚才的位置。如果继续按动 DEL 键来删除查找字符串里的字符，Emacs 会依次返回到以前的匹配位置。

取消一次查找操作（即放弃这次查找操作）的办法是按下“**C-g**”组合键。这个命令会回到开始查找操作之前的地方去。

注意：当 Emacs 正在进行查找操作的时候，“**C-g**”组合键的含义会有点小变化。这一点很重要，特别是当需要在 Emacs 找到它正在查找的东西之前，取消这次查找操作的时候要注意。按一次“**C-g**”组合键只会回到 Emacs 刚才找到的地方，光标将在最后一个字符串上；而这次查找操作仍未取消。按两次“**C-g**”组合键才能彻底取消这次查找操作，回到最初开始查找操作的地方去。

如果想向后（即向文件头方向）进行查找，请使用“**C-r**”组合键。它除了查找方向相反之外，与“**C-s**”完全一样，它会把光标放在找到的文本的开始。就像重复按下“**C-s**”组合键时的情况一样，也可以再次按下“**C-r**”组合键继续查找下一个出现查找字符串的地方而不用再次输入查找字符串。

如果不想输入查找字符串，那么可以把编辑缓冲区里的文本复制到查找字符串里去。用“**C-s C-w**”组合键（可以把“**C-s C-w**”看做“复制单词”命令）可以把从光标位置到下一个标点符号或空格符之间的文本都复制到查找字符串里去。如果想把从光标位置到行尾之间的文本都复制到查找字符串里，就要使用“**C-s C-y**”组合键。用这种办法得到的查找字符串会把原文中的大写字母都转换为小写，这就保证了查找操作不区分字母的大小写情况。用“**C-s ESC y**”组合键可以把删除环里的文本复制到查找字符串去；而在用过这条命令之后，还可以用“**ESC p**”组合键来查看删除环里的上一个条目。如果用“**ESC n**”跑过了头，可以再用“**ESC p**”倒退回来。

在递增查找操作开始之后，有些按键（比如向左键 **RETURN** 和退格键 **DEL**）的功能与平时有所相同。这听起来虽然很费解，可习惯起来却很快。表 3-1 对递增查找操作中特殊按键的功能进行了总结。

表 3-1：递增查找命令速查表

键盘操作	命令名称	动作
C-s	isearch-forward	向前（朝文件尾方向）开始递增查找操作；后面是查找字符串。另外，（向前）查找下一个出现查找字符串的地方

表 3-1：递增查找命令速查表（续）

键盘操作	命令名称	动作
C-r	isearch-backward	向后（朝文件头方向）开始递增查找操作；后面是查找字符串。另外，（向后）查找下一个出现查找字符串的地方
RETURN	(无)	退出查找操作
C-g	keyboard-quit	取消递增查找操作（你可能需要连接它两次）
DEL	(无)	删除查找字符串中的字符
C-s C-w	(无)	开始递增查找操作；把光标位置处的单词用做查找字符串
C-s C-y	(无)	开始递增查找操作；把光标位置到行尾之间的文本用做查找字符串
C-s ESC y	(无)	开始递增查找操作；把删除环中的文本用做查找字符串
C-s C-s	(无)	重复刚才的查找操作

简单查找

如果在输入查找字符串的时候没有出现打字错误，那递增查找的效率还是很高的。可要是打错了字，就不得不改正打错的字母后才能继续操作。如果总是出这样的错，你大概就不会喜欢用递增查找了。Emacs 中的简单查找（或者叫非递增查找）可能更适合你的口味。向前（朝文件尾方向）执行的简单查找的操作方法是：先按下“C-s RETURN”或者从“Search（查找）”菜单里选择执行“Search”操作，然后输入查找字符串、再按下回车键；Emacs 就开始查找了。再次按下“C-s”将让 Emacs 去继续查找下一个。向后（朝文件头方向）执行的简单查找的操作方法是：先按下“C-r RETURN”或者从“Search”菜单里选择执行“Search Backwards（向后查找）”操作，然后输入查找字符串、再按下回车键。表 3-2 对简单查找的有关命令进行了汇总。



表 3-2：简单查找命令速查表

键盘操作	动作
C-s RETURN <i>searchstring</i> RETURN <i>Search→Search</i>	向前（朝文件尾方向）开始一次非递增查找操作
C-s	向前查找下一个
C-r RETURN <i>searchstring</i> RETURN <i>Search→Search Backwards</i>	向后（朝文件头方向）开始一次非递增查找操作
C-r	向后查找下一个

单词查找

如果在查找一个短语的时候，明明文件里有这样的短语，可用递增查找却找不出来，那请试试单词查找。（用递增查找操作找不到这个短语的原因，极可能是这个短语跨越了两个文本行；也就是说，它中间多了一个换行符。）单词查找是一种非递增性的查找，它不会受换行符、空格和标点符号的影响，但要求查找字符串必须与文件里的单词完整地匹配。

单词查找的操作方法是：按下“**C-s RETURN C-w**”（命令名是 **word-search-forward**）组合键，辅助输入区里会出现“Word search”（单词查找）字样（不要被它前面的提示信息所迷惑：按下“**C-s**”后会先出现一个“**I-search**”，按下回车键后又会出现一个“**search**”提示符，别管它们。）；然后输入查找字符串，再按下回车键。Emacs 开始查找给定的字符串。如果想向后（朝文件头方向）进行单词查找，按下“**C-r RETURN C-w**”。举个例子，假设对下面这段文本进行编辑，并且光标放在文本块的第一个字符上：

```
He said, "All good elephants are wise, aren't they?"
She answered, "Some are smarter than others, but we
think this is socially conditioned."
```

则使用“**C-s RETURN C-w they she RETURN**”命令，将把光标放到单词“*She*”的后面。虽然这条命令看起来有点繁琐，可它的意思还是很清楚的：这是一条用来查找（“**C-s RETURN C-w**”）单词“*they*”的单词查找命令，“*they*”的后面必须跟有单词“*she*”，而“*they*”和“*she*”之间的标点符号（“?”）和换行符不会影响查找结果。

假设想查找的单词是“the”，并且不想撞到*thence*、*there*、*theater*、*thesis*和*blithe*或者碰巧包含字母“the”的其他单词。这种情况用递增查找和简单查找都不太好对付——最佳办法是采用单词查找。如果撰写一篇论文，那么单词查找往往是最需要的方法。就这三种基本的查找操作而言，单词查找是惟一能够在将要查找的短语跨越两个文本行的情况下，找到该短语的查找操作。

在学完这三种最常用的查找操作之后，可以多做些练习，看哪一种最能满足自己的工作需要。

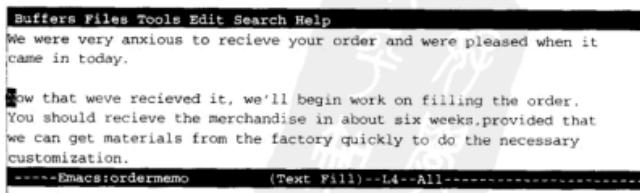
查找和替换操作

在工作中，往往需要把一个查找操作和一个替换操作结合起来使用。比如，如果拼写能力比较弱，就可能出现需要把文件中所有的“recieve”都替换为“receive”的情况。最笨的办法是先查找“recieve”，每找到一个就手动改正。Emacs准备了一个查找和替换操作，用一条命令就可以把所有的“recieve”都改正过来。

基本查找与替换操作

假设正好遇上我们刚讲的情况——想把文件里的某个字符串都替换为另一个字符串。我们不敢肯定是不是把所有的“receive”都错拼为“recieve”了，但肯定得把拼错的地方都改过来。当想把文件里的某个字符串都替换掉的时候，只需用一条简单的命令就能让Emacs干好这件事——先输入“**ESC x replace-string RETURN**”，再输入查找字符串，按下回车键；然后输入替换字符串，再次按下回车键。Emacs将从光标位置开始，把文件里所有出现查找字符串的地方都替换掉。如果想对整个文件进行查找和替换，可以在使用这个命令之前先用“**ESC <**”组合键或**HOME**键移动到文件的开头。请看下面**replace-string**命令的用法示例。

初始状态：



The screenshot shows the Emacs interface with the following details:

- Menu Bar:** Buffers, Files, Tools, Edit, Search, Help.
- Buffer Content:** The buffer contains two paragraphs of text.
 - The first paragraph: "We were very anxious to recieve your order and were pleased when it came in today."
 - The second paragraph: "Now that weve recieved it, we'll begin work on filling the order. You should receive the merchandise in about six weeks, provided that we can get materials from the factory quickly to do the necessary customization."
- Status Bar:** Shows the file name "ordermemo" and the mode "Text Fill". It also indicates the current position is at line 4, column 1, with the message "All-----".

在这个画面里，“receive”被错拼为“recieve”三次，但光标位置在第一个被拼错的单词后面。

现在来改正它们。

输入： **ESC x replace-string RETURN recieve RETURN receive RETURN**

```
Buffers Files Tools Edit Search Help
We were very anxious to recieve your order and were pleased when it
came in today.

Now that we've received it, we'll begin work on filling the order.
You should receive the merchandise in about six weeks, provided that
we can get materials from the factory quickly to do the necessary
customization.
--**-Emacs:ordermemo          (Text Fill)--L5---All-----
Replaced 2 occurrences
```

只有光标位置后面的拼写错误被改正过来了，第一个句子里的单词“recieve”仍然是不正确的。注意 **replace-string** 命令是如何把“recieve”改正为“receive”的。

如果对字母的大小写情况没有把握怎么办？单词“recieve”可能会出现在句子的开始，也就是说，它的第一个字母可能是大写的。是不是要分别进行两次查找呢？没有必要。Emacs 在进行替换操作时能够把字母大小写方面的问题处理好。

查询 - 替换操作

有时候，可能不需要把出现查找字符串的每一个地方都替换过来；如果文件的长度超过五行，对整个文件进行全局性的替换就有些不够慎重。如果想一个一个地决定是否需要进行替换，可以使用查询 - 替换操作，以便能够有选择地对文件里的某个字符串进行改正。Emacs 找到一个出现查找字符串的地方后，会询问是否进行改正，可以根据具体情况做出回答。

查询 - 替换的操作方法是：按下 “**ESC %**” 组合键或者从 “**Search**” 菜单里选择执行 “**Query Replace** (查询 - 替换)” 操作。辅助输入区里会提示 “**Query replace:**”。输入查找字符串并按下回车键。接下来将出现提示信息

Query replace searchstring with:

输入替换字符串并按回车键。到目前为止，情况与简单查找和替换操作差不多；只是提示符不同。

现在，Emacs 开始搜寻查找字符串的第一个出现位置。找到之后，将出现一个新的提示符，如下所示：

Query replacing *searchstring* with *newstring*

Emacs 会停下来等待指示，然后根据回答来决定是否需要进行替换。表 3-3 列出了可能的回答及其结果。

表 3-3：查询 - 替换操作中的响应

键盘操作	动作
ESC %	开始查询 - 替换操作 <i>Search</i> → <i>Query Replace</i>
SPACE 或 y	用新字符串替换查询字符串 <i>searchstring</i> ，然后前进到下一个位置
DEL 或 n	不替换；前进到下一个位置
.	在当前位置做替换后退出查询 - 替换操作
,	替换并显示替换情况（再按空格键或“y”后才移动到下一个位置）
!	对后面的文件内容全部进行替换，不再提问是否要进行替换
^	返回上一次进行了替换的位置
RETURN 或 q	退出查询 - 替换操作
C-r	进入递归编辑状态（马上就要讲到）
C-w	删除此处内容并进入递归编辑状态（好做其他修改）
ESC C-e	退出递归编辑状态，继续完成查询 - 替换操作
C-j	退出递归编辑状态和查询 - 替换操作

看起来好像有很多键盘命令需要记忆，其实只要知道了其中的两三个就已经够用了。在大多数情况下，可以按空格键告诉 Emacs 进行替换并进到下一个位置，或者按“n”键跳过这个位置进到下一个地方。如果对替换后的情况没有把握，按一次逗号 (,)。

键：Emacs 完成此处的替换但不再继续前进，再按一次空格键它才会往下进行。在做了几次之后，可能发现没有必要再挨个查看每一处修改；此时可以按下惊叹号（!）键让 Emacs 把以后的替换都做完，用不着再询问是否需要进行替换了。只要记住了这几个按键，一切就尽在掌握之中。

为了让大家有一个真实的感受，我们再来看看刚才的例子。还是假设“receive”被拼错了。

输入：ESC % recieve RETURN receive RETURN

```
Buffers Files Tools Edit Search Help
Now that we've received it, we'll begin work on filling the order.
You should receive the merchandise in about six weeks, provided that
we can get materials from the factory quickly to do the necessary
customization.

----- Emacs:ordermemo (Text Fill)--L4--All-----
Query replacing recieve with receive:(? for help)
```

做完替换第一个单词的准备工作，按空格键继续。

按下：SPACE

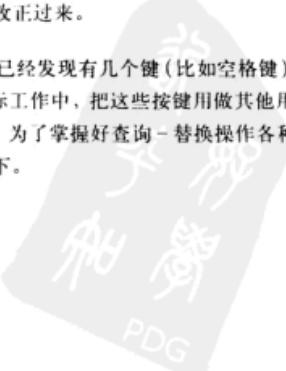
```
Buffers Files Tools Edit Search Help
Now that we've received it, we'll begin work on filling the order.
You should receive the merchandise in about six weeks, provided that
we can get materials from the factory quickly to do the necessary
customization.

----- Emacs:ordermemo (Text Fill)--L5--All-----
Query replacing recieve with receive:(? for help)
```

按下空格键之后，Emacs 替换了第一个单词，查询 - 替换操作进到第二个单词。

重复上述过程直到到达文件的末尾。我们已经说过，按下惊叹号（!）键将把文件其余部分的有关位置都改正过来。

根据表3-3，大家可能已经发现有几个键（比如空格键）在查询 - 替换操作过程中有着特殊的含义。在实际工作中，把这些按键用做其他用途不会造成混乱，但在书里写起来却有点怪怪的。为了掌握好查询 - 替换操作各种响应的使用方法，可能需要找个文件实际操练一下。



查询 - 替换（以及其他复杂命令）的重复执行

在学完查询 - 替换操作的基本用法之后，我们再介绍一种快捷操作方式。这种快捷操作方式不仅能够用在查询 - 替换操作里，还可以用在 Emacs 几乎所有的地方：复杂命令的重复执行（前后两次操作可能会稍有不同）。有时候、会不小心退出了查询 - 替换操作，或者又想对替换字符串再稍加修改。就拿刚才的例子来说，对文件进行的查询 - 替换操作并没有从头开始。用不用再把整个命令重新输入一遍呢？不用。只要把光标移到文件的开始再按下“**C-x ESC ESC**”组合键就行了。最后一次输入的复杂命令将被调出来。如果它不是想要的，可以再用“**ESC p**”组合键调出它前面的那个命令（随便按多少次都行；如果过了头，可以用“**ESC n**”后退一条命令）。举个例子，把光标移到文件的开头，然后重复执行一次刚才进行的查询 - 替换操作。

按下：**ESC <**，然后再按下 **C-x ESC ESC**

```
Buffers Files Tools Edit Search Minibuf Help
We were very anxious to recieve your order and were pleased when it
came in today.

Now that weve received it, we'll begin work on filling the order.
You should receive the merchandise in about six weeks, provided that
we can get materials from the factory quickly to do the necessary
customization.
--**-.Emacs:ordermemo      (Text Fill)--L1--All-----
Redo:(query-replace "recieve" "receive" nil)
```

先用“**ESC <**”组合键移动到文件的开头；然后按下“**C-x ESC ESC**”组合键，最后一条复杂命令被调了出来。

因为它就是想要的命令，所以用不着再按“**ESC p**”组合键查看前一条命令了。如果愿意，还可以在按下回车键之前对查询 - 替换字符串做些改动。

按下：**RETURN**

```
Buffers Files Tools Edit Search Help
We were very anxious to receive your order and were pleased when it
came in today.

Now that weve recieved it, we'll begin work on filling the order.
You should recieve the merchandise in about six weeks, provided that
we can get materials from the factory quickly to do the necessary
customization.
--**-.Emacs:ordermemo      (Text Fill)--L1--All-----
Query replacing recieve with receive:
```

按下回车键将再次执行这条命令。

刚才已经讲过，“**C-x ESC ESC**”这种快捷操作方式对在辅助输入区里有补充性输入内容的一切命令都有效，绝不仅限于查询 - 替换命令。只不过在查询 - 替换操作里用得最多而已（它还很适合用来重复执行键盘宏命令，我们将在第十章讨论这方面的内容。）。

递归编辑

在进行查询 - 替换操作的时候，几乎不可避免地会看到还有其他一些地方需要修改。不信就试试——肯定没错！第一反应是记下那些错误，等完成查询 - 替换操作之后再来修改它们；第二反应是着急上火——等完成替换操作后才发现自己已经想不起来刚才看到哪儿还出了错。

幸好Emacs已经想到这一点了。它允许在一次查询 - 替换操作的过程中再开始一次递归编辑。“开始一次递归编辑”的意思是，暂时放下手里的查询 - 替换操作去任意做一些其他的修改。当退出递归编辑状态的时候，查询 - 替换操作将从刚才暂停的位置开始继续进行。

要想在查询 - 替换操作的过程中开始一次递归编辑，需要按下“**C-r**”组合键。（注意：类似于许多其他的组合键，“**C-r**”在查询 - 替换操作中的含义与它在正常Emacs编辑活动中的含义是不同的。）进入递归编辑状态的时候，状态条上将会多出一对方括号 ([])。再以订单备忘录为例对此做个说明。已经用查询 - 替换操作找到了第一个“*recieve*”并正准备按下空格键来改正它。就在这时，发现没有把“*we've*”里面的上撇号打上去。让递归编辑来帮忙吧。

输入：**C-r**

```
Buffers Files Tools Edit Search Help
Now that we've received it, we'll begin work on filling the order.
You should receive the merchandise in about six weeks, provided that
we can get materials from the factory quickly to do the necessary
customization.
----- Emacs:ordermemo      [(Text Fill)]--L4--All-----
```

在进入递归编辑状态的时候，请注意“(Text Fill)”两端的方括号。

做你想做的编辑工作吧——现在进入一个类似于Emacs标准编辑状态的编辑模式。把光标向回移动几个字符并改正“*we've*”。可以重新开始查询 - 替换操作时，按下

“**ESC C-c**”组合键（这个组合键的输入方法是：按下再释放**ESC**键，然后按“**C-c**”）。这个命令让Emacs退出递归编辑，并重新开始继续执行查询-替换操作。Emacs将回到刚才离开的位置；可以像什么事情也没发生过那样继续进行替换操作。

给“*wewe*”中间加一个上撇号使之变成“*we've*”，然后按“**ESC C-c**”组合键

```
Buffers Files Tools Edit Search Help
Now that we've received it, we'll begin work on filling the order.
You should receive the merchandise in about six weeks, provided that
we can get materials from the factory quickly to do the necessary
customization.
-- *-- Emacs:ordermemo (Text Fill) --L4--All-----
Query replacing receive with receive:(?for help)
```

递归编辑的工作完成后，返回查询-替换操作。按空格键修改“*receive*”。

如果想在退出递归编辑的同时也取消查询-替换操作，可以按下“**C-]**”（命令名是**abort-recursive-edit**）或“**ESC x top-level RETURN**”而不是“**ESC C-c**”。

Emacs高级用户请注意：可以随时（不仅仅是在查询-替换操作的过程中）开始一次递归编辑。“**ESC x recursive-edit RETURN**”命令会进入递归编辑状态；而“**ESC C-c**”会退出递归编辑状态，返回到此前的工作中去。甚至可以在递归编辑里再来一次递归编辑，不过每多增加一个递归层次，把自己弄糊涂的可能性也将增大一层。

找操作中的字母大小写问题

在默认的情况下，Emacs的查找操作是不区分大小写字母的。在对单词“*random*”进行查找的时候，“*random*”、“*Random*”、“*RANDOM*”以及古怪的“*RanDoM*”、“*rANDOM*”等单词都会被找出来。在进行替换的时候，Emacs会根据被替换单词中的字母大小写情况做相应的替换。如果用“*tandem*”来替换“*random*”，那“*Random*”将被替换为“*Tandem*”，而“*RANDOM*”将被替换为“*TANDEM*”。

也就是说，默认的查找和替换操作差不多完全能够满足要求：它们在不考虑字母大小写情况的前提下找到一个查找字符串，然后根据其上下文来调整替换字符串中字母的大小写情况。但有时可能需要更精确的控制。变量**case-fold-search**决定查找操作是否要区分字母的大小写情况。它对各种查找操作——包括递增查找、单词查找、查找-替换操作中的查找等等都有影响。在默认的情况下，**case-fold-search**

的值被设置为“t”，意思是“如果用户输入的字符串不是大小写混杂或全都是大写字母，就不区分大小写字母”。这种默认设置通常也是需要的。但如果真的需要在查找操作中区分大小写字母，就要用“**ESC x set-variable RETURN**”命令把**case-fold-search**的值设置为“nil”。Emacs会提示输入一个变量名，输入“**case-fold-search RETURN**”。Emacs再提示输入一个新值，输入“**nil RETURN**”。

类似地，如果不想对替换字符串中的字母大小写情况进行调整，可以修改变量**case-replace**的值。这个变量的缺省值也是“t”（代表“true”），意思是“根据原始文本调整替换字符串中的字母大小写情况”——如果原单词的首字母是大写的，替换字符串的首字母也要大写等等。把这个变量的值设置为“nil”意味着“不要对替换字符串中的字母大小写情况进行调整，就按输入进行替换”。如果想修改变量**case-replace**的值，请按刚才介绍的步骤来调用**set-variable**命令。我们认为把调整替换字符串字母大小写情况设置为“on”是有用的，特别在使用的查找是不区分大小写字母的时候更有必要。但大家应该在这个问题上花点时间练练手，这样才能真正体会到其中的奥妙。

set-variable命令只能临时性地改变一下Emacs的行为。只要启动了一次新的Emacs会话，甚至只要在同一次Emacs会话里开始编辑另一个文件，就将回到默认的行为。这可能正是需要的，因为分别对首字母大写的单词和小写单词进行查找是不方便的。如果把下面这些语句添加到“*.emacs*”文件里，就可以永久性地改变这些变量的设置值：

```
(setq-default case-fold-search nil)    ; 要求精确匹配  
(setq-default case-replace nil)        ; 替换时不改变大小写情况
```

要想让这些语句起作用，需保存“*.emacs*”文件，再重新启动 Emacs。

查找与替换操作中的正则表达式

前面介绍的查找、查找和替换操作虽然不算很复杂，但用来完成日常工作基本上够用了。但有些场合单靠这些比较简单的操作是无法应付的。正则表达式是UNIX操作系统一项基础性功能，并且已经在Emacs里得到了实现。正则表达式是程序员或UNIX黑客们手里的法宝，它能够利用含有各种通配符的字符串进行复杂的查找。对正则表达式细致全面的讨论请参考第十三章；本小节的目的是把正则表达式在查找操作中的基本用法介绍给大家，让各位对利用正则表达式进行文本查找的基础性知

识有一个了解。如果想对正则表达式做深入的研究,请参考O'Reilly出版公司出版的《sed & awk》,这本书的作者是Dale Dougherty和Arnold Robbins。

表3-4列出了一些可以用在正则表达式里的字符。

表3-4: 用来建立正则表达式的字符

字符	匹配情况
^	匹配行首
\$	匹配行尾
.	匹配任意单个字符(类似于文件名中的问号?)
*	匹配任意(零或以上)个字符(这是一个真正的通配符,类似于文件名中的星号*)
\<	匹配单词的开头
\>	匹配单词的结尾
[]	匹配方括号中的任何一个字符;比如"[a-z]"将匹配任意一个字母表字符

如果用“`^word$`”做一次正则表达式查找,那么找到的将是独占一个文本行的单词“`word`”。字符“`^`”的含义是字母“`w`”必须是文本行的第一个字符,而字符“`$`”的含义是字母“`d`”必须是文本行的最后一个字符。

如果想把前有“`beg`”、后有字母“`s`”的单词都找出来,就应该选用“`beg[a-z]*s`”做为正则表达式。它将把单词“`begins`”、“`begets`”、“`begonias`”以及“`shibegrees`”、“`ultbegaslia`”等不常用的单词都找出来。如果不想要那些不常用的单词——即如果只想找出以“`beg`”开头、以字母“`s`”结尾的单词,就要用“`\<beg[a-z]*s\>`”做为正则表达式。特殊字符组合“`\<`”的作用是匹配一个单词的开头;而字符组合“`\>`”的作用是匹配单词的结尾。如果只想找出“`beg`”、“`big`”、和“`bag`”等以一个字母“`b`”开头、中间有任意一个字母、再以一个字母“`g`”结束的单词,但不想找到“`begonias`”之类的单词,或者其他含有“`beg`”等字符串的古怪单词,就要用“`\<b[a-z]g\>`”做为正则表达式。

如果想对“`^`”、“`$`”、“`.`”、“`*`”、“`[`”、“`]`”或其他特殊字符进行查找,显然不能直接使用这些字符本身。必须在它们的前面加上一个反斜线(\)。也就是说,如果想查找一个英文句号,就必须查找“`\.`”。举个例子,如果想查找下面这个电子邮件地址:

howie@mcds.com

正则表达式就将是：

howie@mcds\ .com

可以把正则表达式用在递增查找和查询 - 替换操作中。表 3-5 列出了正则表达式查找操作所使用的各种命令。除初始命令稍有不同以外，这类查找与本章前面介绍的查找操作都是一样的。

表 3-5：正则表达式查找命令速查表

键盘操作	命令名称	动作
ESC C-s RETURN <i>Search→Regexp Search</i>	re-search-forward	向前（朝文件尾方向）查找一个正则表达式
ESC C-r RETURN <i>Search→</i> <i>Regexp Search Backwards</i>	re-search-backward	向后（朝文件头方向）查找一个正则表达式
ESC C-s	isearch-forward-regexp	向前（朝文件尾方向）递增查找一个递增正则表达式
ESC C-r	isearch-backward-regexp	向后（朝文件头方向）递增查找一个递增正则表达式
(无) <i>Search→</i> <i>Query Replace Regexp</i>	query-replace-regexp	查询 - 替换一个正则表达式
(无)	replace-regexp	无条件地对一个正则表达式做全局性替换（谨慎使用）

拼写检查

Emacs 中的拼写检查可以采用两种方法：使用 Ispell 或者使用 UNIX 操作系统的拼写检查器 **spell**。就我们个人的观点而言，Ispell 是一种更好的选择，所以我们的讨论将从它开始。

在开始之前，我们先讲点关于拼写检查器的故事。拼写检查器都有一个词汇表，里面是它们能够认出来的单词，人们把这类词汇表叫做“字典”。如果单词没有出现在

字典里，拼写检查器也就无法识别它——它会认为这个单词拼错了。有些单词虽然拼写检查器不认识，可单词本身却是正确的（比如人名或者专业术语等）；将要介绍的两种拼写检查器都允许跳过这类单词，并且还允许把这些“额外的”单词插到个人字典里，这样，当拼写检查器下次遇见它们的时候就能认出它们来了。当然，拼写检查器绝对无法取代人工校对；比如，它们根本发现不了把“two”或“too”错写为“to”这样的错误。尽管拼写检查器存在着各种不足，我们仍然认为它是现代社会最伟大的发明之一。请跟我们一起来学习它在 Emacs 中的使用方法吧。

Ispell 的使用方法

Ispell 提供了对一个编辑缓冲区、一个文本块、一条电子邮件消息或者一个单词进行拼写检查的操作选项（注 1）。但在输入命令告诉 Ispell 去检查哪个区域之后，这些操作选项的工作情况都将是一样的。以 **ispell-buffer** 命令为例进行讨论。如果编辑缓冲区里的单词都拼写对了，Ispell 将给出一条“Spell-checking done.”的消息。如果 Ispell 找到了一个拼错了的单词，就会看到类似与下面这样的画面：

输入： **ESC x ispell-buffer RETURN**

```

Buffers Files Tools Edit Search Help
(0)relieve (1)receive
- *Choices* - word: receive
We were very anxious to receive your order and were pleased when it
came in today.

Now that we've received it, we'll begin work on filling the order.
You should receive the merchandise in about six weeks, provided that
we can get materials from the factory quickly to do the necessary
customization.
----- Emacs: ordermemo (Text Fill)--LL--All -----
C-h or ? for more options; SPC to leave unchanged, Character to replace
word

```

Ispell 移动到它不认识的第一个单词。

注 1： 如果 Ispell 能够正常工作，它的这些操作选项还是很容易使用的。先去试试看，如果真的有问题，再来看这个脚注不迟。Emacs 要求使用 Ispell Version 3.1.x。如果安装的是 Ispell Version 4，它就不能工作。在 Emacs 19 某些早期的版本里，我们就遇到过 Emacs-Ispell 接口方面的问题。所以，即使真的遇到了问题，你也不是孤独的。在本章最后一个小节里，我们给出了一些解决这类问题的建议。

辅助输入区里出现“C-h or ? for more options”的字样。因为这是第一次使用Ispell，所以按下“?”来看看它还有哪些操作选项：

输入：?

```
Buffers Files Tools Edit Search Minibuf Help
(0)relieve (1)receive
~ *Choices* - word: recieve
We were very anxious to receive your order and were pleased when it
came in today.

Now that we've received it, we'll begin work on filling the order.
You should receive the merchandise in about six weeks, provided that
we can get materials from the factory quickly to do the necessary
customization.
-----Emacs:ordermemo      (Text Fill)--L1--A1-----
[r/R]eplace word; [a/A]ccept for this session; [i]nsert into private
dictionary [l]ook a word up in alternate dictionary; e[x/X]it;
[q]uit session [u]ncapitalized insert into dictionary.Type
```

Ispell给出一个操作选项清单。

如果Ispell列在屏幕顶部的某个单词就是想要的，按下对应的数字键，Emacs将进行替换。在这个例子里，请按下“1”。如果想自己修改那个单词，请按下“r”，Emacs会在输入了“正确的”单词之后再检查一遍。如果按下的的是“R”，Ispell将开始一次查询-替换操作，以便把编辑缓冲区里所有同样的拼写错误都改正过来。

除了对“错误的”单词进行替换之外，可能还想让Ispell跳过它。按下空格键使Ispell跳过此处拼错了的单词。如果想让Ispell在检查编辑缓冲区余下的内容时，把某个拼错了的单词都跳过去，请按下“a”（表示接受）。按下大写的“A”有另外一层含义：它告诉Ispell在这次编辑工作里接受那个单词，但只有在这个编辑缓冲区里才有效。

如果再次遇到了这个单词，而它确实拼写正确，那么可以按下“i”让Ispell把它插入到个人字典里；这个字典保存在你主目录中一个名为“ispell.words”的文件里。如果这个文件不存在，Ispell会自动创建它。如果想让Ispell把单词以小写形式插入到个人字典里，按下“u”，Ispell会把单词转换为小写形式后再放到个人字典里。



如果发现了比较复杂的错误并打算亲自改正，可以按下“C-r”开始一次递归编辑。改正错误之后，按“ESC C-c”退出递归编辑，Ispell将重新开始继续检查。（可以再去看看本章前面对递归编辑的介绍。）

因为备忘录里多次出现拼错“receive”的情况，找一个改一个未免有些麻烦。一个更好的方法是按下“R”——在编辑缓冲区里对这个单词进行查询-替换。

按下：R

```
Buffers Files Tools Edit Search Minibuf Help
(0)relieve (1)receive
- *Choices* - word: recieve
We were very anxious to recieve your order and were pleased when it
came in today.

Now that weve recievied it, we'll begin work on filling the order.
You should recieve the merchandise in about six weeks, provided that
we can get materials from the factory quickly to do the necessary
customization.
-----Emacs:ordermemo      (Text Fill) --L1--All-----
Query-replacement for:recieve
```

Ispell 提示输入对“recieve”的改正。

把“recieve”改正为“receive”，然后按下回车键

```
Buffers Files Tools Edit Search Help
(0)relieve (1)receive
- *Choices* - word: receive
We were very anxious to receive your order and were pleased when it
came in today.

Now that weve received it, we'll begin work on filling the order.
You should receive the merchandise in about six weeks, provided that
we can get materials from the factory quickly to do the necessary
customization.
-----Emacs:ordermemo      (Text Fill) --L5--All-----
Query replacing recieve with receive:(? for help)
```

Ispell 开始一次查询-替换操作。

因为想把这个拼写错误一次性地都改正过来，所以按下“!”，表示“全部替换，用不着再查询了”。



按下: !

```

Buffers Files Tools Edit Search Help
(0)w-eve (1)w eve (2)wove (3)wive (4)were (5)wee (6)weave
(7)we 've (8)wave (9)neve (:eve
- *Choices* - word: weve
came in today.

Now that we've received it, we'll begin work on filling the order.
You should receive the merchandise in about six weeks, provided that
we can get materials from the factory quickly to do the necessary
customization.
--***-Emacs:ordermemo      (Text Fill)--L4--Bot-----
C-h or ? for more options; SPC to leave unchanged, Character to replace
word

```

Ispell 替换掉这个单词，然后进到下一个拼写有误的地方：“weve”。

按下: 7，选择 “we've”

```

Buffers Files Tools Edit Search Help
(0)relied (1)received
- *Choices* - word: received
We were very anxious to receive your order and were pleased when it
came in today.

Now that we've received it, we'll begin work on filling the order.
You should receive the merchandise in about six weeks, provided that
we can get materials from the factory quickly to do the necessary
customization.
--***-Emacs:ordermemo      (Text Fill)--L4--All-----
C-h or ? for more options; SPC to leave unchanged, Character to replace
word

```

Ispell 改正 “weve”，然后进到下一个拼写出错的地方。

虽然前面已经改正了 “receive”，但 Ispell 会认为 “recieved” 是另外一个单词，所以这个拼写检查器会把它当做另外一个拼写错误找出来（注 2）。

注 2：请注意：Ispell 中的查询—替换操作与我们前面介绍的情况有一个细微的差别。普通的查询—替换操作会把 “receive”的各种形式都找出来，不管它是不是内含在另外一个单词里（比如 “recieved” 等）。但在 Ispell 和我们马上将要讲到的 UNIX 拼写检查器里，单词的不同形式会被认为是不同的。

按下: I

```
Buffers Files Tools Edit Search Help
We were very anxious to receive your order and were pleased when it
came in today.

Now that we've received it, we'll begin work on filling the order.
You should receive the merchandise in about six weeks, provided that
we can get materials from the factory quickly to do the necessary
customization.
*** Emacs:ordermemo      (Text Fill)--L1-- All
Spell-checking done
```

Ispell 结束拼写检查工作。

查单词

在打字的时候，有时会说：“哎，这个单词怎么拼呀？”或者“我好像拼得不对”。如果想对光标位置上的单词进行检查，请按下“**ESC \$**”（命令名是`ispell-word`）组合键，或者从“**Spell**”（拼写）菜单里选择执行“**Check Word**”（检查单词）操作。Ispell 将对那个单词进行检查。如果单词拼写无误，Ispell 将显示“`word: ok`”；如果单词拼写错了，Ispell 会显示一个窗口，窗口里是刚介绍过的各种操作选项。

如果实在想不起某个单词应该怎样拼，可以先输入那个单词的头几个字母，然后按“**ESC TAB**”（命令名是`ispell-complete-word`）组合键。Ispell 会把单词的各种补足形式列出来，可以输入字符从中挑选一个。注意：Ispell 的自动补足功能只能用在文本模式里。（其实它也可以用在另外几个用来对文本进行置标排版的编辑模式里，比如 nroff 模式。另外，“**ESC TAB**”组合键在那些程序设计类编辑模式里另有其他含义。）比如，不知道“`disambiguate`”该怎样拼写。我们将像下面这样进行操作：

输入：**dis ESC TAB**

```
Buffers Files Tools Edit Search Help
(0)dis (1)disaccharide (2)disambiguate (3)disastrous (4)disburse
(5)disc (6)discern (7)discernible (8)disciple (9)disciplinarian
(:)disciplinary (;)discipline (<)disco (=)discoid (>)discomfit
(@)discordant (B)discovery (C)discreet (D)discrepant (E)discrete
(F)discretion (G)discretionary (H)discriminable (I)discriminant
(J)discriminate (K)discriminatory (L)discus (M)discuss (N)discussant
- *Choices* - word: dis
Many words have multiple meanings. If a single meaning is possible for
a word, it takes a linguist to dis
*** Emacs:semantics      (Text Fill)--L2-- All
C-h or ? for more options; SPC to leave unchanged, Character to replace
word
```

按“2”键选择单词“`disambiguate`”的正确拼写。

Ispell 进程

只要启用过 Ispell，它就将一直运行在后台等待再次使用。如果想杀掉 Ispell 进程（比如，因为它而使系统速度变慢了），请输入“**ESC x ispell-kill-ispell**”或者从“**Spell**”菜单里选择执行“**Kill Process**（终止进程）”操作。

表 3-6 对 Ispell 命令进行了汇总，包括 **Spell** 菜单中的有关操作选项。

表 3-6: Ispell 命令速查表

键盘操作	命令名称	动作
ESC \$ <i>Edit→Spell→Check Word</i>	ispell-word	检查光标位置上的单词或者光标后面的单词
(无) <i>Edit→Spell→Check Region</i>	ispell-region	检查文本块里的单词
(无) <i>Edit→Spell→Check Buffer</i>	ispell-buffer	检查缓冲区里的单词
(无) <i>Edit→Spell→Check Message</i>	ispell-message	检查电子邮件正文里的单词
C-u ESC \$ <i>Edit→Spell→Continue Check</i>	ispell-continue	让 Ispell 重新开始继续执行；这个命令只有在运行过 Ispell 并用“C-g”组合键暂停过它的执行时才有效
(无) <i>Edit→Spell→Kill Process</i>	ispell-kill-ispell	杀死（即结束）Ispell 进程
ESC TAB <i>Edit→Spell→Complete Word</i>	ispell-complete-word	在文本模式下，自动补足当前单词

使用 UNIX 拼写检查器

Emacs 还有一个面向 UNIX 拼写检查器的接口。可以用它来检查一个单词、一个编辑缓冲区或者一个文本块。Emacs 将使用 UNIX 操作系统提供的字典，来检查单词的拼写情况。

spell-word 命令对光标位置上的单词进行拼写检查（如果光标处于两个单词的中间，就将对前面的那个进行检查）。它会先在辅助输入区里显示一条下面这样的消息：

Checking spelling of word

这条消息的含义是“正在进行拼写检查，别再做任何编辑了”。如果单词拼写无误，Emacs会告知正确；如果Emacs认为那个单词拼写有误，它会显示一条下面这样的消息：

word is not recognized; edit a replacement: word

如果报错原因只是字典里查不到它，可单词实际上是正确的话（比如人名或者专业术语），可以直接按下回车键。如果确是单词拼写错误，请用标准的Emacs编辑命令来改正。Emacs将会开始一次查询-替换操作，把这个单词的上下文显示出来；再通过查询-替换操作的键盘响应命令，告诉Emacs是否需要替换那个单词即：按空格键改正拼写错误，按“C-g”退出且不做任何修改，按“n”不修改此处并前进到下一个出现单词拼写错误的位置等等。完整的键盘响应命令请参考本章前面“查询-替换操作”小节中的速查表。请看下面这个例子，我们将用UNIX拼写检查器来改正我们的备忘录文件。

输入：ESC x spell-word RETURN

```
Buffers Files Tools Edit Search Minibuf Help
We were very anxious to recieve your order and were pleased when it
came in today.

Now that weve recieved it, we'll begin work on filling the order.
You should recieve the merchandise in about six weeks, provided that
we can get materials from the factory quickly to do the necessary
customization.
-----Emacs:ordermemo      (Text Fill)--L1--All-----
'recieve' not recognized; edit a replacement:recieve
```

在辅助输入区里改正“recieve”的拼写。完成后按回车键开始一次查询-替换操作。

改正拼写错误后按下回车键。

```
Buffers Files Tools Edit Search Help
We were very anxious to recieve your order and were pleased when it
came in today.

Now that weve recieved it, we'll begin work on filling the order.
You should recieve the merchandise in about six weeks, provided that
we can get materials from the factory quickly to do the necessary
customization.

-----Emacs:ordermemo      (Text Fill)--L1--All-----
Query replacing \recieve\ with receive: (?for help)
```

按空格键对单词进行替换并前进到下一个拼错“receive”的地方。

按下： SPACE

```
Buffers Files Tools Edit Search Help
We were very anxious to receive your order and were pleased when it
came in today.

Now that we've received it, we'll begin work on filling the order.
You should receive the merchandise in about six weeks, provided that
we can get materials from the factory quickly to do the necessary
customization.
----- Emacs:ordermemo      (Text Fill)--L4---All-----
Query replacing \receive\ with receive: (?for help)
```

继续执行，直到完成整个编辑缓冲区的拼写检查。

在使用 UNIX 拼写检查器时有两件事情需要大家特别注意：

- **spell-word**命令根据所给单词，在整个编辑缓冲区的范围内查找有无拼错的情况。它会从缓冲区的开始一直查找到缓冲区的末尾。因此，当它开始对拼错的单词（本例中是“*receive*”）进行查询 - 替换的时候，它会把光标移动到编辑缓冲区里第一次出现该单词的地方——不一定是光标原先的位置。
- **spell-word**命令只检查给定单词的拼写情况，不包括单词的各种变形（比如时态和单复数形式等）。在我们的例子里，它所查找和改正的单词是“*receive*”而不是“*received*”。不过，当你看出“*recieved*”也拼错了的时候，完全能够用递归编辑的办法来改正它。

如果想对某个编辑缓冲区进行全面的拼写检查，请输入“**ESC x spell-buffer RETURN**”。Emacs 会从编辑缓冲区的第一个单词开始查找不正确的单词。当它找到一个拼错了的单词时，会显示一条如下所示的消息：

```
word is not recognized; edit a replacement: word
```

如果该单词的拼写确实是正确的，按回车键（就像使用 **spell-word** 命令时那样），Emacs 会前进到下一个拼写出错的单词处。如果单词是拼错了，用标准的编辑命令改正它；Emacs 将开始一次查询 - 替换操作，以便把缓冲区里所有拼错这个单词的地方都改正过来。这次查询 - 替换操作完成之后，Emacs 将继续顺序检查其他的拼写错误。

还可以对一个选定的文本块进行拼写检查。把文本标记放到文本块的一头，再把光标移动到另外一头，然后发出“**ESC x spell-region RETURN**”命令。**spell-region** 命令的行为与 **spell-buffer** 命令是完全一样的。

如果确认某个单词的拼写没有错误（按了回车键），那么在这次编辑活动期间，在此编辑缓冲区里，Emacs 将不会再过问这个单词。不过，只有 **spell-buffer** 和 **spell-region** 命令才有这种“记性”，它不适用于 **spell-word** 命令。

往 UNIX 拼写检查器的字典里添加新单词

可以把任何特殊的拼写情况（比如人名等）保存到一个名为“*.spell*”的文件里。要想利用 UNIX 拼写检查器的这个特长，需要修改变量 **spell-command** 的值，这个变量给出的是 Emacs 用来进行拼写检查的 UNIX 命令。在默认的情况下，这个变量的值是“**spell**”。在“*.emacs*”文件里增加一条语句就能把变量 **spell-command** 的值改为“**spell +.spell**”，如下所示：

```
(setq spell-command 'spell "+.spell")
```

注意：在“*spell*”和加号之间必须有一个空格，在加号和“*.spell*”之间却不能有空格。保存“*.emacs*”文件，再重新进入 Emacs 使这条语句生效。接着，把个人词汇放到一个名为“*.spell*”的文件里面。这个文件必须按字母表顺序排序，并且每行只能有一个单词。这个命令要求每一个子目录都必须有它们各自的“*.spell*”文件；对大多数应用情况来看，这种做法是很适用的，因为每一个工作项目有它们各自的专用词汇表（人名、技术术语等等）。

表 3-7 对 **spell** 拼写检查器的使用命令进行了汇总。

表 3-7：UNIX 拼写检查命令速查表

键盘操作	命令名称	动作
(无)	spell-word	检查光标位置上的单词或者光标后面的单词
(无)	spell-buffer	检查当前编辑缓冲区的拼写
(无)	spell-region	检查当前文本块的拼写

单词简写模式

单词简写模式（word abbreviation mode）可为特殊的单词和短语定义一些简写形式，而那些简写词则有很多种用法。就其传统用法而言，简写模式使得不必逐字逐句地输入很长的单词或者很长的短语。假设正在起草一份合同，合同里会反复用到“National Institute of Standards and Technology”这个短语却不允许使用首字母缩写。重复输入这么长的短语确实很费事，为了能省点力气，可特意定义一个简写词“nist”。有了这个简写词定义之后，只要输入“nist”加一个空格或一个标点符号，Emacs就会把那个短语完整地插入到文本里。Emacs会密切关注内容，一旦输入了一个简写词，并按下了空格键或者某个标点符号键，它就会立刻自动地将此简写词展开为其对应的短语。

先来看一个例子，然后再介绍进入单词简写模式和自行定义简写词汇表的方法。单词简写模式有一个非传统用法是我们非常喜欢的——用它在打字的过程当中来改正拼写错误。几乎每个人都有一些会习惯性打错的单词，这也许跟人的记忆神经有关系。可以采取一种很简单办法来解决这一问题，即告诉Emacs这些拼错的单词是其正确形式的“简写词”，而Emacs会在输入这些单词的时候把它们自动改正过来；在Emacs把打错的单词改正过来之前，用户可能根本不知道自己打错了字。举个例子，假设已经进入单词简写模式，并在此前把“recieve”定义为“receive”的一个简写形式；现在，打字时又不小心打错了这个单词。如下所示：

输入： You will recieve

```
Buffers Files Tools Edit Search Help
You will recieve.

--**-Emacs:letterb      (Text Fill) --L1--A1-----
```

输入了错误的单词但还没有按下空格键。一旦按下空格键，Emacs将把它自动改正过来。

输入： SPACE the materials you requested shortly.

```
Buffers Files Tools Edit Search Help
You will receive the materials you requested shortly.

--**-Emacs:letterb      (Text Abbrev Fill) --L1--A1-----
```

按下空格键时，Emacs 将自动改正这个单词。用户不需要停止录入工作，甚至可能根本就不知道自己打错了一个单词并已经被纠正过来了。

用户不仅能为自己经常用到的短语定义一些简写词以方便工作，还可以为自己经常打错的单词定义一个简短的“简写词汇表”以节约花在文件校对上的时间并减少常见打字错误的次数。

在为短语定义简写词时，千万不要使用现有的英文单词；否则，即使不想让 Emacs 对简写词进行扩展，它也会那样做（Emacs 在对简写词进行扩展的时候不会事先提问）。举个例子，如果经常用到“World Association for Replicant Technology”这个短语，就不要把它的简写词定义为“*wart*”（英文的意思是“瘤状物”）；否则，一输入这个词就会被扩展为短语，再想用“*wart*”来描述癞蛤蟆之类的东西可就不容易了。（如果极少用到单词“*wart*”，认为使用“*wart*”作为简写词非常方便，就可以在需要输入单词“*wart*”的时候用“**ESC x unexpand-abbrev RETURN**”命令告诉 Emacs 取消这个简写词定义。）

Emacs 只会严格按照简写词的定义情况来对它进行扩展。如果已经把“*recieve*”定义为“*receive*”的一个简写词，就还得把“*recieves*”、“*recieving*”、“*recieved*”等也定义为简写词，以包括可能打错的这个单词的各种时态。

在开始定义自己的简写词汇表之前，还有一件更重要的事要记住：Emacs 会根据简写词被定义时的编辑模式对它们进行分类。全局性简写词可以用在任何一种编辑模式里；而局部性简写词则只能用在定义时所在的编辑模式里。比如，如果想让某些简写词只能用在文本模式里，而不能用在 C 语言模式里，就必须在文本模式里把它们定义为局部简写词（local）。如果想让简写词用在任何一种编辑模式里，就必须把它们定义为全局简写词（global）。记住：简写词的作用范围是编辑模式，而非文件或编辑缓冲区。

Emacs 还提供一种简写词的逆向定义方法。说它是“逆向定义方法”的原因是因为先输入的是简写词，后输入的是其定义。有些命令（书中没有对这些命令进行介绍）允许先输入其定义，后输入简写词；但必须使用一些特殊的按键组合，才能让 Emacs 知道简写词是由光标前哪些字符组成的。相比之下，逆向定义方法不仅更方便，而且无论与简写词对应的短语定义是一个单词还是十个单词，它都能应付自如。

为一次编辑工作定义临时性的简写词汇

一般来说，既然已经辛苦地定义了单词的简写形式，那么肯定会在多次的 Emacs 编辑工作中用到。可如果想先试试单词简写模式，然后再决定是否需要让它成为编辑器启动时初始化工作的一部分，那么请按以下步骤进行操作。

为编辑工作临时定义一些简写词汇：

1. 用输入“**ESC x abbrev-mode RETURN**”的办法进入单词简写模式。“Abbrev”字样将出现在状态行上。
2. 如果想定义的是全局性简写词，请在输入简写词之后按下“**C-x a i g**”组合键（命令名是 **add-inverse-global**）。如果想定义的是局部性简写词，请在输入简写词之后按下“**C-x a i l**”组合键（命令名是 **add-inverse-local**）。Emacs 将提示输入其扩展短语。
3. 输入该简写词的扩展短语后按下回车键，Emacs 对简写词进行扩展并会在此后每次输入这个简写词再跟上一个空格或标点符号的时候照章行事。定义的简写词只在此次 Emacs 编辑工作期间有效。

如果喜欢使用 Emacs 的单词简写模式，可以让它成为编辑器启动时初始化工作的一部分；具体操作步骤列在下一小节里。

为多次编辑工作定义永久性的简写词汇

当编辑工作到了离不开单词简写模式的地步时，把它加到“*.emacs*”文件里是最简单的解决方案（再也用不着每次都要回忆上一小节里列出的操作步骤了）。下面给出的操作流程将创建出一个用来保存简写词汇的永久性文件，Emacs 会在每次启动时自动加载这个文件。还可以随时从这个文件里删除简写词汇。删除简写词汇的操作步骤将在下一小节里介绍。

定义简写词汇并使它们成为编辑器启动时初始化工作的一部分：

1. 把下面这些语句添加到“*.emacs*”文件里：

```
(setq-default abbrev-mode t)
(read-abbrev-file "~/abbrev_defs")
(setq save-abbrevs t)
```

2. 保存 “*.emacs*” 文件并重新启动 Emacs。“Abbrev” 字样将出现在状态行上。还会出现一条错误信息，即 Emacs 无法加载简写词汇文件（这条信息很容易理解，因为你还没有创建这个文件）。不要理睬这条错误信息，它以后就不会再出现了。
3. 输入一个简写词，然后按下 “**C-x a -**” 或者 “**C-x a i g**” 组合键。“**C-x a i g**” 创建的是全局性简写词；如果你创建局部性简写词，请按下 “**C-x a i l**”。Emacs 于是提示输入其扩展短语。
4. 输入该简写词的扩展短语后按下回车键。Emacs 对简写词进行扩展并会在此后每次输入这个简写词再跟上一个空格或标点符号的时候照章行事。根据需要重复第 3 步和第 4 步以定义任意多个简写词。
5. 输入 “**ESC x write-abbrev-file RETURN**” 保存简写词汇文件。Emacs 提示输入一个文件名。
6. 输入 “**~/.abbrev_defs RETURN**”。Emacs 保存该文件。

上述流程的第 5 步和第 6 步只有在第一次定义简写词汇的时候才需要执行。这个文件存在以后，Emacs 会自动加载这个简写词汇文件。

在此后的编辑工作中再定义简写词汇的时候，Emacs 会询问是否想保存简写词汇文件（注 3）。回答 “y” 以保存刚才新定义的简写词汇，它们会自动生效。

删除简写词汇条目

如果经常使用简写词汇，那么可能会不断地做些改动。如果想对简写词汇表进行编辑，请输入 “**ESC x edit-abbrevs RETURN**” 命令。如果你想查看（而不是编辑）简写词汇表，请输入 “**ESC x list-abbrevs RETURN**” 命令。

在简写词汇表显示出来以后，按下 “**C-k**” 组合键（或者其他任意编辑命令）来删除不想要的简写词条目。这个词汇表的格式是由 Emacs 自己安排的，所以不要去编

注 3： Emacs 19 的某些早期版本不会询问是否想保存简写词汇文件。如果情况真是这样，就必须在每次定义了一些简写词汇之后输入 “**ESC x write-abbrev-file RETURN**” 以明确地保存自己的简写词汇。因为不可能每天都要定义一些简写词汇，所以这个要求不会有太多不方便。不管怎么说，我们认为在简写词汇上多花些功夫是值得的。

辑这个文件的文本行，或者给它添加新的文本行；删除操作大概是惟一安全的选择。下面是编辑简写词汇表文件时将会出现的简写词排列情况。这个文件会根据简写词是全局性的还是某个模式局部性的而划分为不同的区。如下所示：

```
'iwthout'      1      "without"  
'prhase'       1      "phrase"  
'teh'          1      "the"  
'fo'           1      "of"  
'eams'         2      "Emacs"  
'wrok'         1      "work"  
'aslo'         1      "also"  
'sotred'       1      "stored"  
'inforamtion' 1      "information"  
'esc'          6      "ESC"  
'thaht'        1      "that"  
'chatper'      1      "chapter"  
'adn'          1      "and"  
'iwth'          1      "with"  
'chpater'     1      "chapter"  
'tex'           36     'TeX'  
'loaction'     1      "location"  
'wart'          1      "World Association for Replicant Technology"
```

列在第一栏里的是简写词；上例中的简写词基本上都是一些容易拼错的单词。第二栏供内部记录使用，不用管它。第三栏给出了简写词的定义情况，Emacs 在看到简写词时会用与之对应的单词或短语进行替换。

删除简写词汇条目的方法是：先把光标移动到定义该条目的文本行上，用“C-k”组合键把整个文本行删掉，最后输入“**ESC x write-abbrev-file RETURN**”保存这个文件。按下“**C-x b**”将返回此前正在编辑的编辑缓冲区（这是一个多编辑缓冲区操作命令，我们将在第四章里介绍它。）

禁用简写词汇

以下两种办法中的任何一种都可以用来完全禁用简写词汇。第一种办法是输入“**ESC x kill-all-abbrevs RETURN**”命令，这个命令的作用是在本次编辑工作中禁用简写词汇。

第二种办法是删除保存简写词汇表的文件。如果简写词汇是编辑器启动时初始化工作的一部分，就要从“**.emacs**”文件里删除“**read-abbrev-file**”那一行。

简写词汇与字母的大小写

Emacs 对简写词所做的字母大小写设置，通常都会让人称心如意。但如果对简写词的字母大小写有特殊的要求，那么多知道些内幕往往会有许多好处。请看下面这些规则：

- 只要简写词的短语定义里有大写字母，Emacs 在插入这个短语定义的时候就不会对它做任何改动。比如说，如果把 “ora” 定义为 “O'Reilly & Associates”的一个简写词，那么 “O'Reilly & Associates”的字母大小写情况就将保持原样。
- 如果简写词的短语定义全都是小写字母，Emacs 将根据以下规则对它进行字母大小写设置：
 - 如果把简写词的全部字母都输入为小写，Emacs 就将以小写字母插入其短语定义。
 - 只要输入的简写词里有大写字母，Emacs 就将把其短语定义的第一个单词的第一个字母设置为大写。
 - 如果把简写词的全部字母都输入为大写，并且变量 “**abbrev-all-caps**” 设置为 “nil”，Emacs 就将把其短语定义的每一个单词的第一个字母设置为大写；如果变量 “**abbrev-all-caps**” 设置为 “t”，Emacs 就将把其短语定义的每一个单词的每一个字母设置为大写。

表 3-8 给出了一些例子。

表 3-8：单词简写情况示例

简写词	短语定义	输入	扩展为	说明
lc	lamb chop	lc	lamb chop	“lc” 是小写，所以 “lamb chop” 是小写
lc	lamb chop	Lc	Lamb chop	“Lc” 里有一个大写字母，所以 “Lamb” 的首字母大写
lc	lamb chop	IC	Lamb chop	“IC” 里有一个大写字母，所以 “Lamb”的首字母大写

表 3-8：单词简写情况示例（续）

简写词	短语定义	输入	扩展为	说明
lc	lamb chop	LC	Lamb Chop	“LC” 全都是大写，所以两个单词的首字母都大写
lc	Lamb Chop	lc	Lamb Chop	短语定义里的字母大小写情况永远不变
lc	Lamb Chop	LC	Lamb Chop	短语定义里的字母大小写情况永远不变

要想记住这些规则还真得下点功夫，但用不着死记硬背。Emacs 通常能够很好地完成字母大小写方面的处理工作；只要做点练习，就能掌握 Emacs 的行为规律。

表 3-9 对与简写词有关的操作命令进行了汇总。

表 3-9：简写词编辑命令速查表

键盘操作	命令名称	动作
(无)	abbrev-mode	进入（或退出）单词简写模式
C-x a - 或 C-x a i g	inverse-add-global-abbrev	输入全局性简写词之后，输入其短语定义
C-x a i l	inverse-add-mode-abbrev	输入局部性简写词之后，输入其短语定义
(无)	unexpand-abbrev	撤销最近一个简写词定义条目
(无)	write-abbrev-file	保存简写词汇表文件
(无)	edit-abbrevs	编辑简写词汇表
(无)	list-abbrevs	查看简写词汇表
(无)	kill-all-abbrevs	本次编辑工作禁用简写词功能

疑难解答

- 对一个能够从屏幕画面上看到的字符串进行查找，可 Emacs 就是找不到。最可能的解释是 Emacs 会考虑换行符和标点符号，而你可能没有把它们包括在查找字符串里。请使用单词查找操作，它在对字符串进行查找的时候，会忽略换行符的标点符号。

- **Ispell 工作不正常。**造成这种情况的原因有很多。有时候，**ispell-region** 命令可以工作而其他命令却不行。请用“**C-x h**”把整个编辑缓冲区标记为一个文本块，然后输入“**ESC x ispell-region**”命令。如果这条路走不通，请继续往下看。
- **看到一条错误信息 “Problem starting Ispell; use old-style spell instead? (启动 Ispell 时故障：是否使用老式的 spell?)”。**很可能是 Ispell 没有安装。如果回答“**y**”，Emacs 将启动 UNIX 拼写检查器 **spell** 程序。与 **spell** 程序有关的讨论请参考“使用 UNIX 拼写检查器”一节。
- **看到一条错误信息 “ispell version 3.1.* is required: try renaming ispell4.el ispell.el (要求使用 ispell 3.1.*，请试试把 ispell4.el 更名为 ispell.el)。”**在 Emacs 19 里使用了 Ispell Version 4。Emacs 19 需要的是 Ispell Version 3。请让系统管理员把 Ispell 降级为 Ispell Version 3（也可以首先建议系统管理员把文件 *ispell4.el* 更名为 *ispell.el*）。
- 在使用 **spell** 拼写检查器时，看到一条错误信息“**Searching for program: no such file or directory, spell**（查找程序：没有文件或子目录，**spell**）”。有些 UNIX 系统（特别是 Linux 发行版本）会用 Ispell 替代 UNIX 的 **spell** 拼写检查器程序。如果情况如此，则只能使用 Ispell 或者找个有 **spell** 软件的人把它安装在自己的系统上。
- 在使用 **Ispell-buffer** 的时候，看到一条错误信息“**spell can't read .spell (spell 无法读出 ".spell" 文件)**”。需要在文件所在的子目录里创建一个“**.spell**”文件作为拼写检查文件。
- 定义了一些简写词，可 Emacs 无法保存它们。如果没有在自己的“**.emacs**”文件里进行设置，Emacs 在默认情况下是不会保存简写词汇的。请参考本章中“为多次编辑工作定义永久性的简写词汇”一节里的讨论。如果已经在自己的“**.emacs**”文件里进行了设置却还是没有办法保存简写词汇，那么使用的可能是 Emacs 19 的早期版本，它们的单词简写模式有一个小 bug。只能明确地在自己每次定义了一些简写词汇之后输入“**ESC x write-abbrev-file**”以保存它们。

第四章

使用编辑 缓冲区和窗口

本章内容：

- 文件、编辑缓冲区和窗口
- 同时使用多个编辑缓冲区进行工作
- 使用窗口进行编辑
- 在文档中使用书签
- 临时性地挂起 Emacs
- 使用多个 X 窗口进行编辑

Emacs最有用的功能之一是它能够同时对多个编辑缓冲区进行编辑，并利用窗口同时显示一个以上的编辑缓冲区。有关的操作命令并不复杂，只需掌握有限的几个命令就可以大大提高工作效率。使用多编辑缓冲区和多窗口的次数越多，就越能发现它更多的用途。

在这一章里，我们将介绍编辑缓冲区和窗口的区别，与多编辑缓冲区有关的操作，以及如何利用窗口来显示多个编辑缓冲区。接着，我们将继续讨论如何对编辑缓冲区列表清单进行操作（这是管理多编辑缓冲区的一个简便方法），以及如何利用书签保持正在处理的各种文件里的位置。有时候，可能需要临时性地把 Emacs 挂起来。确实可以这样做，即使正使用多个窗口和编辑缓冲区时也完全没有问题，我们会介绍如何操作。最后，我们在本章的末尾专门为 X 用户准备了一节，介绍如何在 Emacs 编辑工作中使用多个 X 窗口的内容。

文件、编辑缓冲区和窗口

Emacs 里的一切编辑操作都发生在编辑缓冲区 (buffer) 里。虽然编辑缓冲区往往会有许多其他方面的用途，但它通常就是某个文件的工作副本。如果你正在编辑的编辑缓冲区里包含着某个文件的一个副本，那么当你保存自己所做的改动时，Emacs 将把编辑缓冲区的内容复制到文件里。在实际工作中，在对编辑缓冲区进行完编辑

之后，你完全可以不保存自己所做的修改。从另一方面讲，在一次编辑工作中，想把编辑缓冲区里的改动保存到文件里多少次也都可以。如果所做的编辑修改确实非常 important，频繁存盘还是一个很好的习惯呢。

除了工作在文件副本上的编辑缓冲区以外，你还会接触到一些只能用做临时工作区域的编辑缓冲区，这些编辑缓冲区与文件一般没有什么关联。这些临时性的工作区域很适合用来做练习或打草稿。可以随时使用 **write-file** 命令 (“**C-x C-w**”) 或 **save-buffer** 命令 (“**C-x C-s**”) 来保存这类临时性编辑缓冲区里的内容。如果试图保存的编辑缓冲区与任何文件都没有关联，“**C-x C-s**” 命令会要求提供一个文件名——它会在辅助输入区里显示提示信息 “File to save in:(把文件保存为:)”，后面跟着一个默认的子目录名。

可以不受限制地打开任意多个编辑缓冲区。在大部分时间里，同时显示在屏幕上的编辑缓冲区也就是一两个；但尽管看不到它们，在一次 Emacs 编辑工作中创建的编辑缓冲区也仍然是活动的。可以把它们想像为一摞纸，而显示在屏幕上的是最顶上的那张。可以随时翻看另外一张纸（即转到另一个编辑缓冲区），也可以随时给这摞纸再添上一张（即创建一个新的编辑缓冲区）。

特殊用途编辑缓冲区的创建工作由 Emacs 负责。这些内部编辑缓冲区的名字通常采用 “*buffer name*” 的格式。“*Help*”、“*scratch*”、“*mail*” 和 “*Buffer List*” 等几个编辑缓冲区就是由 Emacs 自己创建的。

每个编辑缓冲区都有一个相关的主编辑模式，这个主模式左右着 Emacs 在这个编辑缓冲区里的大部分行为。比如，用来书写英文文本的文本模式，与用来编写 LISP 程序代码的 LISP 模式这两者的行为是不一样的。

那窗口又是些什么东西？窗口是显示器屏幕上用来显示编辑缓冲区内容的区域（注 1）。换句话说，需通过一个窗口来查看一个编辑缓冲区。可能没有想过自己的显示器屏幕就是一个大窗口，但实际情况却是如此。每个窗口都有一个相关状态

注 1： 这是 Emacs 对“窗口”的定义，它与 X 窗口系统和微软公司的 Windows 中的图形化用户界面意义上的“窗口”是不同的概念。在这些用户界面上，窗口是屏幕上一些有程序运行在其中的区域。为了与 Emacs 窗口有所区别，我们将把 X 窗口系统下的多窗口称为“窗格 (frame)”。前者指的是 Emacs 屏幕画面上的区域，而后者则是图形化用户界面意义上的窗口。

行，并且通常也只有一个。从理论上讲，可以在屏幕上不受限制地拥有任意多个不同的窗口。从一个窗口转移到另一个窗口，就能做到同时对多个文件进行编辑、对比两个窗口里的文件、从一个文件复制到另一个文件等等。

虽然窗口和编辑缓冲区是两个很容易混淆的概念，可是两者之间的差异并不复杂。我们已经讲过，编辑缓冲区是文件的一个工作副本；而窗口则是显示器屏幕上的一个区域。可以在屏幕上让好几个窗口显示同一个编辑缓冲区——比如想查看同一文件的两个不同位置、或者想把一个地方的内容复制到另一个地方去的时候。比较常见的情况是在屏幕上用不同的窗口来代表不同的文件；或者是用一个窗口来显示某个文件的编辑缓冲区，用另一个窗口来显示Emacs某个内部的编辑缓冲区——内部编辑缓冲区将在本章后半部分探讨。随着大家对本章学习的深入，文件、编辑缓冲区和窗口之间的区别将越来越直观。

同时使用多个编辑缓冲区进行工作

输入“`emacs filename`”命令就能进入 Emacs 是大家都已经知道的事情了。如果还想再创建一个包含着另一文件内容的编辑缓冲区，只需按下“`C-x C-f`”组合键来查找文件。Emacs 将自动创建出第二个编辑缓冲区并转移至此。如果已经在某个编辑缓冲区里打开过这个文件的一个副本，“`C-x C-f`”命令将进到那个现有的编辑缓冲区里去——这种情况可能正是想要的；如果“`C-x C-f`”每次都从磁盘上读出文件，那么文件就将形成许多个版本，而各版本之间又几乎没有区别。如果给“`C-x C-f`”提供的文件名不存在，Emacs 就会认为要用那个名字创建一个新文件，它会进到一个空白的编辑缓冲区里。（如果在输入文件名时打错了字，按下“`C-x C-v`”组合键将把你带回原来的文件。）

“`C-x C-f`”后面永远要跟有一个文件名；而在编辑缓冲区之间进行切换的“`C-x b`”命令的后面要跟着一个编辑缓冲区名。是否注意到状态行上只显示有编辑缓冲区的名字而没有文件名？有些版本的 Emacs 会把这两个名字都显示出来，但 GNU Emacs 只显示编辑缓冲区名。如果没有对它们进行过修改（请参考本章后面“编辑缓冲区的重新命名”小节），那么编辑缓冲区名和文件名（如果有的话）就将是一样的。

可以用“`C-x b`”来完成以下几件事：

如果在“C-x b”的后面输入了	Emacs 将
一个新的编辑缓冲区名	新创建一个与任何文件都没有联系的编辑缓冲区并转移到那里
一个已有的编辑缓冲区名	进入那个编辑缓冲区（不论该编辑缓冲区是否与某个文件相关联）

在编辑缓冲区之间进行切换要按下“C-x b”组合键。Emacs会显示一个默认的编辑缓冲区名。如果它就是想去的编辑缓冲区，直接按下回车键；否则请输入正确的编辑缓冲区名的前几个字母再按下TAB键，Emacs将自动补足这个名字的剩余部分。现在，按下回车键就能转移到那个编辑缓冲区。

如果想创建第二个（或者第三个、第四个等等）空白的编辑缓冲区，按下“C-x b”组合键，Emacs将要求提供一个编辑缓冲区名。可以随意起个名字，比如parctice，然后按下回车键；Emacs将创建这个编辑缓冲区并进入其中。举个例子，假设在dickens编辑缓冲区里工作时忽然想起了点东西，决定打开一个新的编辑缓冲区来编辑James Joyce（英国诗人乔叟）的作品。如下所示：

初始状态：正在dickens编辑缓冲区里编辑狄更斯的作品。

按下：C-x b

```

Buffers Files Tools Edit Search Minibuf Help
It was the best of times, it was the worst of times, it was the age of
wisdom, it was the age of foolishness it was the epoch of belief, it
was the epoch of incredulity, it was the season of Light, it was the
season of Darkness, it was the spring of hope, it was the winter of
despair, we had everything before us, we had nothing before us, we
were all going direct to Heaven, we were all going direct the other
way—in short, the period was so far like the present period, that
some of its noisiest authorities insisted on its being received, for
good or evil, in the superlative degree of comparison only.

----- Emacs:dickens      (Text File)--L1--All -----
Switch to buffer:(default *scratch*) 

```

准备输入一个新的编辑缓冲区名。



输入: joyce RETURN



现在, 可以往这个名为 joyce 的新编辑缓冲区里输入东西了。

上述操作与使用 “C-x C-f” (命令名是 **find-file**) 组合键的情况并没有什么不同之处, 两者惟一的区别是新编辑缓冲区 joyce 现在还没有与某个文件关联起来。因此, 如果现在就退出 Emacs, 编辑器不会询问你是否想保存它。

“C-x b” 特别适用于没有记住正在编辑的文件的名字的场合。假设正在编辑一个名字非常难记的文件——比如 /tmp/80b16.12344 文件 (注 2); 不小心按了某个键使这个编辑缓冲区从屏幕上消失了(这类命令我们马上就会在本章后面讲到)。怎样才能让 /tmp/80b16.12344 文件重新回到屏幕上呢? 记住这个文件的全名或者哪怕是文件名的一部分了吗? 没记住? 别着急, 在没有进行其他操作之前, 立刻按下 “C-x b” 组合键。默认的编辑缓冲区恰好是刚才消失的那个; 按下回车键, 它又会重新出现。

还有一种办法是选择 “Buffer (编辑缓冲区)” 菜单, 所有的编辑缓冲区都列在其 中, 从中挑选一个就行了。如果看到一些不是自己亲自创建的编辑缓冲区, 可别大惊小怪——Emacs 也会创建编辑缓冲区, 这类编辑缓冲区的名字多是像 “*Help*” 和 “*scratch*” 的形式。

多个编辑缓冲区的保存

大家已经知道要想一个一个地保存编辑缓冲区要按下 “C-x C-s” 组合键。如果正在使用多个编辑缓冲区, 就该知道按下 “C-x s” (命令名是 **save-some-buffers**) 组合键可把它们都保存起来。Emacs 会依次询问是否想保存每一个编辑缓冲区; 回答

注 2: 这个文件名还不算最古怪的。许多程序创建的临时文件都有类似于这样的名字。你可以想像自己正在调试一个这样的程序。

“**y**”保存，回答“**n**”不保存。按下“**!**”键将使 Emacs 无条件地保存所有的编辑缓冲区。如果只想保存某个编辑缓冲区，其他的都不要，请输入一个英文句号(.)。如果想取消这个命令并且不保存当前的编辑缓冲区，按下“**q**”键（在按下“**q**”之前保存的编辑缓冲区都已经保存起来了；**ESC** 对那些文件没有影响）。在决定是否需要保存某个编辑缓冲区之前，如果想先看看它里面的内容，请按下“**C-r**”组合键；Emacs 将进入查看模式（view mode）——这个模式里的编辑缓冲区只能看，不能改。按下“**C-c**”组合键退出查看模式并回到保存编辑缓冲区的操作上来。

编辑缓冲区的删除

编辑缓冲区创建起来很容易，想删除它们也不难。如果觉得 Emacs 因为有太多的编辑缓冲区而变得磕磕绊绊，就会要删除一些编辑缓冲区。开始工作时也许使用了 5 个编辑缓冲区，可现在想换成另外 5 个继续工作；把前 5 个编辑缓冲区都删掉肯定会使工作更顺手一些。删除编辑缓冲区也是一种有用的应急措施——比如某些替换操作引起灾难性后果的时候，可以先删除那个编辑缓冲区并选择不保存所做的修改，然后再重新读取文件。

删除一个编辑缓冲区不会删除它所关联着的文件，也不等同于隐藏一个缓冲区的操作。被隐藏起来的编辑缓冲区依然是活动的，而被删除的编辑缓冲区则将不再是本次 Emacs 工作的一部分。还用一摞纸为例，删除一个编辑缓冲区等于从这摞纸里把一张纸抽出来扔掉。

删除编辑缓冲区也不会让用户冒丢失编辑修改的风险。如果对编辑缓冲区进行过修改（并且这个编辑缓冲区代表着某个文件），Emacs 会询问是否需要在删除这个编辑缓冲区之前保存所做的修改。虽然对与文件没有关联的编辑缓冲区所做的修改将会丢失，但可能这些编辑缓冲区的命运无关紧要。

删除一个编辑缓冲区的方法是按下“**C-x k**”（命令名是 **kill-buffer**）组合键或者从“File (文件)”菜单里选择执行“Kill Current Buffer (删除当前编辑缓冲区)”操作。Emacs 会给出当前显示的编辑缓冲区的名字，按下回车键就可以删除它。如果 Emacs 给出的编辑缓冲区名不是想删除的，那么可以输入另外一个编辑缓冲区的名字再按下回车键。如果准备删除的编辑缓冲区里还有一些没来得及保存的修改，Emacs 会显示一条如下所示的消息：

```
Buffer buffer name modified. Kill anyway? (yes or no)
```

如果不想保存所做的修改，回答“**yes**”，Emacs 将删除这个编辑缓冲区。如果想停止编辑缓冲区的删除操作，回答“**no**”。此后，可以先用“**C-x C-s**”组合键保存这个编辑缓冲区，再用“**C-x k**”组合键删除它。

还可以让 Emacs 就准备删除的每一个编辑缓冲区进行提问，从而有机会一个一个地决定是否要删除它们。如果想用这种办法来删除已经不再需要的编辑缓冲区，请输入“**ESC x kill-some-buffers**”。Emacs 将依次列出每个编辑缓冲区的名字，并指出它里面有没有还未来得及存盘的修改，然后询问是否真的想删除它。

编辑缓冲区的重新命名

编辑一个文件的时候，编辑缓冲区的名字就采用文件的名字。如果觉得长文件名在使用中不方便，可以给编辑缓冲区重新起个名字（这种重命名只改变编辑缓冲区的名字，对文件名不会有影响）。这个功能最适用于某些自动补足功能不太完备的 Emacs 版本。GNU Emacs 的自动补足功能是很不错的：当需要输入编辑缓冲区的名字时，只要输入其前几个字母再按下 **TAB** 键，Emacs 就会自动补足那个名字。但肯定会有你想改变编辑缓冲区名字的时候。

重新命名编辑缓冲区的方法是：输入“**ESC x rename-buffer**”，Emacs 提示输入新名字；输入它，再按回车键。新名字将出现在状态行上。

前面讲过，GNU Emacs 的状态行上只显示编辑缓冲区的名字，而不是一个编辑缓冲区名和一个文件名。即使给包含着文件的编辑缓冲区重新命名，Emacs 也不会忘记编辑缓冲区和文件之间的对应关系；保存文件（“**C-x C-s**”）或查看编辑缓冲区清单（参见本章后面的内容）试试，就会明白了。

可是若有两个同名的编辑缓冲区该怎么办呢？比如，正编辑主目录里一个名为 *memo* 的文件和自己某个子目录里一个名字也为 *memo* 的文件的情况。虽然两个编辑缓冲区都叫做 *memo*，但 Emacs 会给第二个编辑缓冲区的名字后面加上一个“<2>”以区别它们。用户可以用查看编辑缓冲区清单的办法来区分它们，查看编辑缓冲区清单的操作在本章后面就要讲到。下面是编辑缓冲区重新命名操作一种很有用的用法：如果想把哪个文件来自哪个子目录也表示出来，可以在修改编辑缓冲区的名字

时把它们的子目录名也包括在其名字里，比如像“home:memo”和“staff:memo”的样式。

注意：如果有好几个编辑缓冲区的名字是“memo”、“memo<2>”、“memo<3>”的样式，就很可能是在给出文件名时弄错了它们的目录名。如果读取文件的时候弄错了目录，Emacs 会认为想开始编辑一个新文件。举个例子，假设想编辑的文件是“~/work/memo”，但错误地输入成了“~/novel/memo”。因为“~/novel/memo”原来并不存在，所以 Emacs 会新创建一个空白的“memo”编辑缓冲区。如果改正了这个错误（“**C-x C-f ~/work/memo**”），Emacs 也将相应地重新命名编辑缓冲区：那个空白的编辑缓冲区“memo”与“~/novel/memo”文件关联，而真正想要的编辑缓冲区的名字将是“memo<2>”。

弄错文件的名字是一个很常见的错误，用下面这个办法可以解决它。如果发现“**C-x C-f**”命令找到的文件不对，可以使用“**C-x C-v**”命令，用自己想要的文件来替换它。“**C-x C-v**”会把查找到的文件放到当前编辑缓冲区里，而不是去创建一个新的编辑缓冲区。它的含义是“把真正想要的文件找出来并要它替換现在这个”。这个命令可以解决不必要地出现了大量带编号的编辑缓冲区（比如“memo”、“memo<2>”等）的问题。

只读编辑缓冲区

在工作中，可能需要查阅一些并不想对其进行修改的文件——只想浏览它们的内容。但因误碰键盘而造成意外修改的情况很容易发生。我们介绍过几种恢复原始文件的办法，可要是能够从根本上预防类似事故的发生岂不是更好？该怎么办呢？

可以用“**C-x C-q**”组合键把编辑缓冲区设置为只读属性。找个编辑缓冲区试试，状态行左边会出现两个百分号（%），它们就位于在表示编辑缓冲区已被修改的星号（**）位置处。这两个百分号表明这个编辑缓冲区是只读的。如果试图对一个只读的编辑缓冲区进行输入，Emacs 将发出蜂鸣并在辅助输入区里显示一条出错信息“Buffer is read-only (编辑缓冲区是只读的)”。如果改变主意，又想对只读编辑缓冲区进行编辑该怎么办呢？好办，再按一次“**C-x C-q**”组合键即可。这个命令的作用是切换编辑缓冲区的只读状态——也就是说，反复按下“**C-x C-q**”组合键将使编辑缓冲区交替进入只读和读写状态。



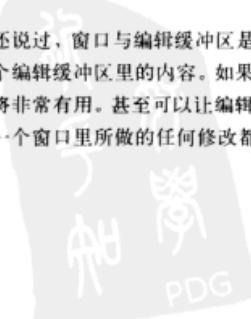
记住：切换只读状态不会改变UNIX文件的权限。如果正在编辑的编辑缓冲区里包含的是别人的文件，“**C-x C-q**”组合键将无法改变其只读状态。编辑别人文件的方法之一是先用**write-file**命令给自己复制一份它的副本，然后再对副本进行修改。如果想对属于别人的电话号码簿文档进行修改，应该先读入这个文件，然后用“**C-x C-w**”组合键把这个文件存盘一次，再用“**C-x C-q**”组合键把它从只读状态改为读写状态。以上操作不会改变原来的电话号码簿文档中的内容；但会得到一份能够随意修改的副本。如果想把某个只读文件里的一小部分内容复制到另外一个文件里，应该先标记好文本块，然后用“**ESC w**”组合键复制它，再移动到要插入这些文本的地方，按下“**C-y**”组合键粘贴它。

使用窗口进行编辑

在X窗口系统出现以前，许多人都把Emacs用做惟一的窗口化系统。如今，X和Emacs已经集成在一起了（在X下运行Emacs的有关内容请参考第十四章），但尽管如此，还是有很多人坚持使用Emacs的窗口功能，甚至在X下也不改初衷——他们看中的是Emacs窗口能够完成许多工作并大大提高工作效率，而不是计较Emacs在具体做法上与X的不同。不管是否使用X，了解Emacs窗口的使用方法都是很有必要的。

前面已经讲过，窗口是屏幕上的一些区域，而Emacs会把正在编辑的编辑缓冲区显示在这些窗口里。可以同时在屏幕上打开许多个窗口，每个窗口显示不同的编辑缓冲区。但打开的窗口越多，每个窗口的面积也就越小——Emacs窗口不能重叠，这是它与X窗口的一个不同之处。随着打开的窗口数目的增加，原有窗口的尺寸将会缩小。屏幕就像是一块馅饼，可以把它分成许多份，但分的份数越多，它们也就越小。可以让窗口左右排列、上下排列，或者混合排列。每个窗口都有它自己的状态行，上面给出了编辑缓冲区的名字、所处的编辑模式、所在编辑缓冲区里的前后位置等。为了表明窗口之间的边界，状态行通常呈反显状态。

我们还说过，窗口与编辑缓冲区是两回事。事实上，可以用一个以上的窗口来显示同一个编辑缓冲区里的内容。如果想同时查看某个大文件不同位置上的内容，这个功能将非常有用。甚至可以让编辑缓冲区的同一块内容同时显示在两个窗口里，在其中一个窗口里所做的任何修改都将出现在另一个窗口里。



想对文本进行标记、剪切和粘贴等操作的时候，编辑缓冲区和窗口之间的区别就变得非常重要了。文本标记是与编辑缓冲区而不是窗口关联的，每个编辑缓冲区只能有一个文本标记。如果在同一编辑缓冲区上的另外一个窗口里又设置了文本标记，Emacs会把文本标记移到新的位置上，在前一个窗口里设置的文本标记将不复存在。

再来说说光标的事。无论何时，屏幕上只能有一个光标；也就是说，无论何时，都只能待在一个窗口里。虽然只有一个光标，可每个窗口都能记住其中的当前编辑位置——也就是说，如果把光标从一个窗口移到另外一个窗口并做了一些编辑工作，那么当再次回到原来的窗口里时，仍会回到刚才离开的位置。在窗口里的当前编辑位置（不管光标是否在这个窗口里）叫做“插入点（point）”。每个窗口都有自己的插入点。插入点和光标这两个术语可以互换使用——但我们将尽量做到对它们有所区分。

编辑缓冲区和窗口之间的区别听起来很混乱，但在实际工作中却很容易把它们区分开。在使用多个窗口对同一个编辑缓冲区进行编辑方面，我们有着多年的经验，至今还没有犯过糊涂。

创建上下排列的窗口

如果想把当前窗口以水平方式分割为两个，请按下“**C-x 2**”组合键。重复这个操作将分割出更多的窗口。如下所示：

初始状态：



```
Buffers Files Tools Edit Search Help
It was the best of times, it was the worst of times, it was the age of
wisdom, it was the age of foolishness it was the epoch of belief, it
was the epoch of incredulity, it was the season of Light, it was the
season of Darkness, it was the spring of hope, it was the winter of
despair, we had everything before us, we had nothing before us, we
were all going direct to Heaven, we were all going direct the other
way—in short, the period was so far like the present period, that
some of its noisiest authorities insisted on its being received, for
good or evil, in the superlative degree of comparison only.

----- Emacs:dickens (Text Fill)--L1--All -----
```

按下：C-x 2

```
Buffers Files Tools Edit Search Help
It was the best of times,it was the worst of times,it was the age of
wisdom,it was the age of foolishness it was the epoch of belief,it
was the epoch of incredulity,it was the season of Light,it was the
season of Darkness,it was the spring of hope,it was the winter of
despair.we had everything before us,we had nothing before us,we
-----Emacs:dickens      (Text Fill)--L1--A1-----
It was the best of times,it was the worst of times,it was the age of
wisdom,it was the age of foolishness it was the epoch of belief,it
was the epoch of incredulity,it was the season of Light,it was the
season of Darkness,it was the spring of hope,it was the winter of
despair.we had everything before us,we had nothing before us,we
-----Emacs:dickens      (Text Fill)--L1--A1-----
```

屏幕被分割为两个水平窗口；每个窗口都有自己的状态行。

可以让Emacs在开始编辑工作时就把窗口安排好。如果想用水平窗口同时对两个文件进行编辑，可以在启动Emacs时同时给出这两个文件的名字。比如，如果想编辑dickens和joyce这两个文件，就应输入“emacs dickens joyce”；Emacs将把这两个文件分别显示在两个水平窗口里。如果想打开两个以上的文件，Emacs将显示两个水平窗口：在一个窗口里显示着某个文件，在另一个窗口里给出一份编辑缓冲区的清单。

创建左右排列的窗口

如果想把窗口垂直地分割为两个相邻的窗口，请按下“C-x 3”组合键。重复这个操作将分割出更多相邻的窗口。如下所示：

按下：C-x 3

```
Buffers Files Tools Edit Search Help
It was the best of times,it was the w$|It was the best of times,it was $| wisdom,it was the age of foolishness $|wisdom,it was the age of foolish$| was the epoch of incredulity,it was t$|was the epoch of incredulity,it $| season of Darkness,it was the spring $|season of Darkness,it was the sp$| despair,we had everything before us,$|despair,we had everything before$|
-----Emacs:dickens      (Text Fill)---Emacs:dickens (Text Fill)
It was the best of times,it was the worst of times,it was the age of
wisdom,it was the age of foolishness it was the epoch of belief,it
was the epoch of incredulity,it was the season of Light,it was the
season of Darkness,it was the spring of hope,it was the winter of
despair.we had everything before us,we had nothing before us,we
-----Emacs:dickens      (Text Fill)--L1--A1-----
C-x 3
```

这个操作将把屏幕分割为两个垂直窗口。



对窗口进行了垂直分割之后，屏显画面的宽度通常就不够Emacs显示一个完整的文本行。因为垂直窗口一般显示不出完整的文本行（行尾的美元符号“\$”表示这一行没有被全部显示出来），所以大家必须知道怎样才能让文本做左、右滚动。

如果想把当前显示的文本向左推（好让自己看到右边的文本），请按下“**C-x <**”组合键。如果想把当前显示的文本向右推（这样就能看到左边的文本），请按下“**C-x >**”组合键。不管是发生在窗口里，只要有文本行超出了屏幕画面的宽度，就可以使用这两个命令（特别是在关闭了自动换行功能的场合）。

在窗口之间移动

按下“**C-x o**”组合键（这个命令里的字母“o”代表“other”，中文意思是“另一个”）就可以从一个窗口移动到另一个窗口。Emacs在窗口之间的移动情况如图4-1所示。如果屏幕上显示有两个以上的窗口，Emacs将沿顺时针方向移动到下一个窗口去。由于Emacs没有提供移动到指定窗口去的方法，因此，如果屏幕上显示有两个以上的窗口，那么可能要按几次“**C-x o**”组合键才能到达想去的窗口。不过，可以用“**ESC n C-x o**”（“n”是一个数字）命令沿顺时针方向一次移动n个窗口。

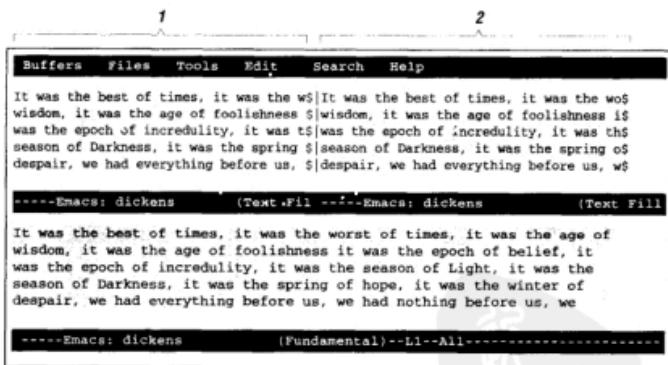


图 4-1：按顺时针方向在窗口之间移动

现在，既能创建出两个窗口，又知道了如何在它们之间移动，还能做些什么？回答是任何事情。看看下面的例子。首先，打开两个dickens窗口，两个窗口呈水平排列。刚开始时，这两个窗口里显示的是同一个文件。

按下：C-x 2

```
Buffers Files Tools Edit Search Help
It was the best of times,it was the worst of times,it was the age of
wisdom,it was the age of foolishness,it was the epoch of belief,it
was the epoch of incredulity,it was the season of Light,it was the
season of Darkness,it was the spring of hope,it was the winter of
despair,we had everything before us.we had nothing before us.we
-----Emacs:dickens      (Text Fill)--L1--Top-----
It was the best of times,it was the worst of times,it was the age of
wisdom,it was the age of foolishness,it was the epoch of belief,it
was the epoch of incredulity,it was the season of Light,it was the
season of Darkness,it was the spring of hope,it was the winter of
despair,we had everything before us.we had nothing before us.we
-----Emacs:dickens      (Text Fill)--L1--Top-----
```

在dickens上打开了两个窗口。

可以在任意一个窗口里执行任何编辑命令。在一个窗口里的前后移动不会对另一个窗口产生任何影响。下面来看看编辑另外一个文件时会发生什么事。

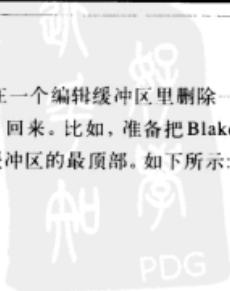
输入：C-x C-f blake

```
Buffers Files Tools Edit Search Help
Mock on Mock on Voltaire Rousseau
Mock on Mock on tis all in vain
You throw the sand against the wind
And the wind blows it back again.

-----Emacs:blake      (Text Fill)--L1--All-----
It was the best of times,it was the worst of times,it was the age of
wisdom,it was the age of foolishness it was the epoch of belief,it
was the epoch of incredulity,it was the season of Light,it was the
season of Darkness,it was the spring of hope,it was the winter of
despair,we had everything before us.we had nothing before us.we
-----Emacs:dickens      (Text Fill)--Top-----
```

现在有两个窗口、两个编辑缓冲区和两个文件。

利用“C-x o”组合键可以依次编辑这两个文件。可以在一个编辑缓冲区里删除一段文字，然后在另一个编辑缓冲区里把它恢复（即粘贴）回来。比如，准备把Blake（一位英国诗人）的诗歌的第一行移动到dickens编辑缓冲区的最顶部。如下所示：



输入： C-k C-x o ESC < C-y RETURN

Blake 的诗歌中的文字被复制到 dickens 编辑缓冲区里。

能够在不同的窗口里对不同的编辑缓冲区进行编辑是一种很有用的功能，特别是想把一个文件的内容复制到另外一个文件里，或者想一边查阅着某个内容为参考资料的文件，一边编辑着另一个文件的时候。程序员经常需要同时查看好几个文件，比如同时查看一个头文件和一个源代码文件，或者同时查看某个函数调用点和那个被调用的例程等。一旦熟悉并掌握了在不同窗口之间进行移动的命令，就会经常在屏幕上同时打开两个、三个甚至四个窗口进行工作。

窗口的删除

删除一个窗口只意味着它将不再出现在屏幕上，而它里面的任何内容和没有存盘的任何修改都不会丢失；与之对应的编辑缓冲区也还在那里，用“C-x b”组合键就能切换过去。如果想删除所在的窗口，请按下“C-x 0”（数字0）组合键。如果想把其他的窗口都删除，只保留自己正在其中进行工作的窗口，请按下“C-x 1”（数字1）组合键——意思是“把当前窗口作为我唯一的窗口”；或者从“Files”菜单里选择执行“One Window（一个窗口）”操作。保留下来的窗口将“膨胀”到填满剩余的空间。还可以用“ESC x delete-windows-on RETURN *buffername* RETURN”命令来删除某个特定编辑缓冲区上的全部窗口。

调整窗口的大小尺寸

Emacs 在对窗口进行分割时，永远会把它分割成同样大小的两块。这样的分割在多

数情况下会正好满足要求，但有时候却未必如此，特别是在打开了很多窗口的时候。如果屏幕上同时有四个、五个甚至六个窗口，适当控制各个窗口的大小就将变得很重要。否则，最感兴趣的窗口可能会变得非常之小；而当只能看到文件的五、六行内容时，想对它进行有效的编辑就几乎是不可能的。如果想加高正在其中工作的窗口，请按下“**C-x ^**”组合键；Emacs 将加高当前窗口，它下方的窗口将被相应地压低。如果你想加宽当前窗口，请按下“**C-x }**”组合键；Emacs 将加宽这个窗口，它右方的窗口将相应地变窄。

如果想让窗口变小一点，则可以收缩它们。如果想在垂直方向上收缩窗口，请输入“**ESC x shrink-window**”命令（注 3）。Emacs 将压低当前窗口一行，屏幕上的其他窗口将被相应地加高。如果想在水平方向上收缩窗口，请使用“**C-x {**”命令。这个命令将压窄当前窗口一列，其他窗口将在水平方向上做相应地扩展。

人们通常更喜欢以比较大的步子，而不是一次一行或一次一列地来调整窗口的大小尺寸。如果在这些命令的前面加上一个“**C-u**”，就能以一次四行或一次四列的方式来执行这些命令。举个例子，用“**C-u C-x ^**”来加高 `dickens` 窗口。

按下： **C-u C-x ^**

```
Buffers Files Tools Edit Search Help
Mock on Mock on Voltaire Rousseau
Mock on Mock on tis all in vain
--**- Emacs:blake          (Text Fill)--L1--Top-
It was the best of times, it was the worst of times, it was the age of
wisdom, it was the age of foolishness, it was the epoch of belief, it
was the epoch of incredulity, it was the season of Light, it was the
season of Darkness, it was the spring of hope, it was the winter of
despair, we had everything before us, we had nothing before us, we
were all going direct to Heaven, we were all going direct the other
way—in short, the period was so far like the present period, that
some of its noisiest authorities insisted on its being received, for
good or evil, in the superlative degree of comparison only.
---- Emacs:dickens          (Text Fill)--L1--All-----
C-u C-x ^
```

正如大家所预期的，当让一个窗口变大的时候，将会有一些原来看不见的编辑缓冲区文本填补进来。下面再介绍几个用来调整窗口尺寸的快捷操作。如果有一个非常

注 3： 如果你需要经常做这个操作，可以考虑把它绑定到一个组合键上去。具体步骤请参考第十一章。

小的编辑缓冲区（假设一天下来，用来定义简写词条目的编辑缓冲区里只有一行文本），就可以用“**C-x -**”（命令名是 **shrink-window-if-larger-than-buffer**）组合键让窗口收缩到编辑缓冲区那么小。如果编辑缓冲区的尺寸比窗口大，这条命令就什么也不做。按下“**C-x +**”（命令名是 **balance-windows**）组合键将使窗口的尺寸全都变成同样的大小。（“**C-x +**”命令在打开的窗口个数是一个奇数时也很有用，它会让那些窗口之间的显示都呈同样的大小。）

窗口大小的上、下限

Emacs 里的窗口最大不能超过屏幕，而最小则要看 Emacs 变量 **window-min-height**（默认值是 4 行）和 **window-min-width**（默认值是 10 个字符）的取值情况。当扩大某个窗口的时候，如果其他窗口被压缩到宽度小于 10 个字符或高度小于 4 行，Emacs 就会删除它们。可以根据自己的具体情况对这两个变量的值进行设置。变量设置方面的详细讨论请参考第十一章。

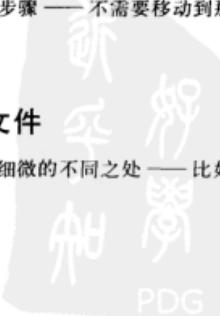
对其他窗口进行操作的快捷命令

Emacs 提供了几个对其他窗口进行操作的命令，这样不必移动到另一个窗口里就能对它进行操作。这些操作里最常用的是卷动其他窗口里的文本内容。如果想卷动（按顺时针方向）下一个窗口里的内容，请使用“**ESC C-v**”命令。

有几个“其他窗口”命令其实是一些中间插有数字“4”的普通命令。比如，如果想在另外一个窗口里查找并打开一个文件，就要按下“**C-x 4 f**”组合键。（如果当前只打开了一个窗口，那么 Emacs 会自动打开一个新窗口。）如果想在另外一个窗口里选择另一个编辑缓冲区，就要按下“**C-x 4 b**”组合键。与正常的“**C-x C-f**”和“**C-x b**”命令相比，在进行多窗口操作的时候，许多用户更喜欢这里介绍的命令，因为它们能够省略一个操作步骤——不需要移动到那个窗口，给出一命令，然后再移动回来。

对比两个窗口中的文件

如果想找出两个大文件之间细微的不同之处——比如想看看自己正在编辑的文件



和它的自动保存文件有什么不同，**compare-windows**命令就可以派上用场。先找到并打开有关文件，然后用**compare-windows**命令找出它们之间的第一个不同之处。

要想使用**compare-windows**命令，必须先把将要进行比较的两个编辑缓冲区分别在两个窗口里打开，它们可以左右并排、也可以水平排列（注4）。然后把两个编辑缓冲区的插入点分别移动到各自的开头，然后输入“**ESC x compare-windows**”或者从“Tools（工具）”菜单里选择“Compare（比较）”再选择“This Window And Next Window（本窗口和下一窗口）”。Emacs会把这两个编辑缓冲区都翻卷到出现不同之处的地方，并且会把两个编辑缓冲区各自的插入点，都放到出现不同之处的位置。这样，只需按下“**C-x o**”组合键在两个编辑缓冲区之间移动光标，就能看出这两个文件到底有哪些不同。

上面的操作只能找到两个编辑缓冲区之间出现的第一个不同之处。如果想找出第二个、第三个或者更多的不同之处，就得动些脑筋了。**compare-windows**命令只有在两个编辑缓冲区的插入点都精确地处于位置时才能工作，所以必须把它们在两个编辑缓冲区里分别移过第一个不同之处以后才能再次输入“**ESC x compare-windows**”。在找出两个文件之间的不同之处方面，UNIX操作系统的**diff**命令提供了更有效的方法（尽管它的输出看着有点费劲）。Emacs还准备了一个面向**ediff**的接口，在“Tools”菜单的“Compare”子菜单里也有一个对应的操作选项。

窗口操作方面的重要命令我们就介绍这么多，表4-1对它们进行了汇总。

表4-1：窗口命令速查表

键盘操作	命令名称	动作
C-x 2 <i>Files→Split Window</i>	split-window-vertically	把当前窗口分割为上、下排列的两个窗口
C-x 3	split-window-horizontally	把当前窗口分割为左、右排列的两个窗口
C-x >	scroll-right	窗口内容右卷
C-x <	scroll-left	窗口内容左卷

注4： 在屏幕上可以有两个以上窗口，但只有两个窗口能够参加比较：光标所在的那个窗口和按顺时针方向的下一个窗口。

表 4-1：窗口命令速查表（续）

键盘操作	命令名称	动作
C-x o	other-window	移动到其他窗口；如果有多个窗口，按顺时针方向移动到下一窗口
C-x 0	delete-window	删除当前窗口
C-x 1 <i>Files→One Window</i>	delete-other-windows	删除所有窗口，只保留当前窗口
(无)	delete-windows-on	删除某个给定编辑缓冲区上的所有窗口
C-x ^	enlarge-window	加高当前窗口
(无)	shrink-window	压低当前窗口
C-x }	enlarge-window- horizontally	加宽当前窗口
C-x {	shrink-window- horizontally	压窄当前窗口
C-x -	shrink-window-if- larger-than-buffer	如果编辑缓冲区比窗口小，就压缩窗口面积
C-x +	balance-windows	把所有窗口调整为同样大小
ESC C-v	scroll-other-window	对其他窗口做卷屏操作
C-x 4 f	find-file-other-window	在其他窗口里查找并打开一个文件
C-x 4 b	switch-to-buffer-other- window	在其他窗口里选择一个编辑缓冲区
(无) <i>Tools→Compare→This Window And Next Window</i>	compare-windows	对两个编辑缓冲区的内容进行比较，并显示它们之间的第一个不同之处

查看编辑缓冲区清单

Emacs 允许在一次编辑工作中创建任意多个编辑缓冲区，可要是打开的编辑缓冲区太多，就不一定都能记住。好在随时都能查到一个编辑缓冲区的清单表。这个清单

表能够提供许多重要的信息，比如上次存盘后是否又对某个编辑缓冲区进行过修改等。

无论何时，只要按下“**C-x C-b**”组合键，Emacs 就会把编辑缓冲区清单表列出来。它会在屏幕上创建出一个新的“*Buffer List*”窗口，所有的编辑缓冲区就列在里面。从“Buffers”菜单里选择执行“List All Buffers (列出所有的编辑缓冲区)”操作也可以调出编辑缓冲区清单。

按下： **C-x C-b**

Buffers Files Tools Edit Search Help			
It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way—in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or evil, in the superlative degree of comparison only.			
----- Emacs:dickens (Text Fill) -- L1 - All -----			
 M-Buffer Size Mode File			
----- ----- ----- -----			
. dickens 608 Text /home/deb/dickens			
* blake 135 Text /home/deb/fiction/blake			
joyce 0 Text			
scratch 0 Lisp Interaction			
* *Messages* 884 Fundamental			
mail 386 Mail			
* RMAIL 78648 RMAIL /home/deb/RMAIL			
Completions 107 Completion List			
* *Buffer List* 414 Buffer Menu			
 --*-- Emacs: *Buffer List* (Buffer Menu) -- L3 - All -----			

Emacs 显示编辑缓冲区的清单。

一般来说，这个画面将至少会列出两个不是自己亲自创建的编辑缓冲区。“*Buffer List*”是按下“**C-x C-b**”组合键查看编辑缓冲区清单时由 Emacs 创建的；而“*scratch*”缓冲区则是每次启动 Emacs 做文字处理工作时，都会由 Emacs 自动创建的一个空白编辑缓冲区。可以把这个清单用做一个提示性的资料窗口（“我的编辑缓冲区”），也可以从这个清单开始对编辑缓冲区进行处理，就像将在下一小节里介绍的那样。图 4-2 给出了编辑缓冲区清单里各种符号的含义。

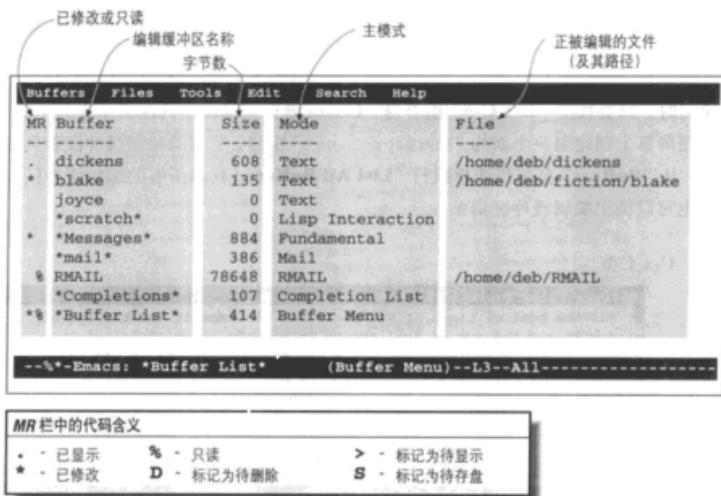


图 4-2：了解编辑缓冲区清单

与编辑缓冲区清单有关的操作

编辑缓冲区清单可不仅仅是一个提示性的资料窗口，可以从编辑缓冲区清单入手对编辑缓冲区进行显示、删除和保存等操作。按下“**C-x o**”组合键就能进入编辑缓冲区清单窗口。Emacs 将把光标放在第一列的位置上。在编辑缓冲区清单里，按“**C-n**”下移一行，按“**C-p**”上移一行，也可以按空格键来下移一行。在这种情形里（或者类似的其他情形里，比如将在下一章介绍的子目录编辑功能），空格键和“**C-n**”的作用是相同的。

可通过一系列单字符命令来对列在清单里的编辑缓冲区进行操作。如果想删除某个编辑缓冲区，则先把光标移动到该编辑缓冲区所在的那一行，然后按下“**d**”或“**k**”键；大写字母“**D**”将出现在第一列里。可以给任意多个编辑缓冲区加上待删除标记。这些编辑缓冲区并不会立刻被删除；只有在完成编辑缓冲区的标记工作并按下“**x**”（表示“execute”执行）键之后，它们才会被真正地删除。如果准备删除的编辑缓冲区与某个文件有关联，Emacs 会询问在继续执行任何操作之前，是否想保存

所做的修改。(注意：如果编辑缓冲区与文件没有关联，Emacs 就不会进行提问，所以在删除它们之前千万别忘了要先存盘。)

如果在按下“**x**”键之前又不想删除某个编辑缓冲区了，可以移动到相应的那一行并按下“**u**”键去掉待删除标记。更简便的办法是用 **DEL** 键来去掉上一个编辑缓冲区上的操作标记。为什么要这样做呢？原因很简单：“**d**”命令会自动前进到下一行。如果刚给一个编辑缓冲区加上待删除标记就改了主意，按一下 **DEL** 键当然要比按“**C-p u**”(移动到上一行，再去掉待删除标记)组合键简单。

如果想对某个编辑缓冲区进行存盘操作，则先把光标移动到该编辑缓冲区所在的那一行，然后按下“**s**”键；大写字母“**S**”将出现第一列里；再按下“**x**”完成真正的存盘动作。这样，可以一边查看着编辑缓冲区清单，一边给编辑缓冲区分别加上待删除标记或待存盘标记，最后按下“**x**”一次完成全部的工作。注意：如果你改了主意，“**u**”或 **DEL** 键也可以用来去掉待存盘标记。

有个命令是一经使用就会立刻作用于编辑缓冲区的，它就是“~(波浪号)”。按下“~”键将给编辑缓冲区加上一个未修改标记。这个符号的作用是告诉 Emacs 不要自动保存这个编辑缓冲区里的修改(既然这个编辑缓冲区没有被修改过，Emacs 也就没有必要通过它的自动保存功能来保存对它的修改)。如果此前真的做过一些修改，它们当然还都在编辑缓冲区里(只是在“欺骗”Emacs 说没有对那个编辑缓冲区做过修改而已)。如果在给编辑缓冲区加上未修改标记之后又对它进行了修改，Emacs 将觉察到它又被进行过改动，并且会自动地把它保存到一个备份文件里去。备份文件的文件名格式是“filename~”。

用百分号“%”键可以把一个编辑缓冲区的状态从读-写改变为只读。按下“%”键将立刻改变编辑缓冲区的读写状态。当编辑缓冲区处于只读状态时，它的状态行上将出现两个百分号(%%)。当正在对编辑缓冲区进行编辑的时候，可以用“**C-x C-q**”组合键来切换它的读-写和只读状态(我们在前面曾经介绍过)。

还可以通过编辑缓冲区清单把多个编辑缓冲区显示到多个窗口里去。如果想让某个编辑缓冲区满屏显示，则先要把光标移动到显示着编辑缓冲区清单的窗口里，再用“**C-p**”或“**C-n**”组合键移动到想要的编辑缓冲区条目上，然后按下“**1**”(数字1)键。Emacs 将把这个编辑缓冲区满屏显示。



如果想用某个编辑缓冲区替换编辑缓冲区清单，可以按下“f”键。如果想把某个编辑缓冲区放到另外一个（不是编辑缓冲区清单所占据的）窗口里，要按下“o”键；Emacs将把这个编辑缓冲区显示到另外一个窗口里，并且会把光标也放到那个窗口里。按下“C-o”组合键的结果稍有些不同：Emacs将把这个编辑缓冲区显示到另外一个窗口，但不会把光标也放到那里。

最后，还有一个编辑缓冲区显示命令。可以让Emacs动态地创建多个窗口来显示多个编辑缓冲区。先挑选出准备在窗口里显示的编辑缓冲区，方法是在你想要的编辑缓冲区条目按下“m”（意思是“mark”，标记）键；Emacs在用“m”键标记的编辑缓冲区旁显示一个大于号“>”。挑选完毕后按下“v”键，Emacs将自动创建一些水平窗口来显示那些加上“>”标记的编辑缓冲区。

删除“*Buffer List*”窗口的办法是：如果在编辑缓冲区清单窗口里，按下“C-x 0”组合键（数字0）；如果在某个编辑缓冲区的窗口里，按下“C-x 1”组合键（数字1）。

表 4-2 对与编辑缓冲区操作有关的命令进行了汇总。

表 4-2：编辑缓冲区操作命令速查表

键盘操作	命令名称	动作
C-x b	switch-to-buffer	移动到指定的编辑缓冲区
C-x C-b <i>Buffers→List All Buffers</i>	list-buffer	显示编辑缓冲区清单
C-x k <i>Files→Kill Current Buffer</i>	kill-buffer	删除指定的编辑缓冲区
(无)	kill-some-buffers	以问答方式删除各个编辑缓冲区
(无)	rename-buffer	把编辑缓冲区的名字改为指定的名字
C-x s	save-some-buffers	以问答方式存盘各个修改过的编辑缓冲区

表 4-3 对与编辑缓冲区清单有关的操作命令进行了汇总。

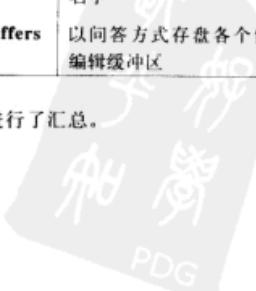


表 4-3：编辑缓冲区清单操作命令速查表

键盘操作	动作	执行情况
C-x n	移动到清单里的下一个编辑缓冲区（即编辑缓冲区清单里的下一行）	立即执行
SPACE	移动到清单里的下一个编辑缓冲区	立即执行
C-p	移动到清单里的上一个编辑缓冲区（即编辑缓冲区清单里的上一行）	立即执行
d	给编辑缓冲区加上待删除标记	按下“x”键时执行
k	给编辑缓冲区加上待删除标记	按下“x”键时执行
s	给编辑缓冲区加上待存盘标记	按下“x”键时执行
u	去掉编辑缓冲区上的操作标记	立即执行
x	对加有操作标记的所有编辑缓冲区执行相应的操作	立即执行
DEL	去掉上一个编辑缓冲区上的操作标记	立即执行
-	给编辑缓冲区加上未修改标记	立即执行
%	转换编辑缓冲区的只读状态	立即执行
l	把编辑缓冲区满屏显示	立即执行
2	把这个编辑缓冲区和下一个编辑缓冲区显示到两个水平窗口里	立即执行
f	在原本显示编辑缓冲区清单的窗口里显示此编辑缓冲区的内容	立即执行
o	把此编辑缓冲区显示到另外一个窗口里	立即执行
m	给编辑缓冲区加上待显示标志。参见“v”命令的说明	按下“v”键时执行
v	显示用“m”命令加上待显示标志的编辑缓冲区。Emacs 将动态地创建足够的窗口来显示加有这类标志的编辑缓冲区	立即执行
q	退出编辑缓冲区清单	立即执行



在文档中使用书签

用多个文件进行工作的时候，要想记住自己刚才正在什么地方进行编辑是一件很不容易的事情。幸好 Emacs 准备了书签功能，这是一种记录在文件中位置的简便办法，通过它容易就能回到书签所在的位置。假设正在编辑的文件有一个很长的路径名。每次启动 Emacs 时都必须输入这么长的路径名当然不是一件愉快的事，可以在这个文件里设置一个书签——假设给书签起的名字叫 “*current project*”。利用这个书签，就可以让 Emacs 自动查找并打开这个文件，然后把光标放到当初设置这个书签时所指定的位置。

书签使文件中的定位工作变得简单了。特别是当前项目在主目录以下几层深的某个子目录里，或者在另外一个完全不同的文件系统里时，在文件里设置一些书签能够快速地到达那里。我们将在第七章里介绍如何通过 ange-ftp 模式来查找远程系统上的文件，用户完全可以给这些文件也加上书签。这样，只需一个简单的查书签操作，就能迅速回到最喜欢的 FTP 档案。

请注意，如果是从 Emacs 里访问 World Wide Web（这部分内容也将在第七章里讨论）的，就不能互换使用 Emacs 书签和 W3 收藏夹条目。这主要是因为某些主流的浏览器把它们的收藏夹条目也叫做“书签”，弄清楚这一点是很重要的。

创建书签的时候，Emacs 将在主目录里创建一个名为 “*.emacs.bmk*” 的书签文件。它会在退出 Emacs 时把新书签都自动添加到这个文件里。

书签是按用户分别保存的。如果你和其他人都在阅读同一篇在线文档（比如 Emacs 在线手册），你将用你的书签来保存你的阅读位置，别人则用他们自己的书签来保存他们自己的阅读位置，各自的阅读是不会相互干扰的。

从 “Search” 菜单选择 “Bookmarks (书签)” 也能到达 “Bookmarks” 菜单，它列出了全部的书签命令以供选用。我们认为 Emacs 的书签菜单界面设计得特别好；即使你通常不使用菜单，也可能会对它情有独钟——至少在你学会有关的键盘操作命令之前会是如此。书签功能很容易让人上瘾，频繁使用它们的时候输入键盘命令总要比通过菜单进行操作要快捷一些。

书签的设置

如果想在光标位置设置一个书签，需要按下“**C-x r m**”（命令名是 **bookmark-set**）组合键或者从“**Bookmarks**”菜单选择执行“**Set Bookmark**（设置书签）”操作。Emacd 提示给书签输入一个名字，书签名的长度从理论上讲没有限制（但实践中不能长过屏幕的宽度），而且里面还可以有空格（例如“current project”、“Moore proposal's greatest flaw”或者“Othello Act 2 Scene 4”等）。如果还没有在这次编辑工作中使用过书签，Emacs 会把文件名放置在一对括号里作为默认书签（如果使用过书签，括号里放的就是书签的名字）。直接按回车键将接受默认的书签名，或者先输入一个书签名再按回车键。现在，就有了一个可以在任何 Emacs 编辑工作当中随时跳转过来的书签。

注意，如果在给新书签起名字时使用了一个曾经用过的书签名，Emacs 将认为是想用老书签来指示一个新位置，不管它原来是不是在另外一个文件里。所以，如果不是真的想移动书签的话，就一定要给书签起一个独一无二的名字。

移动到书签指示的位置

要想移动到书签指示的位置，请按下“**C-x r b**”（命令名是 **bookmark-jump**）组合键。接着输入书签的名字，或者输入它的前几个字母再按下 **TAB** 键；Emacs 将自动补足书签名，或者打开一个窗口把可能的书签名都列在其中。书签名出现后按下车键。Emacs 将查找并打开包含有这个书签的文件，并把光标放到书签位置上；不管文件的路径名有多么复杂，Emacs 都能找到它。用几个“远点的文件”来试试就能掌握这个技巧（注 5），说不定今后会几乎完全放弃使用文件 **find-file**（查找文件）命令“**C-x C-f**”呢。

利用菜单也可以迅速移动到某个书签所指示的位置。从“**Bookmarks**”菜单（它本身是“**Search**”查找菜单里的一个选项）里选择执行了“**Jump to Bookmark**（跳转至书签）”操作时，Emacs 会在一个窗口里把可用的书签都列出来。选择想要的书签，Emacs 会把文件显示在屏幕上，而光标将被放在那个书签所指示的位置上。

注 5： 这里所说的“远点的文件”指的是那些路径名相当复杂的文件，它们可能位于你的本地系统上、位于本地网络的其他系统上，或者位于可以通过 FTP 检索到的因特网主机上。使用 Emacs 来访问因特网的有关内容请参考第七章中的讨论。

书签的重新命名和删除

有时可能会觉得书签名字太一般，如果手里有好几个工作项目，“current project”（中文意思是“当前工作项目”）这样的名字未免有点含糊其词，日子长了就难免忘记它指的到底是什么东西了。对书签重新进行命名的方法是：输入“**ESC x bookmark-rename**”或者从“**Bookmarks**”菜单里选择执行“**Rename Bookmark**（重新命名书签）”操作。如果用键盘给书签改名，就需要在 Emacs 提示的“Old bookmark name:(旧书签名)”处输入书签旧名再按回车键。（如果使用的是 X 操作界面，就可以从一个窗口里选取旧名。）Emacs 接着会提示“New name:”（新名字），输入新的书签名再按回车键。给书签重新命名就这么简单，它既不会改变书签的指示位置，也不会改变它的内容，只是给书签改了个名字而已。

删除书签的方法是：输入“**ESC x bookmark-delete**”或者从“**Bookmarks**”菜单里选择执行“**Delete Bookmark**”操作。输入准备删除的书签的名字或者用鼠标选取它。删除书签对它所标记的文件不会造成任何影响。

这里又引出一个有趣的问题：如果在文件中删除了原来放有一个书签的文本时，会发生什么样的事情？因为书签指向文件里的一个位置而不是一段文本，所以文本被删除后，书签仍旧指向原来的位置——就像删除几个段落之后光标位置不会发生任何变化一样。自己实践一下就能明白了。用删除书签位置上文本的办法是删不掉书签的。

那如果删除一个原来包含着书签的文件时又会发生什么样的事情？如果删除了整个文件，或者哪怕只是给文件改了个名字，那么当再次访问关联于那个文件的书签时，Emacs 会显示一条如下所示的出错信息：

```
filename nonexistent. Relocate 'bookmark name'? (y or n)
```

如果只是对文件进行了重命名或移动而不是删除操作，回答“**y**”之后就可以把文件的新路径输入进去。如果回答“**n**”，Emacs 会再显示一条提示信息：

```
Bookmark not relocated, consider removing it
```

换句话说，Emacs 认为没有人会需要指向并不存在的文件的书签，我们也这样认为。

与书签清单有关的操作

还记得刚才介绍过的编辑缓冲区清单吗？书签也有一个类似的清单。与书签清单有关的操作也都是通过一些单字符命令完成的，这些命令能够一次处理好全部的书签。

开始对书签清单进行编辑的方法是：按下“**C-x r l**”（小写字母“L”）组合键或者从“**Bookmarks**”菜单里选择执行“**Edit Bookmark List**”操作。“*Bookmark List*”编辑缓冲区将出现在屏幕上。如下所示：

按下：**C-x r l**

Name	Path
R6source information	/source/README
RMAIL	/home/deb/RMAIL
blake	/home/deb/fiction/blake
dickens	/home/deb/fiction/dickens
heidegger	/home/deb/fiction/heidegger
hjames	/home/deb/fiction/hjames
joyce	/home/deb/fiction/joyce
capitalization	/online/ref/chicago

--%-- Emacs: *Bookmark List* - (Bookmark Menu) L3 * All -----
Loading bookmark...done

Emacs 把所有的书签及其关联文件的路径都列在这个清单里。

在这个清单里，“**d**”键给书签加上待删除标记，“**x**”键真正删除掉它们（与编辑缓冲区清单不同的是，书签清单里只有删除操作需要二次使用“**x**”命令）。如果改变了主意，在按下“**x**”键之前还可以用**DEL**键去掉“**d**”命令设置的待删除标记。按“**r**”键对书签进行重新命名，Emacs 会提示输入一个新的名字。如果想把书签全都保存起来，请按“**s**”键。如果想查看与书签关联着的文件，可以先用“**m**”键给书签加上待显示标记，书签旁边将出现一个大于号“>”标记。给所有想查看的书签都加好标记后，按“**v**”键，Emacs 会把与这些书签关联着的文件都打开在多窗口里（光标的位置不用说都应该在书签所指示的位置上）。如果只想进入某一个带书签的文件，可以不用先给它加上待显示标记，直接按“**v**”键即可，单按“**f**”键也行。

按“**t**”键可以对书签清单的显示方式稍加改变。在默认的情况下，这个清单先列出书签名，然后是其关联文件的完整路径。如果按下了“**t**”键，它就只列出书签名。

表 4-4 对与书签清单有关的命令进行了汇总。

表 4-4：书签清单编辑命令速查表

命令	动作
d	给书签加上待删除标记
r	对书签重新命名
s	保存清单里的全部书签
f	显示光标位置上的书签
m	给书签加上待显示标记。
v	显示加有待显示标记的书签。如果没有加上待显示标记的，就显示光标所在处的那个书签
t	切换书签关联文件的路径的显示 / 不显示状态
w	显示书签关联文件的存放位置（即路径名）
x	删除加有待删除标记的书签
u	去掉书签上的待操作标记
DEL	去掉上一行书签上的待操作标记
q	退出书签清单

与书签有关的其他编辑命令

除了前面介绍的这些操作以外，书签还有一些奇妙的操作命令。其中包括：**bookmark-insert**，用来把书签文件中的文本插到光标位置上；**bookmark-write**，提示输入一个用来保存书签的新文件名；**bookmark-load**，加载其他书签文件等等。这些命令不常用，不过也许各位会替它们想出一些合适的用途来。通过“**Bookmarks**”菜单也能访问这些命令。

表 4-5 对书签命令进行了汇总，包括“**Bookmarks**”菜单里的有关选项。

表 4-5：书签命令速查表

键盘操作	命令名称	动作
C-x r m <i>Search→Bookmarks→Set Bookmark</i>	bookmark-set	在当前光标位置处设置一个书签
C-x r b <i>Search→Bookmarks→Jump to Bookmark</i>	bookmark-jump	跳转到书签指示位置
(无) <i>Search→Bookmarks→Rename Bookmark</i>	bookmark-rename	重新命名一个书签
(无) <i>Search→Bookmarks→Delete Bookmark</i>	bookmark-delete	删除一个书签
(无) <i>Search→Bookmarks→Save (in default file)</i>	bookmark-save	把书签全都保存到默认的书签文件里
C-x r l <i>Search→Bookmarks→Edit Bookmark List</i>	bookmark-menu-list	进入“*Bookmark List*” 编辑缓冲区
(无) <i>Search→Bookmarks→Insert Content</i>	bookmark-insert	插入与给定书签关联着的文件的完整内容
(无) <i>Search→Bookmarks→Write (to another file)</i>	bookmark-write	把书签全都保存到一个指定的文件里
(无) <i>Search→Bookmarks→Load a Bookmark file</i>	bookmark-load	从指定文件里加载书签

临时性地挂起 Emacs

在大多数情况下，都能用“C-z”组合键把 Emacs 的运行暂停下来。如果使用的是

一个字符终端，就会到达 UNIX 操作系统的命令行提示符。重新返回 Emacs 的方法有两种，一种是在 UNIX 的命令行提示符处输入 “**fg**” 命令（如果 shell 有作业控制功能），另一种是在 UNIX 的命令行提示符处输入 “**exit**” 命令（如果 shell 要使用一个子 shell 来仿真作业控制功能）。如果不能肯定自己的 shell 有没有作业控制功能，或者是一个对作业控制还不太了解的新用户，可以两个命令都试试，不会有什么害处。肯定会有一个命令能工作，而另一个命令会给你一条无害的出错信息。

如果在运行 X 窗口系统时按下 “**C-z**” 组合键，结果就会稍有不同。Emacs 窗口将缩小为一个图标（图标可能是一只非洲羚羊的图案，也可能是一个流着水的水槽图案）。如果用鼠标打开这个图标（注 6），Emacs 窗口将恢复为原状。

重新回到 Emacs 时，一切都和刚才离开时一样。原来的文件还以原来的状态显示在屏幕上。甚至在挂起 Emacs 之前都用不着先把文件存盘（重新返回后再存盘也不迟）。如果不想让别人看到正在编辑的文字内容（比如，你正在为朋友安排一个惊喜聚会，可恰好有个朋友走进了你的办公室），你可以立刻把 Emacs 挂起来，不让他看到。话虽如此，我们还是建议大家在挂起 Emacs 之前最好把文件存盘，因为没人知道离开的时候都会发生哪些事情：忘记自己挂起过 Emacs 而去吃饭了、可计算机恰好就在这时候出了问题、而你前面编辑的东西也就都找不回来了。

能够临时性地把 Emacs 挂起来，去做一些其他工作是很有用的，特别是当工作在一台老式终端，而不是一台现代化的图形工作站上时更是如此。我们再次告诫大家：人们很容易忘记自己有一个“暂停”的 Emacs 编辑工作而又重新开始一次，而编辑的还是原来的那个文件。这样做很容易导致严重的混乱，如果比较懒于对文件进行存盘，这种情况就更容易发生。可能会有多个 Emacs 任务在同时运行，而各个任务编辑的却是同一个文件稍有不同的几个版本。可以用 UNIX 操作系统的 **ps** 命令（虽然它的输出不太容易看得懂）来查看是否有遗忘的 Emacs 任务。下面的输出情况告诉我们机器里运行着两个 Emacs 任务：

```
% ps
 PID TT STAT   TIME COMMAND
 129 co IW    0:01 sunview
 131 co IW    0:01 selection_svc
 134 co IW    2:39 clock -Wp 497 32 -Ws 210 47 -WP 704 0 -Wi
```

注 6： 打开一个图标所需要的鼠标动作将取决于使用的窗口管理器；它可能是一次鼠标左键单击、一次鼠标左键双击或者是一次鼠标中键单击。

```
135 co IW    1:01 mailtool -Wp 492 71 -Wa 670 770 -WP 908 0 -Wi
137 co IW    0:00 Mail -N -B -f /tmp/MTda00135
3474 p2 S    1:16 emacs ch01.2.mS
3585 p2 T    0:01 emacs others
3484 p3 S    0:00 ~bin/csh -i (csh)
3586 p3 R    0:00 ps
```

如果使用的 shell 支持 **jobs** 命令（比如 **csh** 和 **ksh**），就能得到下面这种更清楚的输出：

```
% jobs
[1] + Stopped                  emacs ch01.2.mS
[2] - Stopped                  emacs others
```

如果不知道自己用的是哪一种 shell，或者根本就不知道 shell 是什么，试试 **jobs** 命令也不会对你的系统造成任何损害，最多不过是看到一条“Command not found”（命令没有找到）的出错信息。

注意，**ps** 命令和 **jobs** 命令都只能告诉 Emacs 任务是如何启动的，它们不能告诉正在进行的工作是什么。这两个命令给出的文件名，即例子里的 *ch01.2.mS* 和 *others*，只是启动 Emacs 时指定的文件名。因为 Emacs 允许在编辑工作中随时打开和切换不同的文件，所以现在编辑的可能已经不是这两个文件了。

最后一句忠告：几乎永远没有任何理由离开 Emacs。即使自由软件基金会明天就抛弃了“**C-z**”命令，也不要丢掉它。我们现在还不能说明这是为什么，学到第五章时自然就会明白。

使用多个 X 窗口进行编辑

如果使用 X 窗口系统，那么就不仅限于 Emacs 窗口，还可以使用多个 X 窗口进行工作。Emacs 把 X 窗口称为“窗格（frame）”，这是为了把它们与前面已经介绍过的 Emacs 窗口区分开来。

Emacs 有一套完整的用来对窗格进行操作的命令，这些命令与 Emacs 窗口操作命令很相似。比如，如果想在另一个窗口里查找一个文件，相应的命令是“**C-x 4 f**”；而如果想在另外一个窗格里查找一个文件，相应的命令则是“**C-x 5 f**”。

窗格操作命令

打开一个新窗格的方法是：从“Files”菜单里选择执行“**Make New Frame**”制作新窗格操作或者按下“**C-x 5 2**”（命令名是 **make-frame**）组合键。Emacs 会创建一个包含当前编辑缓冲区的新窗格，并把它放到当前窗格的上面。

如果新窗格完全遮盖住当前窗格，则可能需要对新窗格的大小进行调整以区分它们。作为一个相对一劳永逸的解决方案，可以把下面这几条语句添加到“*.emacs*”文件里：

```
(setq initial-frame-alist '((width . 75) (height . 34)))
(setq default-frame-alist '((width . 60) (height . 20)))
```

这些语句对 Emacs 窗格尺寸的宽度和高度进行了设置。第一个窗格的尺寸将采用 **initial-frame-alist** 中的设置值（在上例中，宽度是 75 个字符，高度是 34 行），后续的窗格将采用 **default-frame-alist** 中的设置值，即 60 个字符宽、20 个字符高。可以根据显示器屏幕的尺寸对这些数字进行上、下调整。

按下：**C-x 5 2**（如图 4-3 所示）：

请注意新窗格的顶部。Emacs 在默认的情况下会把编辑缓冲区的名字放在那里（如果只有一个窗格，Emacs 会把系统的名字放在其顶部）。因为这里的情况是在同一个编辑缓冲区上有两个窗格，所以 Emacs 会把同样的名字放到新窗格的顶部。下面我们来在我们的 *dickens* 编辑缓冲区上打开一个窗格。如下所示：

输入：**C-x 5 f dickens RETURN**（如图 4-4 所示）：

新窗格的名字是 *dickens*，与它的编辑缓冲区同名。如果用“**C-x b**”组合键移动到另外一个编辑缓冲区，窗格顶部的名字将相应地改变为新编辑缓冲区的名字。（如果想移动到另外一个编辑缓冲区，并把它放到一个新窗格里，请按下“**C-x 5 b**”组合键——大家可能已经猜到了。）

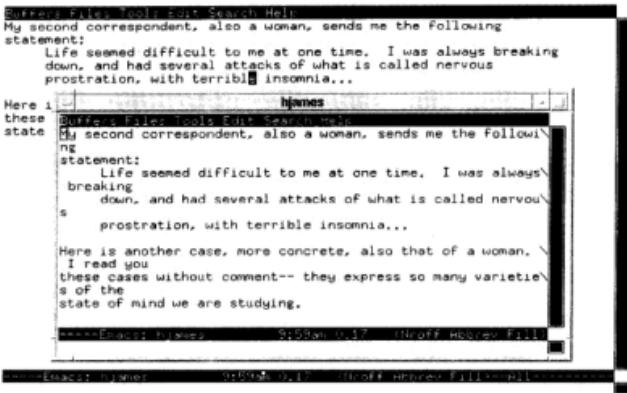


图 4-3: Emacs 打开了一个标题为 “hjames”的新窗格

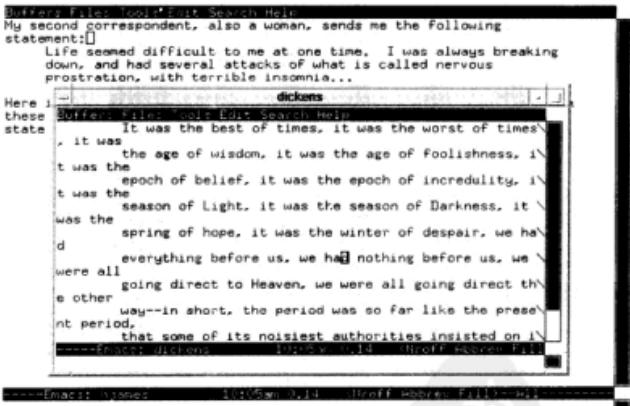


图 4-4: Emacs 打开了一个标题为 “dickens”的新窗格

有好几种办法可以用来在窗格之间移动。可以用鼠标选取窗格，也可以通过“C-x 5 o”组合键进入另外一个窗格。如果想查看所有当前窗格的一个清单列表，请在“Buffer”菜单里选择执行相应的操作。如果打开了一个以上的窗格，Emacs会弹出

一个窗口让用户对编辑缓冲区或窗格进行选择。如果选择的是“Frames (窗格)”，Emacs 就显示一个当前活动窗格的清单列表，可以用鼠标左键从中进行挑选。可以在“Buffer”菜单下来回切换显示编辑缓冲区清单和窗格清单。

如果想删除一个窗格，请按下“C-x 5 0”组合键或者从“Files”菜单里选择执行“Delete Frame (删除窗格)”操作；Emacs 将删除所在的窗格。删除一个窗格和删除一个窗口都只影响屏幕上的显示情况。原先的编辑缓冲区依然是活动的，可以用“C-x b”组合键来删除编辑缓冲区。

假设你需要经常在线查阅一本参考书——比如说《Chicago Manual of Style》吧。你可以用“C-x 5 r”命令以只读方式打开一个窗格，来查阅这份文件的一个副本（因为你并不是想对这本参考书的内容进行编辑修改），如下所示：

输入：C-x 5 r chicago (如图 4-5 所示)：

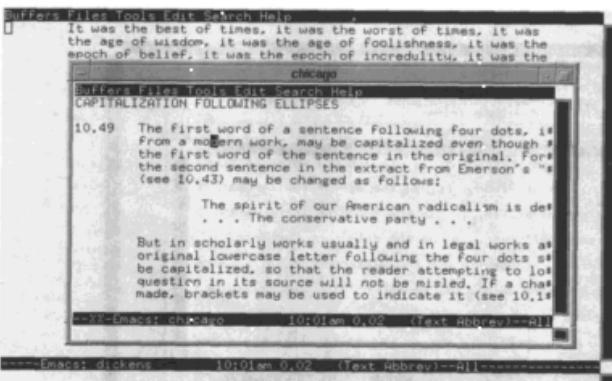


图 4-5：Emacs 打开一个新窗格查阅《Chicago Manual of Style》的内容

如果窗格遮住了屏幕上的其他东西怎么办？举个例子，只是偶尔查一下《Chicago Manual of Style》一书中的内容。那么，如果这个窗格遮住了屏幕上的其他东西，则可以把它挪到其他窗格的下面，或者就用“C-z”（命令名是iconify-or-deiconify-frame）组合键把它缩小为一个图标，可以随时用鼠标来重新打开它。利用“Buffer”

菜单来选取一个窗格，也会把它从图标恢复为原来的大小。（一旦窗格被缩小为图标，就不能再通过“**C-x 5 o**”组合键切换到它里面去；Emacs 会认为用户不想查看一个被缩小为图标的窗格。）

窗格与编辑缓冲区的关系

窗格和编辑缓冲区有着相当紧密的联系。Emacs 会给窗格起一个与编辑缓冲区同名的名字，这我们已经看到了。如果想查看编辑缓冲区清单，那么所有的编辑缓冲区都会被列出来，显示在其他窗格里的那些编辑缓冲区也将列在这个清单里。窗格与窗口有很多相似之处，它们都只是用来显示编辑缓冲区内容的一种手段。如果准备退出 Emacs，它会询问是否想保存被修改过的编辑缓冲区。回答完这些问题之后，所有的窗格都将被关闭。

与多编辑缓冲区、多窗格和多窗口有关的操作，到这里就介绍得差不多了。下一章将介绍它们的一些实际应用情况，有些操作（比如，子目录编辑器的使用和在 Emacs 里启动 UNIXshell 等）已经在这一章里一笔带过地提到。表 4-6 对窗格命令进行了汇总。

表 4-6：窗格命令速查表

键盘操作	命令名称	动作
C-x 5 o <i>Buffers→Frames</i>	other-frame	移动到其他窗格
C-x 5 0 <i>Files→Delete Frame</i>	delete-frame	删除当前窗格
C-x 5 2 <i>Files→Make New Frame</i>	make-frame	在当前编辑缓冲区上创建一个新窗格
C-x 5 r	find-file-read-only-other-frame	创建新窗格并查找文件，把编辑缓冲区设置为只读的（用来查阅你不小心修改了的文件）
C-x 5 f	find-file-other-frame	在一个新窗格里查找文件
C-x 5 b	switch-to-buffer-other-frame	创建新窗格并在其中显示另外一个编辑缓冲区

疑难解答

- 所有窗格的名字都一样。如果在对不同名字的多个编辑缓冲区进行编辑时遇到了这个问题，那么最可能的原因是“*.Xdefaults*”文件给 Emacs 窗格定义了一个统一的名字。需要对“*.Xdefaults*”文件进行编辑，删除出现“*emacs.title*”字样的那一行。保存“*.Xdefaults*”文件、退出 Emacs，然后退出登录，再重新登录上机。Emacs 将会根据正在编辑的编辑缓冲区给窗格加上标题。详细资料请参考第十四章。
- 菜单栏不见了。菜单栏只有在使用 Emacs 19.30 或以上的版本，或者是在 X 窗口系统里才能工作。如果确实有 X 窗口系统而菜单栏真的不见了（这一般出现在使用窗格进行工作的时候），输入“**ESC x menu-bar-mode**”就可以把它找回来。
- 书签文件的名字是“*.emacs-bkmrks*”而不是“*.emacs.bmk*”。这其实算不上是一个问题，这说明使用的 Emacs 版本低于 19.29。如果升级到 19.29 或者更高的版本，Emacs 将自动对这个文件进行转换，并且把它的名字改为“*.emacs.bmk*”。



第五章

Emacs 工作环境

本章内容：

- 在 shell 编辑缓冲区里执行 UNIX 命令
- 文件和目录操作
- Emacs 中的打印操作
- 用 Emacs 查阅 UNIX 的在线文档
- 时间管理工具的使用
- 用好 Emacs 工作环境

需要用 UNIX 的 shell 完成的许多工作，用 Emacs 也能完成得很好。诸如执行 UNIX 命令、对目录进行操作、打印文件之类的事情根本用不着离开 Emacs。想转去进行另外一项工作吗？很简单，切换到相应的编辑缓冲区或窗口就行了。如果屏幕大得足以容纳多个窗口，就可以用“C-x o”组合键在任务之间进行切换。

这有什么重要的吗？当然很重要，能够在任务之间快速切换本身就很有点好处；更重要的是不管正在做些什么，都会拥有一个统一的编辑环境：通过各种 Emacs 编辑命令，能够对文件进行文字处理、发出 shell 命令和启动目录编辑器做一些文件维护方面的工作等等。把文本从一个窗口移动到另外一个窗口，只能算是一种很简单的操作。可以执行一条 UNIX 命令，然后用 Emacs 命令把它的结果复制到某个文件里。虽然现代的窗口系统有很多优点，但就许多日常工作而言，Emacs 所提供的集成性是最好的。在此后的各章里，我们将讨论 Emacs 在电子邮件、Usenet 新闻、Telnet、FTP 和 WWW 等方面的使用方法。这一章只能算是一个入门，目的是介绍怎样才能把各种日常工作集成到 Emacs 里去。

在 shell 编辑缓冲区里执行 UNIX 命令

Emacs 的一个重要特色之一是它能够在一个编辑缓冲区里运行一个 UNIX 的 shell。只要进入 shell 编辑缓冲区，就可以在 Emacs 里完成各种常见的 UNIX 工作。这样做有哪些好处呢？

- 不必离开 Emacs 就能得到 UNIX 命令行提示符。如果想对正在编辑的文件进行打印或者编译，完全可以立即动手。
- 可以利用 Emacs 的编辑功能写出自己的命令。
- 可以利用 Emacs 的编辑功能“备份”自己的命令清单，复制旧命令，修改它，然后再执行它。
- 可以保存自己的 shell 编辑缓冲区，即保留一份自己编辑工作的日志——它将自动把运行的每一条命令的输出情况包括在其中。
- 可以把命令的执行结果复制到其他的窗口（其他文件、电子邮件消息等）里去。
- 可以把复杂的命令保存到一个文件里，然后把这个文件插入到命令行提示符处——这样就不必反复输入那些命令了。

习惯了使用 Emacs 进行工作之后，就会发现它越来越多的用途。

Emacs 还提供了几种能够不通过 shell 访问 UNIX 操作系统的功能。Emacs 有一个对目录进行（列文件清单和删除文件等）操作的工具，它比 ls 和 rm 等命令用起来还要方便。只要愿意，在执行有关操作的时候完全可以抛开那些 UNIX 命令。它还有一些用来对文件进行编译（即运行 make 命令）和打印的内置命令。

在这一小节里，我们将对 shell 模式进行讨论。在本章的后半部分，我们将讨论目录编辑、打印以及在 Emacs 里对个人日程进行简单管理的日历和日记等功能。现在从 shell 模式的一个简单变体开始讲起，这个功能能够一次执行一条 UNIX 命令。

一次执行一条 UNIX 命令

如果想在一次 Emacs 编辑工作的过程当中执行一条 UNIX 命令，请按下“**ESC !**”组合键。Emacs 将提示输入一条准备执行的命令。输入一个命令并按回车键，Emacs 将打开一个名为“*Shell Command Output*”的窗口，并把该命令的执行结果显示在其中。按下“**C-x 1**”组合键可以关闭这个窗口。



按下: **ESC !**

```
Buffers Files Tools Edit Search Minibuf Help
Stately,plump Buck Mulligan came from the stairhead,bearing a bowl
of lather on which a mirror and a razor lay crossed.A yellow
dressing gown,ungirdled,was sustained gently behind him by the mild
morning air.He held the bowl aloft and intoned:
  -Introibo ad altare Dei
-----Emacs:joyce          (Text Fill)--L1--All-----
Shell command: █
```

Emacs 提示输入一条准备执行的命令。

输入: **diff joyce joyce2**

```
Buffers Files Tools Edit Search Help
Stately,plump Buck Mulligan came from the stairhead,bearing a bowl
of lather on which a mirror and a razor lay crossed.A yellow
dressing gown,ungirdled,was sustained gently behind him by the mild
-----Emacs:joyce          (Text Fill)--L1--All-----
1,5c1,6
<Stately,plump Buck Mulligan came from the stairhead,bearing a bowl
---
>Stately,plump Duck Mulligan came from the stairhead,bearing a bowl
---*- Emacs: *Shell Command Output*          (Fundamental)--L6--Top-----
```

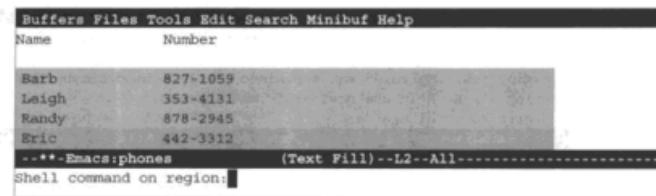
Emacs 执行 **diff** 命令并把其输出放到一个 “*Shell Command Output*” 编辑缓冲区里。

因为 **diff** 命令的输出是在一个编辑缓冲区里，所以可以对它进行编辑、保存，或者做任何想做的操作。

Emacs 的 shell 命令工具有一个很有意思的功能，即可以用某个编辑缓冲区中的一个文本块，而不是一个传统的文件作为命令的输入。比如，假设对一个电话号码表进行排序。首先，要把光标放到电话号码表里的某个位置（比如“Eric”的头一个字符上）；然后，发出 **mark-paragraph** 命令（“**ESC h**”组合键）。这个命令将把电话号码表定义为一个文本块，光标在这个段落的开头，文本标记在这个段落的末尾。在下面的示例中，加阴影的区域代表准备进行排序的文本块范围。在选取好文本块之后，按下 “**ESC !**”（命令名是 **shell-command-on-region**）组合键，Emacs 提示输入将要执行的 shell 命令。



按下: **ESC h ESC!**



The screenshot shows the Emacs interface with a menu bar at the top. The buffer content is a table of names and phone numbers:

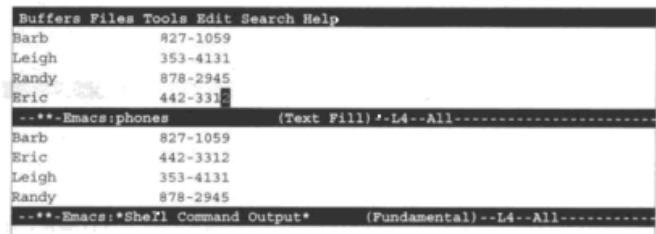
Name	Number
Barb	827-1059
Leigh	353-4131
Randy	878-2945
Eric	442-3312

Below the table, the status bar displays: **--***-Emacs:phones (Text Fill)--L2--All-----**. A message in the minibuffer says: **Shell command on region: |**.

Emacs 提示输入一条准备执行的命令。

接着, 发出 **sort** 命令, 但不必指定任何输入文件 (注 1)。提供输入数据的工作将由 Emacs 负责。

输入: **sort RETURN**



The screenshot shows the Emacs interface with a menu bar at the top. The buffer content is the same table of names and phone numbers, but now it is sorted by name:

Name	Number
Barb	827-1059
Leigh	353-4131
Randy	878-2945
Eric	442-3312

Below the table, the status bar displays: **--***-Emacs:phones (Text Fill)--L4--All-----**. The minibuffer shows the previous command: **--***-Emacs:phones (Text Fill)--L4--All-----**.

Emacs 对刚才定义的文本块进行了排序。

Emacs 对电话号码表 (即文本块中的全部内容) 进行了排序。“**ESC !**”有一个很有用的变体, 即允许把命令的执行结果直接放到当前编辑缓冲区而不是“*Shell Command Output*”编辑缓冲区里。这需要在“**ESC !**”命令的前面加上一个“**C-u**”; 比如, “**C-u ESC !**”将执行一条 shell 命令, 并把其输出放到当前编辑缓冲区里。如下所示:

注 1: 事实上, Emacs 自己的“**ESC x sort-lines**”命令执行起来会更快一些。我们那么做是为了找个简单点的例子。

输入: C-u ESC ! ls RETURN

```
Buffers Files Tools Edit Search Help
Here's a list of files in my current directory:
INDEX
News
RMAIL
RMAIL~
addresses
albert
broken.ps
--*** Emacs:directory (Text Fill)--L1--Top-----
```

Emacs 执行 ls 命令并把其执行结果放到当前编辑位置。

在一般情况下, Emacs 使用默认的 shell 来执行各种命令。如果你平时使用的是 C shell(**csh**), 它使用的也将是 C shell; 如果你平时使用的是 Bourne shell(**sh**), Emacs 也将使用 Bourne shell 来执行各种命令。依次类推, 可以用修改变量 **shell-file-name** 取值的方法把它换成另外一种不同的 shell。这个变量的取值必须是新 shell 的可执行文件的完整路径。比如, 如果想用 GNU 项目下的 Bourne-again shell (**bash**) (注 2) 来执行命令, 就需要把下面这条语句添加到 “*.emacs*” 文件里:

```
(setq explicit-shell-file-name "/bin/sh")
```

使用 shell 模式

从现在开始, 我们将介绍 Emacs 编辑器的 shell 模式, 用来执行 UNIX 命令的交互式功能组件。启动 shell 编辑缓冲区的方法是给出 “**ESC x shell**” 命令, 这将创建出一个名为 “*shell*” 的编辑缓冲区。在这个编辑缓冲区里会出现 UNIX 操作系统的 shell 的提示符。如下所示:

输入: **ESC x shell**

```
Buffers Files Tools Edit Search Complete In/Out Signals Help
>

--*** Emacs:*shell* (Shell:run)--L2--All-----
Loading shell...done
```

现在, 可以输入 shell 命令了。

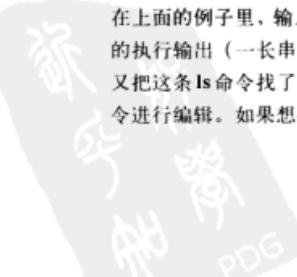
注 2: 对不同的计算机系统而言, **bash** 的存放位置是会变化的。它甚至有可能在你的系统上根本就不可用。

除了可以在输入命令时用Emacs对之进行编辑以外，就其大部分而言，shell模式与正常的UNIX命令界面几乎是一模一样的。可以把命令从一个地方复制到另一个地方，或者把命令的执行结果复制到某个文件里等等。

有几个注意事项是必须知道的。比如，通常会通过按下“C-c”组合键来中止一个作业；但如果想在shell模式里用按下“C-c”组合键的办法来中止一个程序（因为有许多Emacs命令都是以“C-c”开头），Emacs将认为这个“C-c”是它自己的编辑命令的一部分。因此，如果在Emacs中想通过“C-c”组合键来中止当前作业，就必须按下“C-c C-c”组合键才行。类似地，必须按下“C-c C-z”组合键而不是“C-z”组合键才能停止一个作业的运行；必须按下“C-c C-d”组合键而不是“C-d”组合键等等（严格地说，不一定非得按下“C-c C-d”组合键，因为Emacs能够根据上下文领会“C-d”的含义。如果是在编辑缓冲区的末尾，“C-d”的含义将是“文件尾”；如果是在其他地方，“C-d”将删除一个字符）。另外，如有必要，可以从“Signals (信号)”菜单里选择适当的选项而不使用控制字符（比如，通过菜单选择“EOF”而不是按下“C-d”组合键）。

shell模式还准备了一些简化操作的快捷键。比如，命令“**ESC p**”把输入的最后一个命令放到shell模式的命令提示符处，不管已经在编辑缓冲区里移动了多远。连续按下“**ESC p**”组合键将把更早的命令找回来。如下所示：

按下： **ESC p**



```
Buffers Files Tools Edit Search Complete In/Out Signals Help
coatimundi.txt fish.not lemur.c quetzal.c zooanimals.txt
detroit.ms fossa.ms ocelot.txt whales.txt zymurgy.ms
<ruby>
` ls zoostuff
---- Emacs: *shell* (Shell: run) --L4-- Bot
History item: 1
```

“**ESC p**”把你输入的最后一个命令放到shell模式的命令提示符处，即使它已经在屏幕上去了。

在上面的例子中，输入的最后一条命令是“**ls zoostuff**”。它已经不在屏幕上去了，它的执行输出（一长串文件清单）已经把它从屏幕顶部推出去了。可命令“**ESC p**”又把这条**ls**命令找了回来，但不会执行它；在按下回车键之前，还有机会对这个命令进行编辑。如果想把再早一些的命令找回来，可以按下“**ESC n**”组合键。

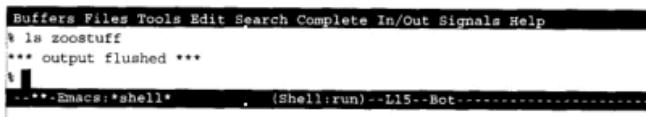
这些命令可能很面熟，的确如此。它们就是所谓的“历史记录”命令，与在第三章里讨论的辅助输入区的历史记录命令是完全一样的。Emacs专门准备了一个完备的“In/Out（输入/输出）”菜单，来负责命令的历史记录方面的工作。

在shell模式里，回车键（RETURN）和制表键（TAB）有特殊的功用。按下回车键将执行当前命令行上的命令，即使移动到某条老命令所在的那一行上。按下回车键的时候，Emacs会把这条命令复制到shell编辑缓冲区的末尾并执行它。在按下回车键之前，还有机会对任何命令进行编辑修改。

按下制表键（TAB）将启用Emacs的自动补足功能。自动补足的对象可以是操作系统命令、文件名和变量等等。可以从“Complete（自动补足）”菜单开始访问这些命令，但按TAB键通常更简便一些。

如果输入了一个会产生大量输出的命令，并因此而妨碍了编辑工作，那么有一个简便的办法可以预防这种情况：键入命令后按下“C-c C-o”（命令名是comint-kill-output）组合键，或者从“In/Out”（输入/输出）菜单里选择执行“Kill Current Output Group”（清除当前输出组）操作，不要按回车键。如下所示：

按下：C-c C-o



The screenshot shows the Emacs interface with the shell buffer active. The buffer contains the following text:

```
Buffers Files Tools Edit Search Complete In/Out Signals Help
* ls zoostuff
*** output flushed ***
t
***- Emacs: *shell* . (Shell:run)--L15--Bot-----
```

“C-c C-o”自动删除上一个命令的执行输出。

前一条命令（ls zoostuff）还在屏幕上，可它那一长串文件清单输出却没有出现在屏幕上。“C-c C-o”只能删除刚键入的那条命令的输出；如果以前的命令都是按回车键执行的，那么命令的执行结果将出现在屏幕上，“C-c C-o”是不能删掉这类命令输出的。“C-c C-o”特别适用于给出了一个可能会产生大量输出的命令，但却并不关心它们的情况。“C-c C-o”还提供了一种把UNIX命令的执行结果，复制到另外一个编辑缓冲区里去的便利手段。只要移动到另外一个编辑缓冲区里再按下“C-y”（命令名是yank），就可以把那些输出结果找回来。

shell 模式另外一个很有用的命令是 “C-c C-r”（命令名是 **comint-show-output**）。这个命令特别适用于这样的情况：命令产生大量的输出而输出内容的前几行卷出了屏幕。“C-c C-r”会对窗口进行调整，使最后一条命令的输出内容的前几行出现在窗口的最顶部。如果想查看输出内容的最后几行，请按下 “C-c C-e”（命令名是 **comint-show-maximum-output**）组合键或者从 “In/Out” 菜单里选择执行 “Show Maximum Output（显示最大输出）” 操作。这个命令的作用是把输出内容的最后一行挪到窗口的最底一行。

编写书刊的时候，希望能够以段落为单位进行移动是一种很合理的要求；但在使用 shell 模式的时候，以输出组为单位进行移动可能更有用一些。一个输出组由一条命令和它的执行输出内容构成。如果想移动到前一个输出组，请按下 “C-c C-p” 组合键或者从 “In/Out” 菜单里选择执行 “Backward Output Group”（后退到上一输出组）操作。如果想移动到后一个输出组，请按下 “C-c C-n” 组合键或者从 “In/Out” 菜单里选择执行 “Forward Output Group”（前进到下一输出组）操作。

shell 模式的一个突出优点是可以在一条命令开始执行之后，在这条命令的执行过程中继续对其他一些编辑缓冲区进行编辑。shell 编辑缓冲区甚至可以不出现在屏幕上；只要输入 “**ESC x shell**” 就可以再把 shell 编辑缓冲区调回到屏幕上。（与此形成鲜明对照的是 “**ESC !**” 和 “**ESC !**”，它们将强制性地等到 UNIX 命令执行完毕后，才能继续进行其他编辑工作。）如果 shell 不具备作业控制功能，这种允许在命令执行期间继续进行编辑的做法无疑是一种福音。即使 shell 具备作业控制功能，与让命令运行在后台而继续编辑工作的情况相比，shell 模式还是要方便得多。有了 Emacs，就用不着担心后台作业的输出会干扰到正在进行的工作了。它的输出会完整地保存到 shell 编辑缓冲区里。可以用 “**ESC x rename-uniquely**” 命令重新命名 shell 编辑缓冲区，而这将同时拥有多个 shell 编辑缓冲区。可以重复、重复、再重复地这样做，直到 shell 编辑缓冲区的个数满足工作需要为止。

当前目录

在本书的开头，我们介绍了默认目录的概念。具体规定是：如果光标在某个文件里，则默认目录就是该文件所在的那个目录。启动一个 shell 的时候，Emacs 会自动进入它的默认目录。也就是说，创建 shell 编辑缓冲区时所在的文件目录就是最初的工作目录。

此后的事情也许会变得很复杂。Emacs 会根据 **cd**、**popd** 和 **pushd** 等命令（C shell 中用来改变目录的命令）来改变其默认目录的表示方法。如果还没有被弄糊涂，就该知道用来查找文件的默认目录永远都和当前目录一样。比如，如果执行完 “**cd /home/src/emacs/lisp**” 命令之后，又准备用 “**C-x C-f**” 编辑一个文件，则用来查找文件的默认目录就将是 “**/home/src/emacs/lisp**”。

这个功能存在着以下几个问题：

- Emacs 只有在把 **cd**、**popd** 和 **pushd** 等命令输入到命令行的最开始时才能检测到它们。
- Emacs 理解不了假名或其他任何会改变目录路径的程序。可以用修改变量 **shell-cd-regexp**、**shell-pushd-regexp**、**shell-popd-regexp** 的方法来帮助它识别这些命令。这几个变量的值都是正则表达式，它们匹配的东西是用来完成 **cd**、**popd** 和 **pushd** 等操作的命令。对正则表达式的详细讨论见第十三章。
- 最后，有些用户（包括我们几位作者在内）不喜欢自己的默认目录变来变去。如果各位也像我们一样不喜欢它，请把下面这几条语句添加到 “**.emacs**” 文件里：

```
(setq-default shell-cd-regexp nil)
(setq-default shell-pushd-regexp nil)
(setq-default shell-popd-regexp nil)
```

shell 的初始化

Emacs 是怎样知道应该启动哪个 shell 的呢？它首先查看的是变量 **explicit-shell-file-name**；接着查看的是一个名为 **ESHELL** 的 UNIX 环境变量；最后查看的是一个名为 **SHELL** 的环境变量。因此，如果在 Emacs 里时想另外运行一个特定的 shell（比如 Bourne shell），就应该把下面这条语句添加到 “**.emacs**” 文件里：

```
(setq explicit-shell-file-name "/bin/sh")
```

当 Emacs 启动一个交互式的 shell 时，它会在 shell 正常的启动文件之后再额外运行一个初始化文件。这个文件的名字是 “**.emacs_shell-name**”，其中的 “**shell-name**” 是打算在 Emacs 里使用的 shell 的名字。它必须保存在主目录里。比如，如果打算使用 C shell，就必须把它们放到文件 “**&.emacs_csh**” 里作为 Emacs 专用的启动命

令。再比如，如果想把 Emacs 的 shell 模式的命令行提示符设置为“emacs:%”，还想把一个名为 **WITHIN_EDITOR** 的环境变量设置为“T”，就必须在“*.emacs_csh*”文件里加上如下所示的语句：

```
set prompt="emacs:% "
setenv WITHIN_EDITOR T
```

在 shell 编辑缓冲区里，Emacs 还会把环境变量 **EMACS** 设置为“t”、把终端类型（即环境变量 **TERM**）设置为“**emacs**”。

远程 shell 的问题

如果是在一个网络上，那么可能会想先运行 Emacs，进入它的 shell 编辑缓冲区，然后在这个窗口里运行 **rlogin** 来访问其他的系统。这种情况经常会在屏幕显示方面出现一个小问题——已输入的 UNIX 命令在屏幕上出现了两次，而且大多数命令行的后面还多出了一个“^M”字样。而除此之外，其他一切都很正常。如下图所示：

初始状态：



The screenshot shows an Emacs window with a menu bar at the top. The menu items are: Buffers, Files, Tools, Edit, Search, Complete, In/Out, Signals, Help. Below the menu, there is a shell-like interface. The user has run several commands: rlogin remhost, remhost % pwd, remhost % ls, remhost % ls, remhost % file1 file2 file3, remhost % . At the bottom of the buffer, it says '--*- Emacs: *shell*'. The status bar at the bottom right indicates '(Shell:run)--L37--Bot--'.

每个命令都会重复显示一遍，而且还有一些不知从哪儿冒出来的“^M”。

这个问题很容易解决。在远程系统的 shell 编辑缓冲区里，输入 UNIX 命令“**stty -echo nl**”。如下图所示：



输入: **stty -echo nl**

```
Buffers Files Tools Edit Search Complete In/Out Signals Help
* rlogin remhost
remhost% stty -echo nl
stty -echo nl ^M
remhost% pwd
/home/remhost/deb
remhost% [REDACTED]
-- Emacs: *shell*          (Shell: run) -- L39 -- Bot -----
```

使用 **stty** 命令加以改正之后的来自远程系统的输出情况。

如果这条 **stty** 命令能自动执行，那当然就更理想了。可以做到这一点，方法是对打算使用的所有远程系统上的 “*.cshrc*” 文件（适用于 C shell 用户）和 “*.profile*” 文件（适用于 Korn shell 或 Bourne shell 用户）进行修改。具体方法如下：

如果是 **csh**:

```
if ($TERM == emacs) stty -echo nl
```

如果是 **ksh** 或 **sh**:

```
if [ $TERM = "emacs" ]
then
    stty -echo nl
fi
```

在上面那个针对 **ksh** 或 **sh** 进行设置的例子里，千万要在括号左右多加一个空格。忘记这些空格是一种经常出现的错误。

用 shell 模式预防安全隐患

在默认的情况下，shell 模式将把输入的一切内容显示在屏幕上。如果输入了用来转换到另外一个 UNIX 账户去的 “**su**” 命令，并输入了相应的口令字，则那个口令字也将显示在屏幕上，这无疑是一个巨大的系统安全漏洞。不过，这个问题是有办法解决的。在输入口令字之前，需要先输入 “**ESC x send-invisible**”，然后 Emacs 提示输入不再显示的文本。输入一个字符时，Emacs 会在辅助输入区里显示一个星号，按下回车键，Emacs 把口令字送出去。整个过程中，口令字始终没有以明文的形式显示在屏幕上。如果想让 Emacs 在输入口令字的时候，总是把口令字隐藏起来，请把下面这两行语句添加到 “*.emacs*” 文件里：

```
(add-hook 'comint-output-filter-functions
          'comint-watch-for-password-prompt)
```

以后，只要屏幕上再出现口令字提示符，Emacs 就会要求把不显示的文字输入到辅助输入区里，这就保证了口令字将再也不会出现在屏幕上。但要注意的是，这个功

能在最近才添加到 Emacs 19 里去的。Emacs 19.22 就没有这项功能，Emacs 19.28 里有。（用“**ESC x version**”命令可以查出使用的 Emacs 版本。）

表 5-1 对 shell 模式里使用的各种命令进行了汇总。

表 5-1：shell 模式命令速查表

键盘操作	命令名称	动作
(无)	shell	进入 shell 模式
C-c C-c <i>Signals → BREAK</i>	comint-interrupt-subjob	中断当前作业；相当于 UNIX 的 shell 中的“C-c”组合键
C-d	comint-delchar-or-maybe-eof	如果是在编辑缓冲区的末尾，送出 EOF 字符；如果是其他位置，删除一个字符
C-c C-d <i>Signals → EOF</i>	comint-send-eof	送出 EOF 字符
C-c C-u	comint-kill-input	删除当前行；相当于 UNIX 的 shell 中的“C-u”组合键
C-c C-z <i>Signals → STOP</i>	comint-stop-subjob	对非 X 用户，挂起或者停止一个作业；相当于 UNIX 的 shell 中的“C-z”组合键
ESC p <i>In/Out → Previous Input</i>	comint-previous-input	检索此前的上一个命令（可以重复执行以找回更早的命令）
ESC n <i>In/Out → Next Input</i>	comint-next-input	检索此后的下一个命令（可以重复执行以找回更近的命令）
RETURN	comint-send-input	送出输入在当前行上的命令
TAB <i>Complete → Complete Before Point</i>	comint-dynamic-complete	自动补足当前命令、文件名或者变量名
C-c C-o <i>In/Out → Kill Current Output Group</i>	comint-kill-output	删除最后一条命令的输出
C-c C-r	comint-show-output	把输出内容的第一行移到窗口的顶部

表 5-1: shell 模式命令速查表 (续)

键盘操作	命令名称	动作
C-c C-e <i>In/Out →</i> Show Maximum Output	comint-show-maximum-output	把输出内容的最后一行移到窗口的底部
C-c C-p <i>In/Out →</i> Backward Output Group	comint-previous-prompt	移动到前一条命令
C-c C-n <i>In/Out →</i> Forward Output Group	comint-next-prompt	移动到后一条命令

文件和目录操作

Dired 模式 (directory editing mode, 目录编辑模式) 是 Emacs 最引人注意的功能之一, 它提供了对文件目录进行编辑的有效手段。可以查看目录里所有文件的一个完整清单, 对它们进行删除、重命名、复制以及其他各种基本的文件操作。一旦用熟了 *Dired*, 就再也不需要 UNIX 下的 **cp**、**rm** 或 **mv** 等命令了。尤其重要的是, *Dired* 能够提高工作效率。可以同时对一组文件进行操作, 比如删除、移动和压缩等等, 甚至可以对文件里的字符串进行查询 - 替换。

进入目录编辑模式的方法有好几种。如果还没有进入 Emacs, 可以在启动 Emacs 的时候用一个目录名作为其命令行参数, 比如:

```
% emacs fiction
```

则 Emacs 将开始对目录 *fiction* 进行编辑——出现一个单独的窗口, 里面是 *fiction* 目录里的文件清单。用 “C-x C-f” 命令 (或任何可以打开文件的其他编辑命令) 也能进入目录编辑器, 但前提是给出的必须是一个目录名而不是一个文件名。比如, 输入 “C-x C-f *fiction*” 将开始对 *fiction* 目录进行编辑。按下 “C-x d” (命令名是 *dired*) 组合键或者从 “Files” 菜单里选择执行 “Open Directory” 操作也能启动 *Dired* (还得再输入一个目录名)。不管用什么办法启动目录编辑器, 结果都将是一样的。

输入： C-x C-f fiction RETURN

Buffers	Files	Tools	Search	Operate	Mark	Regexp	Immediate	Subdir	Help
<code>/home/deb/fiction:</code>									
	total 641								
	drwxrwxr-x 2 deb			512	Jun 23	13:22 .			
	drwxrwxr-x 13 deb			7680	Jun 23	13:22 ..			
	-rwxrwxrwx 1 deb			54576	Apr 22	18:55 ch01			
	-rwxrwxrwx 1 deb			79915	Apr 22	18:55 ch02			
	-rwxrwxrwx 1 deb			82809	Apr 22	18:56 ch03			
	-rwxrwxrwx 1 deb			79434	Apr 22	18:56 ch04			
	-rwxrwxrwx 1 deb			85805	Apr 22	18:56 ch05			
	-rwxrwxrwx 1 deb			109282	Apr 22	18:56 ch06			
	-rwxrwxrwx 1 deb			38191	Apr 22	18:57 ch07			
	-rwxrwxrwx 1 deb			91235	Apr 22	18:57 ch08			
	-rw-rw-r-- 1 deb			611	Aug 25	1995 dickens			
	-rw-rw-r-- 1 deb			315	Aug 26	1995 elephant			
	-rw-rw-r-- 1 deb			418	Aug 26	1995 hjames			
<code>--%-%-Dired:fiction</code>									
<code>(Dired by name) --L5-- Top</code>									
<code>Reading directory /home/deb/fiction/...done</code>									

目录编辑器的初始画面将出现在屏幕上。

正如大家所看到的，Dired的初始画面与在UNIX提示符处输入ls-l命令后看到的内容基本相同。在文件名的前面依次列出了文件的存取权限、文件的属主、系统名、文件长度和最近一次修改日期等项目。这个目录里所有的文件和下级子目录都列在里面，包括那些文件名以句点打头的文件。光标的初始位置在一个文件名上，不在第一列里。

在默认的情况下，文件清单将根据文件名进行排序，但也可以让它根据日期来排序。再来看看状态行，它上面将出现“(Dired by name)”字样。如果想对画面中的文件清单重新进行排序，请按“s”键（命令名是dired-sort-toggle-or-edit）。这个命令将把最新的文件放在清单的最顶部，很容易地就解决了“我昨天编辑的文件是哪一个？”的问题，状态行将相应地出现“(Dired by date)”字样。再按一次“s”键，将回到按文件名的字母顺序排序显示文件清单的画面。

如果各位还记得用来对编辑缓冲区清单进行编辑的命令（参见第四章），就会发现它们与将要介绍的用在目录编辑器里的命令几乎都是一样的。不过，虽然命令还是那些基本的命令，但能做的事情可大不一样。

注意：请记住，目录编辑器是直接对文件而不是编辑缓冲区进行工作的。如果用Dired删除了一个文件，它可就永久性地“消失”了。



在 Dired 里有好几种移动办法。“**SPACE**”、“**C-n**”和“**n**”等键盘命令都会移动到文件清单中的下一个文件上；而“**DEL**”、“**C-p**”和“**p**”等键盘命令都会移动到前一个文件上。还可以使用任何一种查找命令（递增查找、单词查找等等）来找出某个特定的文件。

查看和编辑文件

在查看某个目录的文件清单时，可能需要快速查看一下那些文件的内容。Dired 的“**v**”命令就是用来完成这项工作的：把光标放到想查看的文件上，然后按下“**v**”（命令名是 **dired-view-file**）键或者从“**Immediate (立即)**”菜单里选择执行“**View This File**”（查看这个文件）操作即可。Emacs 会把这个文件以查看模式（view mode）显示在屏幕上，此时的文件内容只能看，不能改。按下“**C-e**”或“**q**”键（哪个方便按哪个），将返回到目录的文件清单窗口。在查看文件内容的时候，可以用“**s**”键开始一次递增查找操作，或者按下回车键前进到下一行。按下“**=**”键，Emacs 会告知光标在哪一行。还有一些其他 Emacs 命令（比如标记文本块等）的快捷键，但老实说，常规命令都能正确工作——如果已经掌握的那些命令都能工作得很好，再去另外记忆一套特殊的命令似乎就没什么必要了。

如果想从 Dired 编辑缓冲区开始去编辑某个文件，请按下字母“**f**”或者从“**Immediate**”菜单里选择执行“**Find This File**（找出这个文件）”操作。Emacs 会把这个文件找出来以便对它进行编辑。这可完全是一个普通的编辑缓冲区可以做任意的编辑修改、保存、查看其他文件等各种操作。按下“**C-x b**”组合键后，再输入正在工作的子目录名将回到 Dired 编辑缓冲区，也可以通过编辑缓冲区清单（“**C-x C-b**”组合键）重新回到 Dired 编辑缓冲区。

能够从 Dired 编辑缓冲区出发查看和编辑文件当然是件好事，不过编辑方面的事情我们都已经向大家介绍过了。下面来学点我们还不知道的事情——如何删除文件。

文件的删除、复制、重命名操作

已经讲过，文件的删除与通过编辑缓冲区清单进行的编辑缓冲区删除在操作上是一样的。只要知道怎样删除编辑缓冲区，就等于已经掌握了通过 Dired 删除文件的基本方法。首先，需要给文件加上一个待删除标记——把光标移动到那个文件的名

字上并按下“**d**”键，这会在该文件名的左边放上一个大写的字母“D”，而光标将移动到文件清单里的下一个文件上。可以随心所欲地给任意多个文件加上待删除标记；如果此时改变了主意，还来得及用“**u**”键去掉文件的待删除标记。然后，按下“**x**”键把加有待删除标记的全部文件一次性地都删除掉（具体情况马上就要讲到）。下面是给几个文件加上待删除标记后 **Dired** 编辑缓冲区的显示画面示例：

按下： **d** (可以给任意多个文件加上待删除标记)

```

Buffers Files Tools Search Operate Mark Regexp Immediate Subdir Help
/home/deb/fiction:
total 641
drwxrwxr-x 2 deb      512 Jun 23 13:22 .
drwxrwxr-x 13 deb     7680 Jun 23 13:22 ..
-rw-rw-rwx 1 deb      54576 Apr 22 18:55 ch01
-rw-rw-rwx 1 deb      79915 Apr 22 18:55 ch02
-rw-rw-rwx 1 deb      82809 Apr 22 18:56 ch03
-rw-rw-rwx 1 deb      79434 Apr 22 18:56 ch04
-rw-rw-rwx 1 deb      85805 Apr 22 18:56 ch05
-rw-rw-rwx 1 deb      109282 Apr 22 18:56 ch06
-rw-rw-rwx 1 deb      38191 Apr 22 18:57 ch07
-rw-rw-rwx 1 deb      91235 Apr 22 18:57 ch08
-rw-rw-r-- 1 deb      611 Aug 25 1995 dickens
D -rw-rw-r-- 1 deb      315 Aug 26 1995 elephant
D -rw-rw-r-- 1 deb      418 Aug 26 1995 hJames
D -rw-rw-r-- 1 deb      865 Jun 23 13:01 iostuff.c
--%*--Dired:fiction  (Dired by name)--L17--Top-----

```

本例中给 3 个文件加上了待删除标记。

前面讲过，可以随时利用“**u**”命令去掉某个文件前面的待删除标记。按下“**u**”键，将移动到上一个加有待删除标记的文件处。也可以用**DEL**键去掉文件前面的待删除标记——这个命令的作用是去掉文件清单里前一个文件上的待删除标记并上移一行（即移动到刚被去掉待删除标记的那个文件处）。

因为 Emacs 会创建备份文件和自动保存文件，所以可能会不时地需要对它们进行一番清理。Emacs 提供了一些快捷命令来给这些文件加上待删除标记。按下“**#**”键会给所有的自动保存文件加上待删除标记（自动保存文件的文件名，其首尾各有一个“#”字符），Emacs 会给它们都加上待删除标记，并报告总共有多少个文件被加上了待删除标记。按下“**~**”键会给所有的备份文件加上待删除标记（备份文件的文件名都以“~”字符结尾）。因为这些文件只是被加上了待删除标记，所以如果还想保留某个备份文件（比如最近编辑的文件的备份拷贝），现在还来得及去掉其上的待删除标记。

等到真的想从磁盘上把文件删掉的时候，请按下“**x**”键。Emacs 将把所有被加上待删除标记的文件列出来，然后询问是否真的想删除掉它们。如下所示：

按下：**x**

```
Buffers Files Tools Edit Search Minibuf Help
-rw-rw-rwx 1 deb          91235 Apr 22 18:57 ch08
-rw-rw-r-- 1 deb          611 Aug 25 1995 dickens
D -rw-rw-r-- 1 deb         315 Aug 26 1995 elephant
D -rw-rw-r-- 1 deb         418 Aug 26 1995 hijames
D -rw-rw-r-- 1 deb         865 Jun 23 13:01 iostuff.c
-rw-rw-r-- 1 deb          280 Jun 23 13:02 joyce
-rw-rw-r-- 1 deb          280 Aug 25 1995 joyce~
-rw-rw-r-- 1 deb          53 Jan 19 09:12 letterb
-rw-rw-r-- 1 deb          226 Aug 26 1995 phone
--%*-Dired:fiction        (Dired by name)--L17--Bot-----
elephant hijames iostuff.c
--***-Emacs:*Deletions*    (Fundamental)--L17--All-----
Delete D [3 files] (yes or no) █
```

“**x**”键的作用是把文件真正地删除掉；回答“**yes**”表示“开动”。

回答“**yes**”将把它们都删除掉，而回答“**no**”将不做删除并返回 Dired 编辑缓冲区。

这是删除文件的常用办法，但如果想立刻删掉某个文件，请输入大写字母“**D**”。Emacs 将询问是否真的想删掉这个文件（**yes** 或 **no**）。回答“**yes**”将立刻删掉这个文件，回答“**no**”则不做删除。在 Dired 里，这是少数几种命令的小写字母形式（比如“**d**”用来设置待删除标记）和大写字母形式（比如“**D**”用来立刻删除某个文件）有不同含义的情况。

如果想通过 Dired 来复制某个文件，请在它旁边按下“**C**”（必须是大写字母）键或者从“**Operate**（操作）”菜单选择执行“**Copy to**（复制到）”操作。Emacs 会要求提供一个文件名作为复制操作的目标文件名。输入这个文件名再按回车键，Emacs 报告“Copied: 1 file（完成复制一个文件）”。如果想对文件清单中的多个文件进行复制，可以在大写字母“**C**”的前面加上一个数字。比如，输入“**3 C**”将复制光标位置上的文件和紧随其后的两个文件。

如果想通过 Dired 来重新命名某个文件（类似于 UNIX 中的 **mv** 命令），请在其文件名的旁边按下“**R**”键或者从“**Operate**”菜单选择执行“**Rename**”操作。Emacs 会询问新文件名是什么。输入它并按回车键，Emacs 报告“Moved: 1 file（完成移动一个文件）”。

文件的压缩和解压缩操作

对文件进行压缩可以节省出大量的磁盘空间，而Dired使压缩文件这种操作变得非常简单。把光标放到想压缩的文件所在的那一行上，然后按下“Z”（命令名是**dired-do-compress**）键或者从“Operate”菜单选择执行“**Compress**（压缩）”操作。Emacs 提问：

```
Compress or uncompress filename? (y or n)
```

如果文件还没有被压缩过，那么Emacs将对它进行压缩操作；如果已经被压缩过了，则Emacs将对它进行解压缩操作（注3）。回答“y”将开始对文件接进行压缩或解压缩操作。压缩工作是立刻进行的，所以，能够在Emacs对文件进行压缩的同时观察到文件扩展名的改变和文件长度的变化。当然，不能直接对压缩文件进行编辑，所以在对它进行编辑之前必须先对它进行解压缩操作（注4）。

对文件进行比较

上一章对分处两个窗口的文件之间的比较操作进行了讨论。Emacs在Dired里提供了一种利用UNIX的**diff**命令进行这类比较的方法。先给想用**diff**命令进行比较的文件之一设置操作标志，再把光标放到另一个文件上，然后按下“=”键或者从“**Immediate**”菜单里选择执行“**Diff**（比较）”操作。Emacs将比较这两个文件，并且会打开一个窗口——窗口中的“***diff***”编辑缓冲区里包含该命令的执行结果。

Emacs还提供了一个对某个文件及其备份文件进行比较的操作选项。把光标放在想

注3： Emacs只能识别**compress**和**gzip**格式，对PKZIP或其他专利性的文件压缩算法无能为力。对文件进行解压缩操作的时候，Emacs能够识别出后缀为“.z”、“.Z”或“.gz”的文件并正确地完成解压缩操作。在对文件进行压缩时，Emacs将使用FSF的**gzip**程序，其结果文件将以“.gz”为后缀名。

注4： Emacs有一个能够自动完成压缩/解压缩操作的自动压缩模式(**auto-compress mode**)。这个编辑模式在默认情况下是处于启用状态的，所以用不着特意去启用它。但是，在我们写这本书的时候，它的工作情况还不太正常。为了让它工作，请输入“**ESC x auto-compress-mode RETURN**”命令，这条命令的作用是关闭自动压缩功能。然后，请输入“**C-x ESC ESC RETURN**”重复执行一次这条命令。现在，自动压缩模式才真正被激活了。不过，要是让所有的文件都能被自动地压缩（以节约磁盘空间）和解压缩（以便对它们进行编辑），即使需要把这个命令重复输入两次也算不上是多大的麻烦。Emacs甚至会在完成文件的编辑工作之后再次自动对它进行压缩。

与其备份文件进行比较的文件上，然后按下“**ESC =**”组合键或者从“**Immediate**”菜单里选择执行“**Compare With Backup**（与备份文件进行比较）”操作。Emacs 将把两个文件之间的差异显示在一个“***diff***”编辑缓冲区里。

在文件上运行 UNIX 命令

在Dired里实现的**diff**命令当然很有用（Dired里还有**chmod**、**grep**和**find**等命令实现），但从更广义的方面讲，只要按下“**!**”键，就可以在文件上执行任何UNIX命令。举个例子，用**sort**命令对前面的电话号码表按字母表顺序进行排序。如下所示：

移动到*phone*文件并按“**!**”键。

```
Buffers Files Tools Search Minibuf Help
-rwxrwxrwx 1 deb          91235 Apr 22 18:57 ch08
-rw-rw-r-- 1 deb          611 Aug 25 1995 dickens
-rw-rw-r-- 1 deb          280 Jun 23 13:02 joyce
-rw-rw-r-- 1 deb          280 Aug 25 1995 joyce-
-rw-rw-r-- 1 deb          53 Jan 19 09:12 letterb
-rw-rw-r-- 1 deb          226 Aug 26 1995 phone
--*-Dired:fiction        (Dired by name)--L18--Bot-----
!on phone: [
```

Emacs询问要运行的命令。

输入：**sort**

```
Buffers Files Tools Search Operate Mark Regexp Immediate Subdir Help
-rwxrwxrwx 1 deb          91235 Apr 22 18:57 ch08
-rw-rw-r-- 1 deb          611 Aug 25 1995 dickens
-rw-rw-r-- 1 deb          280 Jun 23 13:02 joyce
-rw-rw-r-- 1 deb          280 Aug 25 1995 joyce-
-rw-rw-r-- 1 deb          53 Jan 19 09:12 letterb
-rw-rw-r-- 1 deb          226 Aug 26 1995 phone
--*-Dired:fiction        (Dired by name)--L15--Bot-----
Allen   (814) 723-8799
Alvin   (412) 833-7833
Bob     (412) 238-7732
Eileen  (717) 355-5894
Ellen   (904) 735-5894
Fred    (814) 353-3325
Horace  (814) 676-5476
Nicole  (305) 977-7721
Olivia  (617) 822-3532
--**- Emacs: Shell Command Output*      (Fundamental)--L10--All-----
```

Emacs在另外一个窗口里给出这个命令的输出结果。



稍微复杂点的例子是给出的命令要求使用一个以上的文件作为参数。请看下面这个例子：想把排好序的电话号码表保存到一个新文件里。如下所示：

把光标移动到 *phone* 文件并按 “!” 键。

Buffers	Files	Tools	Edit	Search	Minibuf	Help
-rwxrwxrwx 1 deb	91235	Apr 22	18:57	ch08		
-rw-rw-r-- 1 deb	611	Aug 25	1995	dickens		
-rw-rw-r-- 1 deb	280	Jun 23	13:02	joyce		
-rw-rw-r-- 1 deb	280	Aug 25	1995	joyce~		
-rw-rw-r-- 1 deb	53	Jan 19	09:12	letterb		
-rw-rw-r-- 1 deb	226	Aug 26	1995	phone		
--%*-Dired:fiction	(Dired by name)					--L18--Bot-----
!on phone:						

Emacs 询问要运行的命令。

现在，告诉 Emacs 要对 *phone* 文件进行排序并把其结果输出保存到一个名为 *newphone* 的新文件里。因为光标就在 *phone* 文件上，所以不需要在命令里重复输入它的名字。可以用一个星号 (*) 来代表这个文件的名字，如下所示：

输入： **sort * > newphone**

Buffers	Files	Search	Operate	Mark	Regexp	Immediate	Subdir	Help
-rwxrwxrwx 1 deb	91235	Apr 22	18:57	ch08				
-rw-rw-r-- 1 deb	611	Aug 25	1995	dickens				
-rw-rw-r-- 1 deb	280	Jun 23	13:02	joyce				
-rw-rw-r-- 1 deb	280	Aug 25	1995	joyce~				
-rw-rw-r-- 1 deb	53	Jan 19	09:12	letterb				
-rw-rw-r-- 1 deb	226	Aug 26	1995	phone				
--%*-Dired:fiction	(Dired by name)					--L17--Bot-----		
(Shell command completed with no output)								

操作系统对 *phone* 文件进行了排序，然后把执行结果输出到新文件 *newphone* 里。

我们已经创建了 *newphone* 文件，但它没有出现在屏幕画面里。要想看到 *newphone*，请按下 “g” 键。如下所示：



按下: g

Buffers	Files	Tools	Search	Operate	Mark	Regexp	Immediate	Subdir	Help
-rwxrwxrwx	1 deb						91235	Apr 22 18:57	ch08
-rw-rw-r--	1 deb						611	Aug 25 1995	dickens
-rw-rw-r--	1 deb						280	Jun 23 13:02	joyce
-rw-rw-r--	1 deb						280	Aug 25 1995	joyce~
-rw-rw-r--	1 deb						53	Jan 19 09:12	letterb
-rw-rw-r--	1 deb						211	Jun 23 13:29	newphone
-rw-rw-r--	1 deb						211	Jun 23 13:27	phone
--%*-Dired:fiction						(Dired by name)--Lib--Bot-----			
Reading directory /home/deb/fiction/...done									

Emacs 刷新了 Dired 的显示情况，把 newphone 文件赫然列在其中。

那是不是必须先按下“g”键，Dired 才刷新屏显画面呢？这不一定。有些命令（马上就要讲到）会立刻刷新屏显画面；而另外一些命令（比如部分在文件上运行的 shell 命令）就不是这样。希望大家记住下面这条操作规则：如果没有在屏幕上看到预期应该出现的东西，那最好按下“g”键试试。

对文件组进行操作

到目前为止，我们讨论的都是一次对一个文件进行操作——给出的命令只能对光标位置上的文件进行操作。Dired 的真正强大之处在于它能同时对多个文件进行操作。一旦掌握了几个快捷键，就能毫不费力地把目录管理得更好。先介绍几种用来选取文件的操作方法，然后再来说说对那些被选中的文件都能做些什么操作。

选取文件

到目前为止，我们只介绍过如何给文件加上待删除标记的方法。当想对一组文件进行一些其他的操作时，必须先给它们加上一个星号作为待操作标记。按下“m”键给光标位置上的文件加上待操作标记，在以前会出现待删除标记，即字母“D”的地方将出现一个星号。输入“3m”将给本文件和它下面的两个文件加上待操作标记。给有关文件都加上星号标记之后，Emacs 会认为此后的操作都将是以这些加有操作标记的文件为对象的。也就是说，如果给 3 个文件加上了待操作标记并按下了“Z”键做压缩操作，Emacs 会认为想对那 3 个文件进行压缩。但是，在压缩操作结束后，那 3 个文件上的待操作标记还会保留在那里。那么，怎样才能在完成了这些文件上的操作之后去掉星号标记呢？



去掉星号待操作标记的方法是：按下“**ESC DEL**”（命令名是 **dired-unmark-all-files**）组合键，或者从“**Mark (标记)**”菜单里选择执行“**Unmark All (去除全部标记)**”操作。Emacs 会询问想去掉哪些标记。按回车键，Emacs 将去掉所有的待操作标记。

用正则表达式来选取文件

经常会有需要对一组彼此相关的文件进行选取，然后再对它们进行归档、移动、压缩或者删除等操作。在 UNIX 操作系统（以及大多数其他的操作系统）里，要想选取多个文件必须使用通配符；但在 Dired 里，得使用正则表达式。用正则表达式选取多个文件的方法是：先输入一个百分号“%”，再用“m”给文件加上待操作标记；或者再用“d”给文件加上待删除标记。

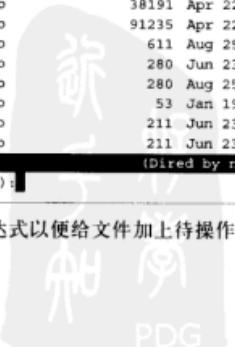
请看下面的例子。选取文件名以“ch”打头的所有文件。根据第三章里的正则表达式快速教程，我们知道用来匹配单词开头的特殊字符应该是“^”；因此，正则表达式“^ch”将把文件名以“ch”打头的所有文件都找出来。如下所示：

输入：%m

```
Buffer: $ Files Tools Edit Search Minibuf Help
/home/deb/fiction:

total 639
drwxrwxr-x 2 deb          512 Jun 23 13:29 .
drwxrwxr-x 13 deb         7680 Jun 23 13:26 ..
-rwxrwxrwx 1 deb          54576 Apr 22 18:55 ch01
-rwxrwxrwx 1 deb          79915 Apr 22 18:55 ch02
-rwxrwxrwx 1 deb          82809 Apr 22 18:56 ch03
-rwxrwxrwx 1 deb          79434 Apr 22 18:56 ch04
-rwxrwxrwx 1 deb          85805 Apr 22 18:56 ch05
-rwxrwxrwx 1 deb          109282 Apr 22 18:56 ch06
-rwxrwxrwx 1 deb          38191 Apr 22 18:57 ch07
-rwxrwxrwx 1 deb          91235 Apr 22 18:57 ch08
-rw-rw-r-- 1 deb           611 Aug 25 1995 dickens
-rw-rw-r-- 1 deb           280 Jun 23 13:02 joyce
-rw-rw-r-- 1 deb           280 Aug 25 1995 joyce-
-rw-rw-r-- 1 deb            53 Jan 19 09:12 letterb
-rw-rw-r-- 1 deb           211 Jun 23 13:29 newphone
-rw-rw-r-- 1 deb           211 Jun 23 13:27 phone
--%*-Dired:fiction      (Dired by name) - L18--All-----
Mark files (regexp):
```

Emacs 要求输入一个正则表达式以便给文件加上待操作标记。



输入: ^ch

Buffers	Files	Tools	Search	Operate	Mark	Regexp	Immediate	Subdir	Help
<code>/home/deb/fiction:</code>									
<code>total 639</code>									
	drwxrwxr-x	2	deb		512	Jun 23 13:29	.		
	drwxrwxr-x	13	deb		7680	Jun 23 13:26	..		
*	-rwxrwxrwx	1	deb		54576	Apr 22 18:55	ch01		
*	-rwxrwxrwx	1	deb		79915	Apr 22 18:55	ch02		
*	-rwxrwxrwx	1	deb		82809	Apr 22 18:56	ch03		
*	-rwxrwxrwx	1	deb		79434	Apr 22 18:56	ch04		
*	-rwxrwxrwx	1	deb		85805	Apr 22 18:56	ch05		
*	-rwxrwxrwx	1	deb		109282	Apr 22 18:56	ch06		
*	-rwxrwxrwx	1	deb		38191	Apr 22 18:57	ch07		
*	-rwxrwxrwx	1	deb		91235	Apr 22 18:57	ch08		
-rw-rw-r--	1	deb			611	Aug 25 1995	dickens		
-rw-rw-r--	1	deb			280	Jun 23 13:02	joyce		
-rw-rw-r--	1	deb			280	Aug 25 1995	joyce"		
-rw-rw-r--	1	deb			53	Jan 19 09:12	letterb		
-rw-rw-r--	1	deb			211	Jun 23 13:29	newphone		
-rw-rw-r--	1	deb			211	Jun 23 13:27	phone		
--%*-Dired:fiction					(Dired by name) --L18--All-----				
8 matching files marked.									

Emacs 给所有以“ch”打头的文件都加上选取标记，并且会告知它到底选取了多少个文件。

现在，已经有了一些加上了选取标记的文件，下面来看看都能对它们进行哪些操作。

文件组上的操作

在日常工作中，某个目录经常会因为文件过多而显得乱七八糟的。于是，决定按项目划分对它们进行归置。先建立一些子目录，再把文件分别移动到相关的子目录里去。这两件事都可以在 Dired 里完成。

假设文件名以“ch”打头的文件是一部小说的各个章节。需要一个名为“novel”的下级子目录来保存它们。于是，先按下“+”键（命令名是 **dired-create-directory**）或者从“Immediate”菜单里选择执行“Create Directory (创建目录)”操作以创建一个目录。如下所示：



按下： +

```
Buffers Files Tools Edit Search Minibuf Help
/home/deb/fiction:
total 639
drwxrwxr-x 2 deb      512 Jun 23 13:29 .
drwxrwxr-x 13 deb     7680 Jun 23 13:26 ..
* -rwxrwxrwx 1 deb     54576 Apr 22 18:55 ch01
* -rwxrwxrwx 1 deb     79915 Apr 22 18:55 ch02
* -rwxrwxrwx 1 deb     82809 Apr 22 18:56 ch03
* -rwxrwxrwx 1 deb     79434 Apr 22 18:56 ch04
* -rwxrwxrwx 1 deb     85805 Apr 22 18:56 ch05
* -rwxrwxrwx 1 deb     109282 Apr 22 18:56 ch06
* -rwxrwxrwx 1 deb     38191 Apr 22 18:57 ch07
* -rwxrwxrwx 1 deb     91235 Apr 22 18:57 ch08
-rw-rw-r-- 1 deb      611 Aug 25 1995 dickens
-rw-rw-r-- 1 deb      280 Jun 23 13:02 joyce
-rw-rw-r-- 1 deb      280 Aug 25 1995 joyce-
-rw-rw-r-- 1 deb      53 Jan 19 09:12 letterb
-rw-rw-r-- 1 deb      211 Jun 23 13:29 newphone
-rw-rw-r-- 1 deb      211 Jun 23 13:27 phone
--%*-Dired:fiction . '(Dired by name)--L18--All-----
Create directory:~/fiction/
```

Emacs 要求输入一个目录名。

输入： novel RETURN

```
Buffers Files Tools Search Operate Mark Regexp Immediate Subdir Help
/home/deb/fiction:
total 639
drwxrwxr-x 2 deb      512 Jun 23 13:29 .
drwxrwxr-x 13 deb     7680 Jun 23 13:26 ..
* -rwxrwxrwx 1 deb     54576 Apr 22 18:55 ch01
* -rwxrwxrwx 1 deb     79915 Apr 22 18:55 ch02
* -rwxrwxrwx 1 deb     82809 Apr 22 18:56 ch03
* -rwxrwxrwx 1 deb     79434 Apr 22 18:56 ch04
* -rwxrwxrwx 1 deb     85805 Apr 22 18:56 ch05
* -rwxrwxrwx 1 deb     109282 Apr 22 18:56 ch06
* -rwxrwxrwx 1 deb     38191 Apr 22 18:57 ch07
* -rwxrwxrwx 1 deb     91235 Apr 22 18:57 ch08
-rw-rw-r-- 1 deb      611 Aug 25 1995 dickens
-rw-rw-r-- 1 deb      280 Jun 23 13:02 joyce
-rw-rw-r-- 1 deb      280 Aug 25 1995 joyce-
-rw-rw-r-- 1 deb      53 Jan 19 09:12 letterb
-rw-rw-r-- 1 deb      211 Jun 23 13:29 newphone
drwxrwxr-x 2 deb      512 Jun 23 13:31 novel
-rw-rw-r-- 1 deb      211 Jun 23 13:27 phone
--%*-Dired:fiction . '(Dired by name)--L18--All-----
```

Emacs 创建一个新的子目录并把它显示在屏幕上。



接着，再把刚才加上待操作标记的那些“*ch*”文件移到这个新子目录里。用重命名命令“**R**”来做这件事。记住，Dired 的重命名命令与 UNIX 下的 **mv** 命令在执行效果方面是完全一样的。因为已经给一些文件加上星号标记了，所以，当按下“**R**”键的时候，Emacs 将认为是想对这些加有待操作标记的文件进行操作。如下所示：

按下：R

```
Buffers Files Tools Edit Search Minibuf Help
* -rwxrwxrwx 1 deb          109282 Apr 22 18:56 ch06
* -rwxrwxrwx 1 deb          38191 Apr 22 18:57 ch07
* -rwxrwxrwx 1 deb          91235 Apr 22 18:57 ch08
-rw-rw-r-- 1 deb             611 Aug 25 1995 dickens
-rw-rw-r-- 1 deb             280 Jun 23 13:02 joyce
-rw-rw-r-- 1 deb             280 Aug 25 1995 joyce~
-rw-rw-r-- 1 deb             53 Jan 19 09:12 letterrb
-rw-rw-r-- 1 deb             211 Jun 23 13:29 newphone
drwxrwxr-x 2 deb             512 Jun 23 13:31 novel
-rw-rw-r-- 1 deb             211 Jun 23 13:27 phone
--%*-Dired:fiction          (Dired by name)--L10-Bot-----
ch01 ch02 ch03 ch04 ch05 ch06 ch07 ch08
`--- Emacs:Marked Files*      (Fundamental)--L1--All-----
Move *{8 files} to: ~/fiction/
```

Emacs 询问想把加上选取标记的那些文件移到何处。

输入：novel RETURN

```
Buffers Files Tools Search Operate Mark Regexp Immediate Subdir Help
/home/deb/fiction:
total 639
drwxrwxr-x 2 deb             512 Jun 23 13:29 .
drwxrwxr-x 13 deb            7680 Jun 23 13:26 ..
-rw-rw-r-- 1 deb              611 Aug 25 1995 dickens
-rw-rw-r-- 1 deb              280 Jun 23 13:02 joyce
-rw-rw-r-- 1 deb              280 Aug 25 1995 joyce~
-rw-rw-r-- 1 deb              53 Jan 19 09:12 letterrb
-rw-rw-r-- 1 deb              211 Jun 23 13:29 newphone
drwxrwxr-x 2 deb              512 Jun 23 13:31 novel
-rw-rw-r-- 1 deb              211 Jun 23 13:27 phone
--%*-Dired:fiction          (Dired by name)--L10--All-----
Move: 8 files
```

Emacs 完成文件移动操作。

这样，那些文件就都被移到新子目录里。用正则表达式来给文件加上待操作标记的方法，能够快速选中一组文件并对它们进行各种操作。

在允许对文件组进行的操作里还有一些更令人感兴趣的东西，其中之一是只需一条命令就能对它们的内容进行全面的查询 - 替换。对大型项目来说，所谓“最后一分

钟”的修改，往往需要对很多文件里的某些特定文本进行全面的查找和替换。这个操作的具体步骤是：先把想包括在查询-替换操作里的文件都选上，然后按下“Q”（命令名是 `dired-do-query-replace`）键；接下来，先后输入查找字符串和替换字符串（这两个字符串既可以是普通文本，也可以是一个正则表达式）；Emacs 将开始一次查询-替换操作，依次进入被选中的各个文件里（注 5）。但有一点必须注意：如果在这次查询-替换操作的过程中另外开始了一次递归编辑，这次查询-替换操作就将到此为止，只能重新回到 Dired 编辑缓冲区里才能再次继续这个查询-替换操作。

正如大家所看到的，文件维护和清理方面的大部分工作都可以在 Dired 里轻松地完成。表 5-2 对 Dired 命令进行了汇总，其中有些命令或者没有介绍，或者介绍得不够细致。Dired 里需要学习的东西确实还有很多，不过，既然大家已经入了门，以后的修行还是靠自己吧。

表 5-2: Dired 命令速查表

键盘操作	命令名称	动作
C-x d <i>Files → Open Directory</i>	<code>dired</code>	启动 Dired
C <i>Operate → Copy to</i>	<code>dired-do-copy</code>	复制文件
d <i>Mark → Flag</i>	<code>dired-flag-file-deletion</code>	给文件加上待删除标记
D <i>Operate → Delete</i>	<code>dired-do-delete</code>	以问答方式立刻删除某个文件
e <i>Immediate → Find This File</i>	<code>dired-find-file</code>	编辑文件
f	<code>dired-advertised-find-file</code>	查找文件（以对它进行编辑）
g <i>Files → Revert Buffer</i>	<code>revert-buffer</code>	从磁盘上重新读入目录
G <i>Operate → Change Group</i>	<code>dired-do-chgrp</code>	改变文件的组权限

注 5：这项功能是在版本 19.29 里增加进来的。

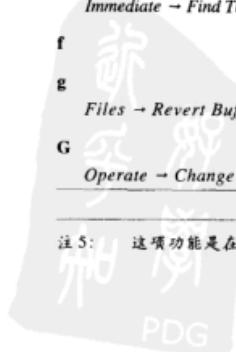


表 5-2: *Dired* 命令速查表 (续)

键盘操作	命令名称	动作
k	dired-do-kill-lines	从画面上删除光标所在的那一行 (不是删除文件)
m <i>Mark → Mark</i>	dired-mark	给文件加上“*”待操作标记
n	dired-next-line	移动到下一行
o <i>Immediate → Find in Other Window</i>	dired-find-file-other-window	在另外一个窗口里查找文件， 移动到新窗口
C-o <i>Immediate → Display in Other Window</i>	dired-display-file	在另外一个窗口里查找文件， 但不移动到新窗口
P <i>Operate → Print</i>	dired-do-print	打印文件
q	dired-quit	退出 <i>Dired</i>
Q	dired-do-query-replace	在加有待操作标记的文件里对字符串进行查找 - 替换操作
R <i>Operate → Rename to</i>	dired-do-rename	重新命名文件
u <i>Mark → Unmark</i>	dired-unmark	去掉待操作标记
v <i>Immediate → View This File</i>	dired-view-file	查看文件内容
x	dired-do-flagged-delete	删除加有待删除标记“D”的文件
Z <i>Operate → Compress</i>	dired-do-compress	对文件进行压缩或解压缩操作
ESC DEL <i>Mark → Unmark All</i>	dired-unmark-all-files	把文件上的各种待操作标记都去掉
~ <i>Mark → Flag Backup Files</i>	dired-flag-backup-files	给备份文件加上待删除标记；去掉这些标记的命令是“C-u ~”

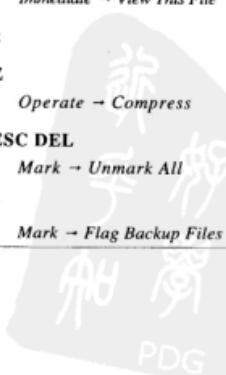


表 5-2: *Dired* 命令速查表 (续)

键盘操作	命令名称	动作
*	dired-mark-executables	给可执行文件加上“*”标记；去掉这些标记的命令是“C-u *”
#	dired-flag-auto-save-files	给自动保存文件加上待删除标记；去掉这些标记的命令是“C-u #”
-	dired-clean-directory	给带编号的备份文件（如果有的话）加上待删除标记
/	dired-mark-directories	给目录加上“*”标记；去掉这些标志的命令是“C-u /”
=	dired-diff	把这个文件与（文本标记处的）另一个文件进行比较
ESC =	dired-backup-diff	把这个文件与它的备份文件进行比较
!	dired-do-shell-command	以问答方式执行 shell 命令；命令的操作对象是加有待操作标记的文件
ESC }	dired-next-marked-file	移动到加有“*”或“D”标记的下一个文件
ESC {	dired-prev-marked-file	移动到加有“*”或“D”标记的上一个文件
%d	dired-flag-files-regexp	给匹配这个正则表达式的文件加上待删除标记
%m	dired-mark-files-regexp	给匹配这个正则表达式的文件加上待操作标记
+	dired-create-directory	创建一个目录
>	dired-next-dirline	移动到下一个目录
<	dired-prev-dirline	移动到上一个目录

表 5-2: *Dired* 命令速查表 (续)

键盘操作	命令名称	动作
s	dired-sort-toggle-or-edit	对 <i>Dired</i> 编辑缓冲区里的文件清单按日期或按文件名重新排序 (在两者之间切换)

简单的目录清单

有两个命令可以把目录的文件清单放到一个编辑缓冲区里以供查看: 命令 “**C-u C-x C-d**” 给出的是一个详细的目录文件清单, 类似于 “ls -l” 命令的执行结果; 而命令 “**C-x C-d**” 给出的是一个简单的目录文件清单, 类似于 “ls -F” 命令的执行结果——文件清单里的目录名后面跟有一个斜线字符 (/), 可执行文件名后面跟有一个星号 (*), 符号链接名后面跟有一个 "@" 字符。

与直接对文件进行操作的 *Dired* 模式不同, 这些文件清单都只是些临时性的编辑缓冲区, 可以像对待其他 Emacs 编辑缓冲区那样对它们做任意编辑——在屏幕上做的修改对底层的文件结构不会有任何影响。这个功能很适合用来打印文件目录清单——可以给文件加上一些注释或者根据打印出来的文件清单决定需要删除哪些东西。

Emacs 中的打印操作

Emacs 提供了好几个用来打印编辑缓冲区和文本块内容的命令。如果想在打印编辑缓冲区内容时加上页码、页眉和文件名, 请输入 “**ESC x print-buffer RETURN**” 或者从 “Tools” 菜单里选择执行 “**Print Buffer** (打印编辑缓冲区)” 操作。这个命令会先把整个编辑缓冲区的内容送往 pr 程序 (这是一个对文本进行简单排版的程序), 然后再送往 lpr 程序 (这个程序负责把文件内容送往打印机) (注 6)。如果想直接打印出文件, 不需要 pr 程序提供的页眉和页码, 那么请输入 “**ESC x lpr-buffer**” 命令。还可以用这些命令来打印文件中某个被选定的区域。具体做法是: 先定义一个文本块, 在文本块的一端设置一个文本标记, 然后把光标移动到文本块的另一端去; 给出命令 “**ESC x print-region RETURN**” (或命令 “**ESC x lpr-region**

注 6: 如果使用的是 System V UNIX, 与此对应的 UNIX 命令可能会稍有不同。



RETURN”或者从“Tools”菜单里选择执行“Print Region (打印文本块)”操作。

lpr-buffer和**lpr-region**命令会检查变量**lpr-switches**的取值，以判断是否还有需要向UNIX的**lpr**命令传递的其他选项参数。这些选项参数可以用来指定具体完成打印工作的打印机或者用来对许多其他事项进行设定；详细资料请参考UNIX使用手册里对**lpr**命令的介绍。比如，如果想让Emacs中的打印工作都使用名为“lpt1”的打印机来完成，就需要把变量**lpr-switches**设置为“-Plpt1”，也就是说，需要把下面这条语句添加到“*.emacs*”文件里：

```
(setq lpr-switches '("-Plpt1"))
```

请注意这条语句中的单引号、括号和字符串“-Plpt1”。这种奇怪的写法是LISP语法所要求的，详细资料请参考第十三章中的有关讨论。

也可以直接从Dired里开始打印。如果想打印光标所在位置上的文件，按下“P”或者从“Operate”菜单里选择执行“Print”操作。Emacs将把默认的打印命令及其各种必要的附加参数（比如用来指定特定打印机的参数）放到辅助输入区里。在按下回车键之前，还有机会对打印命令做一些调整。

Emacs还提供把编辑缓冲区当做一个PostScript文件打印输出的命令。如果在编辑缓冲区里使用了粗体字之类的文本属性（这方面内容可以在第十四章查到），并且想在打印这个编辑缓冲区时带上这些属性，请输入“**ESC x ps-print-buffer-with-faces**”命令或者从“Tools”菜单里选择执行“Print Postscript Buffer (按Postscript格式打印编辑缓冲区)”操作。

表5-3对Emacs在打印方面的操作命令进行了汇总。

表5-3：打印命令速查表

键盘操作	动作
ESC x print-buffer <i>Tools → Print Buffer</i>	打印编辑缓冲区（类似于UNIX中的“ pr lpr ”命令）
ESC x print-region <i>Tools → Print Region</i>	打印文本块（类似于UNIX中的“ pr lpr ”命令）
ESC x lpr-buffer	打印编辑缓冲区，但不带页码（类似于UNIX的 lpr 命令）

表 5-3: 打印命令速查表 (续)

键盘操作	动作
ESC x lpr-region	打印文本块，但不带页码（类似于 UNIX 的 lpr 命令）
ESC x direfd-do-print <i>Operate → Print</i>	在 Dired 里，把默认的打印命令放到辅助输入区里；在按回车键执行它之前还可以对它进行修改
ESC x ps-print-buffer-with-faces <i>Tools → Postscript Print Buffer</i>	打印带有文本属性的编辑缓冲区

用 Emacs 查阅 UNIX 的在线文档

Emacs 还可以用来查阅 UNIX 的在线文档（即“manpages”——UNIX 命令的使用手册页），具体做法是输入“**ESC x man**”或者从“**Help**”菜单里选择执行“**Man**”操作。这个命令将把排好版的使用手册页，放到一个新创建的编辑缓冲区里，用 Emacs 命令前后翻阅（或者复制其中的内容）。做法很简单：只要输入“**ESC x manual -entry RETURN UNIX-command-name RETURN or**”或者从“**Help**”菜单里选择执行“**Man**”操作即可。

说到 UNIX 命令的名字，既可以使用简单如“ls”这样的名字，也可以使用“**ttytab(5)**”之类的使用手册页章节号。

使用**manual-entry** 命令的好处是，可以随心所欲地前后翻阅使用手册页，这在大部分 UNIX 版本下都不容易做到。再有，如果准备在 shell 模式下查阅使用手册页，如果设置不当，它们就会显示得乱七八糟；而**manual-entry** 命令就不会出现这样的问题，它永远能显示出整齐的文本。但要注意的是：**manual-entry** 命令将运行在后台——这可能会让人觉得 Emacs 似乎没有听到指示。不过，别着急，一个显示使用手册页输出的窗口最终会在屏幕上弹出，并且能够像对待任何 Emacs 编辑缓冲区那样前后翻阅之。

时间管理工具的使用

Emacs 说不定会成为最离不开的个人信息助理。它可以把时间显示在状态行上，而

Emacs的日历和日记功能要大大优于它们的UNIX对手。Emacs的日历功能可以同时显示3个月，而它的日记功能则可以显示每天的活动和日程安排记录。

显示时间

有时工作可能需要能够随时看到时间。输入“**ESC x display-time**”命令就可以在状态行上显示一个时钟。如果想让时钟自动出现，请把下面这条语句添加到“*.emacs*”文件里：

```
(display-time)
```

这样，当下次进入 Emacs 的时候，状态行上就会自动出现一个时钟。

显示日历

如果想显示日历，请输入“**ESC x calendar**”命令。Emacs会在一个日历窗口里显示3个月（上个月、本月、下个月）的日历。如下所示：

输入： **ESC x calendar**

Buffers'Files Tools Edit Search Help													
It was the best of times,it was the worst of times,it was the age of wisdom,it was the age of foolishness,it was the epoch of belief,it was the epoch of incredulity,it was the season of Light,it was the													
-----Emacs:dickens (Text Fill) --Li-- All-----													
March 1997				April 1997				May 1997					
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
2	3	4	5	6	7	8	6	7	8	9	10	11	12
9	10	11	12	13	14	15	13	14	15	16	17	18	19
16	17	18	19	20	21	22	20	21	22	23	24	25	26
23	24	25	26	27	28	29	27	28	29	30	31	28	29
30	31											30	31
C-x < Calendar ? info/o other/. today Fri,Apr 18,1997 C-x >													

Emacs将把光标放在今天的日期上，而这个日期也将同时出现在状态行上。日历上没有书写的空间（这也就是为什么会有日记功能的原因，我们马上就会讲到日记了）。

在默认情况下，星期是从星期日开始的。如果想从星期一开始，请输入“**ESC x set-variable calendar-week-start RETURN 1 RETURN**”命令。再次进入日历时这个



设置就会生效。如果想让日历永远从星期一开始，请把下面这条语句添加到“*.emacs*”文件里：

```
(setq calendar-week-start-day 1)
```

如果想在每次进入 Emacs 的时候都能看到日历，请把下面这条语句添加到“*.emacs*”文件里：

```
(calendar)
```

在日历中移动

身处日历中的时候，Emacs 会明智地按日期而不是按字符移动。“C-f”组合键会移动到后一天；“C-b”组合键会移动到前一天；“C-n”组合键会移动到下一星期的同一天；而“C-p”组合键则会移动到上一星期去。键盘上的方向键有与此相同的作用。“ESC }”和“ESC {”以月份为单位做前后移动；“C-x]”和“C-x [”以年为单位做前后移动。“C-v”前卷三个月，“ESC v”后卷三个月。

以上讨论的这些移动命令都是相对于光标位置的移动。如果今天是星期二，按下“C-n”将移动到上一个星期二。如果今天是1月25日，按下“ESC]”将移动到2月25日。如果今天是1997年8月15日，按下“C-x [”将移动到1996年8月15日。

其他一些命令可以移动到星期、月份、年的开始或者结尾。“C-a”和“C-e”能够分别移动到星期的开始或结束；“ESC a”可移动到月份的开始；而“ESC <”则移动到年的开始。表5-4对这些日历中的移动命令进行了汇总。

如果想直接到达某个特定的日期，请使用“g d”命令。Emacs 将依次提示输入年、月、日，然后会移动到选定的日期（这个命令用来回答诸如“我在2002年的生日是星期几？”之类的问题是再合适不过了）。

表5-4：日历移动命令速查表

键盘操作	命令名称	动作
Goto → Today	calendar-goto-today	移动到今天的日期
C-f	calendar-forward-day	向前移动一天
C-b	calendar-backward-day	向后移动一天

表 5-4：日历移动命令速查表（续）

键盘操作	命令名称	动作
C-n	calendar-forward-week	向前移动一星期
C-p	calendar-backward-week	向后移动一星期
ESC }	calendar-forward-month	向前移动一个月
ESC {	calendar-backward-month	向后移动一个月
C-x]	calendar-forward-year	向前移动一年
C-x [calendar-backward-year	向后移动一年
C-a <i>Goto → Beginning of Week</i>	calendar-beginning-of-week	移动到本星期的开始
C-e <i>Goto → End of Week</i>	calendar-end-of-week	移动到本星期的结束
ESC a <i>Goto → Beginning of Month</i>	calendar-beginning-of-month	移动到本月的开始
ESC e <i>Goto → End of Month</i>	calendar-end-of-month	移动到本月的结束
ESC < <i>Goto → Beginning of Year</i>	calendar-beginning-of-year	移动到本年的开始
ESC > <i>Goto → End of Year</i>	calendar-end-of-year	移动到本年的结束
C-u n	universal-argument	重复执行随后命令 <i>n</i> 次
g d <i>Goto → Other Date</i>	calendar-goto-date	移动到指定日期
o	calendar-other-month	把指定月份放在日历画面的中间
C-x < <i>Scroll → Forward 1 Month</i>	scroll-calendar-left	前卷一个月
C-x > <i>Scroll → Backward 1 Month</i>	scroll-calendar-right	后卷一个月
SPACE	scroll-other-window	滚动另外一个窗口

显示节假日

下面再来说说人们都很感兴趣的一个话题：节假日。如果想查看当前日历画面中的

3个月里都有哪些节假日，请按下“a”键（命令名是 **list-calendar-holidays**）或者从“**Holidays**（节假日）”菜单里选择执行“**3 Months**”（3个月）操作。如下所示：

按下：a

```

Buffers Files Tools Edit Search Help
Monday, March 17, 1997: St. Patrick's Day
Thursday, March 20, 1997: Vernal Equinox 8:58am (EST)
Friday, March 28, 1997: Good Friday
Sunday, March 30, 1997: Easter Sunday
Tuesday, April 1, 1997: April Fool's Day
Sunday, April 6, 1997: Daylight Savings Time Begins 2:00am (EST)
Tuesday, April 22, 1997: Passover
Friday, May 9, 1997: Islamic New Year 1418
Sunday, May 11, 1997: Mother's Day
Monday, May 26, 1997: Memorial Day
----- Notable Dates from March to May, 1997 -----
March 1997           April 1997           May 1997
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
  1               1 2 3 4 5       1               1 2 3
  2 3 4 5 6 7 8       6 7 8 9 10 11 12   4 5 6 7 8 9 10
  9 10 11 12 13 14 15  13 14 15 16 17 18 19   11 12 13 14 15 16 17
  16 17 18 19 20 21 22  20 21 22 23 24 25 26   18 19 20 21 22 23 24
  23 24 25 26 27 28 29  27 28 29 30           25 26 27 28 29 30 31
  30 31
C-x <   Calendar   ? info/o other/. today   Fri, Apr 18, 1997   C-x >
Looking up holidays...done

```

Emacs 列出屏显日历期间内的节假日。

正如大家所看到的，Emacs 知道的节假日还真不少呢。如果在日历中的其他地方却想查看本月前后的节假日，请输入“**ESC x holidays**”命令；Emacs 会把它们列出来。如果想查看今天是不是一个节假日，请按下“**h**”键或者从“**Holidays**（节假日）”菜单里选择执行“**One Day**（当天）”操作。

按下“**x**”键会让节假日显示为特殊形式。如果可能，它们就会显示为另外一种字体或者颜色；如果不能，Emacs 将会在节假日的右边显示一个星号（*）。再按下“**u**”键将去掉这些标记。

这里介绍的只是一些基本的日历命令。Emacs 还能告知日出、日落时间和月相。可以选择查看其他种类的日历，比如玛雅历、法国革命历、伊斯兰历和犹太历等等。不过这些东西还是留给大家自己去探索吧。

日记功能的使用

日记功能与日历功能是紧密相关的，它可以给某个特定的日期加上一些文字记录。用户可以输入一个完整的日程安排，也可以只注明重大的事件。文字记录的详细程度完全由自己掌握。

创建一个日记文件

要想使用日记，必须有一个用来记录重大事件或者日程安排的“*.diary*”文件。可以用它来提醒自己每个星期四要对系统进行备份，每两星期领一次薪水，七月的前两个星期是休假，或者母亲的生日是8月6日等等。

这个文件的名字必须是“*.diary*”，并且必须存在于自己的主目录里。把自己打算记住的日期插入到这个文件里，或者说是让Emacs把它们记在里面。*.diary*文件不需要全部是统一的格式，也不需要按什么顺序进行排序。日期的格式允许混用：可以把“1997年12月19日”写成“12/19/97”、“Dec 19 97”，或者“dec 19 1997”。下面是从某个“*.diary*”文件里节选出来的几行记录，用它来说明一下日记条目的具体用法：

```
11/14 My birthday  
July 17 1997 Company picnic  
March 18 1997 Annual report due  
January 8 1997 Hair appointment  
&Saturday Tea with Queen Elizabeth  
Friday Payday
```

如果没有指定年份，Emacs将认为想把每一年的这个日期都记下来，比如“*birthday*”（生日）那一行。如果指定的是星期几而不是某个确切的日期（比如“Tea with Queen Elizabeth（与女王喝茶）”的那一行），Emacs就会在每个星期六显示这个日记条目。在某个日记条目之前放上一个“&”字符的意思是让Emacs不要在日历上把该条目显露出来（不想每个星期六上都有一个记号，或者不想让别人知道自己与皇家有什么瓜葛）。

Emacs对欧式日期格式（“DD/MM/YY”或“9 October 1997”）和美式日期格式

(“MM/DD/YY”或“October 9, 1997”)都接受，但在使用“*.diary*”文件之前(注7)，必须先把自己想用的日期格式确定下来。

添加日记条目

既可以自己写日记条目，也可以让Emacs帮助把条目放入。如果想让Emacs帮助，需要先用“**ESC x calendar**”命令进入日历功能；然后用“**g d**”命令移动到某个特定的日期；再按下“**i d**”(命令名是**insert-diary-entry**)两键或者从“**Diary**”菜单里选择执行“**Insert Daily** (插入日记)”操作。Emacs将进入日记窗口，窗口里面已经写好了日期；可以接在日期的后面写出一个日记条目。如果写的东西不止一行，请在第二行和以后每一行的开始留出一个空格，这是为了让Emacs明白它是一个续行。在写完这个日记条目之后，Emacs会留在*diary*编辑缓冲区里以便用户继续写出其他条目。如果想离开*diary*编辑缓冲区进入另一个编辑缓冲区，请按下“**C-x b**”组合键。

insert-diary-entry命令会认为只想写一条一次性的日记条目。如果想写出一个周期性的日记条目，就必须使用另外几个命令。如果想插入以一星期为一个周期的日记条目，请按下“**i w**”两键或者从“**Diary**”菜单里选择执行“**Insert Weekly** (插入周记)”操作。Emacs将进入*diary*编辑缓冲区，那里已经写好了是星期几。输入以一星期为一个周期的活动(比如员工会等)，然后保存“*.diary*”文件。如果想插入以一年为一个周期的日记条目，请按下“**i y**”两键或者从“**Diary**”菜单里选择执行“**Insert Yearly** (插入年记)”操作。Emacs进入*diary*编辑缓冲区，那里已经写好了月份和日期，然后就可以输入以一年为一个周期的事情了。

还可以输入以任意时间段为循环周期的日记条目，这些事件将按固定的时间间隔反复发生，比如用来提醒自己每3个月给汽车更换一次机油的备忘条这类的东西。具体操作方法是：先移动到上一次更换机油的日子，然后按下“**i c**”或者从“**Diary**”菜单里选择执行“**Insert Cyclic** (插入循环性事件)”操作。Emacs提问“Repeat every how many days:(每隔多少天重复一次)”，输入更换机油的天数。Emacs将写出一个负责处理此次循环时间段的LISP函数，并把它插入到*diary*编辑缓冲区，接在这个LISP函数的后面把事项写出来。比如提醒更换机油的备忘条内容。

注7： 默认的日期格式是美式日期格式。如果你想指定使用欧式日期格式，请在“*.emacs*”文件里加上“(setq european-calendar-style 't)”这条语句。

Emacs插入的日记条目如下所示(笔者的日记里就有这样一个提醒更换机油的备忘条目):

```
%{(diary-cyclic 90 12 23 1997) Change the oil}
```

这个日记条目的意思是:从插入这个条目的那一天——1997年12月23日开始,每隔90天更换一次机油。

可以把一个日期段都标记下来,比如持续一个星期的会议或者假期等。具体做法是:先把光标移动到日期段的第一天并按下“**C-SPACE**”组合键以设置一个时间标记(注8);然后移动到日期段的最后一天(要使用“**C-f**”、“**C-n**”等日历移动命令)并按下“**i b**”两键或者从“**Diary**”菜单里选择执行“**Insert Block**”操作。Emacs将进入**diary**编辑缓冲区,那里将已经插入有代表在日历上选取的那个星期的LISP函数。接在Emacs插入的那个LISP函数的后面把想要的文本写进去即可。Emacs插入的日记条目可能如下所示:

```
%{(diary-block 3 15 1997 3 20 1997) Trip to Alabama}
```

这个条目的意思是从3月15日到3月20日,要去Alabama州旅行。

如果需要在每个月的15号提交开支报告该怎么办?Emacs允许在日期里使用星号通配符(*),就像在按下“**i m**”(命令名是**insert-monthly-diary-entry**)两键时将会看到的那样。Emacs会先插入一个星号来代替月份,然后是日子。比如,如果想对每个月的15号都做出安排,日记条目里的日期设置就将是“* 15”;日记内容还像往常一样写在日期项的后面。

现在,你们已经知道Emacs是如何构造日记条目的了,大家不妨参照Emacs所做的设置试着写几条自己的日记条目。说到底,“**.diary**”文件与任何其他的Emacs文件是一样的,大家可以随心所欲地对它进行修改、插入条目或者删除条目。惟一的要

注8: 如果平时把**set-mark**命令绑定到另外一个键盘动作上或者通常拼写出那个命令,那么要想在日历里标记一个时间段就会遇到麻烦。在日历里,“**C-SPACE**”和“**C-@**”组合键将运行**calendar-set-mark**命令而不是**set-mark**命令。也就是说,所加的标记将以时间日期为对象,而不再以屏幕上的文字内容为对象。为了在日历里正确地选取一个时间段(即选取对象是时间日期而不是屏幕上的文字内容),必须用“**C-SPACE**”、“**C-@**”或者“**ESC x calendar-set-mark**”来设置时间标记。

求是必须在完成设置工作之后保存这个文件。下面，我们来看看怎样才能在到期的时候把日记条目显示出来。

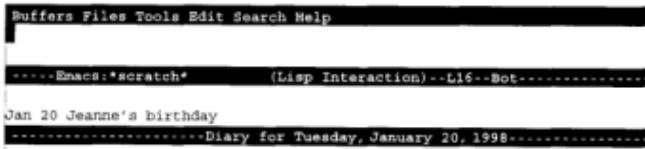
显示日记条目

如果想查看每个给定日期的日记条目，请在日历画面里按下“d”键。如果想查看整个的“.diary”文件，请在日历画面里按下“s”键。如果想在启动 Emacs 的时候让今天的日记条目自动显示出来，请把下面这条语句添加到“.emacs”文件里：

```
(diary)
```

这样，当启动 Emacs 的时候，如果当天有日记条目，那么此日记条目就将自动显示出来。举个例子，假设前些时候已经给好友的生日加上了标记，而今天恰好就是这一天。那么，当启动 Emacs 的时候，屏幕画面就可能是下面这样的：

启动 Emacs。



Emacs 把关于朋友生日的日记条目显示在屏幕上。

如果某一天没有日记条目，Emacs 也就不会显示什么日记了。如果在启动 Emacs 的时候给出了两个文件名，也就是说将在两个窗口里进行编辑工作，那么日记也不会被显示出来。

如果已经在“.emacs”文件里添加了“(calendar)”项，以便日历自动显示在屏幕上，那么日历会阻止日记的显示；如果想查看日记，就必须删除日历。

如果想把有日记条目的日期在日历上标记出来——也许是使用另外一种字体、另外一种颜色或者是在日期后面加上一个加号（+），那么请在日历画面里按下“m”键或者从“Diary”菜单里选择执行“Mark All”操作。如果想去掉这些标记，请按下“u”键或者从“Holidays”菜单里选择执行“Unmark”操作（这个命令将去掉用来突出日记条目和节假日的有关标记）。

表 5-5 对与日历和日记操作有关的命令进行了汇总。

表 5-5：日历和日记命令速查表

键盘操作	命令名称	动作
p d	calendar-print-day-of-year	显示今天是本年度的第几天（比如 365 天里的第 364 天）
SPACE	scroll-other-window	滚动另一个窗口
q	exit-calendar	退出日历功能
a <i>Holidays → 3 Months</i>	list-calendar-holidays	显示日历画面中的 3 个月里的节假日情况
h <i>Holidays → 1 Day</i>	calendar-cursor-holidays	显示今天的节假日（如果有节假日）
x <i>Holidays → Mark</i>	mark-calendar-holidays	突出显示节假日，节假日将被显示为另外一种字体、另外一种颜色或者在它们旁边加上一个星号 (*)
u <i>Holidays → Unmark</i>	calendar-unmark	去掉用来突出节假日的标记（与 x 命令的功能正好相反）
i w <i>Diary → Insert Weekly</i>	insert-weekly-diary-entry	根据此时是星期几添加一项以一星期为循环周期的日记条目
i y <i>Diary → Insert Yearly</i>	insert-yearly-diary-entry	添加一项以一年为循环周期的日记条目
i d <i>Diary → Insert Daily</i>	insert-diary-entry	为指定日期添加一项日记条目
i m <i>Diary → Insert Monthly</i>	insert-monthly-diary-entry	添加一项以一个月为循环周期的日记条目
i c <i>Diary → Insert Cyclic</i>	insert-cyclic-diary-entry	添加一项以 n 天为循环周期的日记条目
i a <i>Diary → Insert Anniversary</i>	insert-anniversary-diary-entry	添加一项以一年为循环周期的日记条目
i b <i>Diary → Insert Block</i>	insert-block-diary-entry	添加一项日期段条目

表 5-5：日历和日记命令速查表（续）

键盘操作	命令名称	动作
m	mark-diary-entries	突出显示日记条目，它们将被显示为另外一种字体、另外一种颜色或者在它们旁边加上一个加号(+)
d	view-diary-entries	显示当前日期的日记条目
s <i>Diary → Show All</i>	show-all-diary-entries	显示 ".diary" 文件的内容
ESC =	calendar-count-days-region	计算某个时间段里的天数
M <i>Moon → Lunar Phases</i>	calendar-phases-of-moon	显示 3 个月期间的月相情况
S	calendar-sunrise-sunset	根据给定的经度和纬度，显示当前日期的日出、日落时间
C-SPACE 或 C-@	calendar-set-mark	以时间日期为对象设置 (时间) 块标记，而不是按普通情况设置文本标记

用好 Emacs 工作环境

现在，我们已经知道有些工作不用离开 Emacs 也仍然能够完成。接下来的两章将对能够在 Emacs 里完成的工作做更多的介绍。第六章将介绍如何在 Emacs 里收 / 发电子邮件，以及张贴 / 阅读 Usenet 新闻。第七章讨论如何使用 Emacs 的 Telnet、FTP 和 WWW 等功能访问因特网。通过学习使用这些 Emacs 里的常用工具，大家将能够把自己更多的工作集成到 Emacs 工作环境中。



第六章

电子邮件和 Usenet 新闻

本章内容：

- Emacs 的电子邮件功能
- 用 Emacs 发送邮件
- 用 Emacs 读取邮件
- 用 Gnus 读取 Usenet 新闻

第五章里的学习内容可以分为两大块：一是如何在 Emacs 环境下对文件进行操作和处理，二是如何在环境下 Emacs 环境下与操作系统进行互动。这一章将给 Emacs 工作环境增加两个工具：电子邮件和网上新闻。可以把 Emacs 当做一个电子邮件系统和一个新闻阅读器来使用。你可能已经有一个用得很熟的电子邮件系统或新闻阅读器了。如果真是这样，你完全可以不用学习这类软件的 Emacs 版本的使用方法。可如果你还不能完全驾驭现有的电子邮件系统和新闻阅读器，或者想用 Emacs 完成大部分工作，那么就应该来学习将在这一章里介绍的应用软件。

Emacs 的电子邮件功能

这一节介绍 Emacs 发送电子邮件的基本命令，和它的电子邮件阅读器基本组件 RMAIL。Emacs 还可以使用其他一些电子邮件系统，如 MH 和 vm 等；但我们不准备在这里对它们进行讨论了。MH 电子邮件系统有一个面向 Emacs 的接口叫做 mh-e；对它感兴趣的读者可以去读读 O'Reilly & Associates 出版公司出版的《MH & mh-e: Email for users and Programmers》一书，这本书的作者是 Jerry Peek。在它的各种功能当中，mh-e 提供了 MIME 支持，因而允许图像、程序、声音以及其他数据对象用做电子邮件的附件。RMAIL 目前尚不具备这些功能。

在以下各小节里，我们将讨论向他人发送电子邮件的问题；但请注意，为了将邮件发送给正确的接收者，必须输入一个真实的电子邮件地址。如果需要定期给一些电

电子邮件地址很复杂的人发信，可以在本章“使用假名”一节里找到一个能够减轻这类负担的方法。

用 Emacs 发送邮件

从 Emacs 里向外发送电子邮件是很简单的。如果想发送一个邮件，那么请按下“**C-x m**”组合键，进入由 Emacs 创建的一个名为“*mail*”的编辑缓冲区。如下所示：

按下： **C-x m**

The screenshot shows the Emacs interface with the title bar "Buffers Files Tools Edit Search Headers Mail Help". Below it is the "*mail*" buffer containing the following text:

```
Buffers Files Tools Edit Search Headers Mail Help
To: [REDACTED]
Subject: [REDACTED]
--text follows this line--

----- Emacs: *mail* (Mail Fill) --L1--All-----
```

准备开始输入邮件的内容。

在“**To:** (收信人)”栏里填上邮件接收人的电子邮件地址；然后在下一行填上这封邮件的“**Subject (主题)**”——这一行可以不填。如果在“**To:**”栏填写了多个地址，就可以把邮件同时发送给好几个人；但不要忘了用逗号把各个地址隔开。如果收信人不止一行，别忘记在第二行和以后各行的开始加上一个空格（只要是用来填写电子邮件地址的地址栏，比如马上就要介绍到的“**CC:**”、“**BCC:**”栏等就都适用于这条规则）。

完成上述书写邮件内容前的准备工作之后，把光标移动到“**--text follows this line--**”一行的下面开始输入邮件内容，邮件内容可以用任何 Emacs 命令来编辑。千万不要删除或者修改“**--text follows this line--**”标记；Emacs 要靠它来区分邮件信头和邮件信体，如果删除，Emacs 的电子邮件功能就无法正常工作。这一行不会随邮件一起发送，它的作用只是告诉邮件的信头（收信人、主题等）在哪里结束，邮件的信体又是从哪里开始的。

在书写邮件内容的时候，可以使用 Emacs 任何一种编辑功能，比如拼写检查、单词简写模式等等。写好邮件之后，按下“**C-c C-c**”组合键或者从“**Mail**”菜单里选

择执行“Send Mail (发送邮件)”操作就可以把它发送出去。我们后面还会介绍一些更花哨的东西，但上面介绍的是发送电子邮件最基本的做法。被发送的所有邮件都会标明其发送者，并且会自动加上当时的日期和时间——虽然这些都没有在“*mail*”编辑缓冲区里显示出来。

编写邮件消息的正常编辑画面：

```
Buffers Files Tools Edit Search Headers Mail Help
To: deb, ralph, fred@grom.com, tom@jones.net, victor@hugo.org,
dorothy@sayers.com, randal@pierre.com
Subject: Upcoming Naysayers Meeting
text follows this line-
The naysayers will meet Thursday at 4:00 p.m.in the Green Room.

--***-Emacs: *mail* (Mail Fill) --L4--All-----
```

写好了一条消息，现在就可以把它发送出去。

按下： **C-c C-C**

```
Buffers Files Tools Edit Search Help
Stately, plump Jack Mulligan came from the stairhead, bearing a bowl
of lather on which a mirror and a razor lay crossed. A yellow
dressing gown, ungirdled, was sustained gently behind him by the mild
morning air. He held the bowl aloft and intoned:
--Introibo ad altare Dei
-- (Text Fill) --L1--All-----
Sending...done
```

已经把消息发送出去，Emacs 又回到刚才编辑的编辑缓冲区里。

把邮件发送出去以后，Emacs 将把邮件编辑缓冲区从屏幕上删除，然后回到刚才正在进行编辑的编辑缓冲区里——在这个例子里，就是回到那个叫做“joyce”的编辑缓冲区里。

有一件事需要特别提醒大家注意：因为“**C-c C-c**”或“**C-c C-s**”（这个命令的作用是送出邮件但仍然停留在邮件编辑缓冲区里）组合键在“*mail*”编辑缓冲区外很常用，所以很容易出现因不小心按下了这两个组合键，而错误地把一封没有写完的邮件发送出去的情况。如果在这方面总是遇到麻烦，可以先在另外一个编辑缓冲区里写好邮件的内容，等准备发送它们的时候再把它们复制到“*mail*”编辑缓冲区里。

从窗口或窗格里发送邮件

把邮件编辑缓冲区放到另外一个窗口里，而不是放到正在做其他编辑工作的当前窗口里往往更方便，也更容易在编辑缓冲区之间复制它们。在另外一个窗口里发送电子邮件的操作方法是：按下“**C-x 4 m**”组合键，填好“To:”和“Subject:”栏，然后移到下面输入邮件内容。如下所示：

按下：**C-x 4 m**

```
Buffers Files Tools Edit Search Headers Mail Help
Stately, plump Buck Mulligan came from the stairhead, bearing a bowl
of lather on which a mirror and a razor lay crossed. A yellow
dressing gown, ungirdled, was sustained gently behind him by the maid
-----Emacs:joyce          (Text Fill)--L1--All-----
To: 
Subject:
-text follows this line-
-----Emacs:*mail*          (Mail Fill)--L1--All-----
```

现在，在另一个窗口里开始输入一条新消息。

写好邮件内容后，按“**C-c C-c**”组合键把它发送出去（并退出电子邮件功能）。Emacs 会像前面介绍的那样把邮件从屏幕上删除，但它会把那个窗口留在屏幕上。如果想删掉那个此时已属多余的窗口，请按下“**C-x 0**”组合键。如果又不想发出那封邮件了，按“**C-x k**”组合键将关闭那个编辑缓冲区，同时也删掉了那封邮件。

从另一个窗格里发送电子邮件的命令对 X 用户来说也大致相同。如果想在另外一个窗格里书写邮件内容，请按下“**C-x 5 m**”组合键。发送邮件仍要像其他情况那样使用“**C-c C-c**”组合键。如果想在送出邮件之后删除那个窗格，请按下“**C-x 5 0**”组合键。

对邮件内容进行拼写检查

电子邮件是一种非常方便的交流手段。为了让工作尽快地完成，人们在书写邮件时可能会不那么正规。因此，对邮件进行拼写检查就成为一种值得培养的好习惯。如果使用的是 Ispell，请输入“**ESC x ispell-message**”命令或者从“**Spell (拼写)**”菜单里选择执行“**Check Message (检查邮件)**”操作。如果现在用的不是 Ispell，那么可以试试这种方法。虽然 UNIX 的拼写检查器 spell 也能对邮件进行拼写检查，但从使用效果上讲还是 Ispell 更好用一些。**ispell-message** 命令很聪明，它会跳过在

邮件里引述的他人邮件内容和电子邮件地址，只对书写内容进行拼写检查。与拼写检查器有关的更多内容请参考第三章中的讨论。

把邮件抄送给其他收信人

如果还想把邮件的副本发送给没有被列在“TO:”（收信人）名单里的其他人，就需要在信头部分加上一个“CC:（抄送）”行；但正常的邮件编辑模板里是不带这个“CC:”行的。因此，需要通过按下“**C-e C-f C-c**”组合键让 Emacs 创建一个“CC:”栏，或者从“**Headers**（信头）”菜单里选择执行“CC”操作，或者自己动手输入“CC:”行。自己动手输入“CC:”行的方法是：在“TO:”栏里把收信人的名字都输入好以后，按回车键创建一个空白行。然后在这个空白行上先按下“CC:”组合键，再把想让其他也收到这封邮件的人的电子邮件地址输进去——别忘了要在两个地址之间加上一个逗号。如果因地址太多而在一行里放不下，请按回车键，再先留出一个空格，然后继续输入地址。

把邮件密抄给其他收信人

如果想把邮件发送给某些人，却又不想让这些人的名字出现在信头里（即想发一封“密抄”邮件），请在信头里增加一个“BCC:（密抄）”行，然后把密抄邮件的收信人地址写到这一行上。也可以按下“**C-e C-f C-b**”组合键或者从“**Headers**”菜单里选择执行“Bcc（密抄）”操作，Emacs 会在信头里增加一个“BCC:”栏。

使用密抄的原因有很多（有的好，有的坏），什么场合该用、什么场合不该用完全由自己来决定。不过，密抄至少能够很有效地解决一个传统邮件上存在的问题——发出去的邮件没有存底。一般说来，人们很容易忘记给自己留一份外发信件的存底；这样日积月累下来，当真的需要引用以前发出的某封信件时，手里却根本没有它。把 Emacs 设置为自动给本人密抄一份邮件就可以解决这个问题。

如果想利用这种密抄功能给所有的邮件都留一份存底，那么请把 **mail-self-blind** 变量设置为“t”。这样，Emacs 就会在每一封邮件的信头部分自动加上一个“BCC:”栏，而且里面已经有你自己的用户名了。此后，如果不想要收到某特定邮件的副本，可以手动删除该邮件的“BCC:”栏。如果想永久性地把 **mail-self-blind** 变量设置为“t”，请把下面这条语句添加到 “.emacs” 文件里：

```
(setq mail-self-blind t)
```

然后，保存 “*.emacs*” 文件再重新进入 Emacs，好让这条语句生效。

如果做了这样的设置，以后只要一发送邮件，Emacs 就会自动把下面这一行：

```
BCC: yourname
```

添加到发出邮件的信头部分。其中的 *yourname* 是你的用户名。

把邮件发送到一个文件

利用“BCC：”密抄功能，能自动收到一份自己邮件的副本。还有一种方法是让 Emacs 自动地把一份邮件副本发送到某个文件——只要在信头里输入“FCC：”，再紧跟着输入一个文件名（当然还要有这个文件的完整路径），邮件就被追加到这个指定的文件里。如果使用了这项功能，则既能给自己发出的每封邮件保留一份存底，又不会因给自己发送的邮件而把邮箱弄得乱七八糟。

类似于密抄功能的设置情况，也可以让 Emacs 把邮件的副本自动发送到指定的文件里——只要设置好变量 **mail-archive-file-name** 即可。比如，如果想把自己所有外发邮件的副本都保存在主目录里一个名为“*mycopies*”的文件里，就需要把下面这条语句添加到 “*.emacs*” 文件里：

```
(setq mail-archive-file-name 'mycopies)
```

然后，退出 Emacs 再重新进入。下次发送电子邮件的时候，就会看到：

```
FCC: mycopies
```

出现在邮件的信头。如果发送了很多的邮件，那么这个文件将会变得非常大。如果磁盘空间比较紧缺，就需要定期清理自己的外发邮件，好让这个文件不至于增长得太大。

在邮件里插入一个文件

Emacs 不支持把文件用做电子邮件附件的做法，但允许把 ASCII 文本文件插入到邮件信体里。具体操作步骤是：先按下“C-x m”组合键并填写好信头；在输入信体

的时候按下“**C-x l**”组合键，然后根据提示给出一个文件名。Emacs 将把这个文件插入到信体里。可以对它进行必要的编辑，最后用“**C-c C-c**”组合键把它发送出去。

给邮件加上签名

如果想把一个标准的签名放到邮件消息里，那么先要把它保存到主目录里一个名为“*.signature*”的文件里，写完邮件内容后再按下“**C-c C-w**”组合键或者从“Mail”菜单里选择执行“**Insert Signature**（插入签名）”操作，就可以把这个文件插入到邮件消息里。（注意：有的系统可能已经设置成会自动给电子邮件加上一个“*.signature*”签名文件的情况，如果真是这样，当再按下“**C-c C-w**”组合键时，邮件里的签名就会变成两个。）许多人喜欢用一些有意思的图案来作为自己的签名（注 1），在 Emacs 里绘制图形的方法请参考第八章“绘制简单的图形”一节里对图形模式（picture mode）的讨论。另外一些人喜欢使用文字形式的签名，下面就是一种签名的格式：

Your name.	A brief quotation.
Your work address.	
Your email address.	Your phone number.

签名中的“brief quotation”通常是某种形式的声明，表明作者的看法不代表其雇主的观点。这些话一般说得都很巧妙，比如“If anyone speaks for Btsfphbl Grommets, Inc., it's certainly not me.（如果有人是 Btsfphbl Grommets 公司的发言人，那肯定不是我）”之类。

下面的例子给出了签名的使用方法。

注 1： 但我们希望大家知道，把大块的 ASCII 图案用做自己的签名并不是一个好习惯。

按下: C-c C-w

```

Buffers Files Tools Edit Search Headers Mail Help
To: david
Subject: grommets
-text follows this line-
Your ideas about grommets are completely losing.

Sincerely,
Jo[ ]
-----
Joe Btsfphbl           "It's not my fault-
Btsfphbl Grommets, Inc.   I was born obnoxious"
email: joe@grom.com      phone: (777)666-4444
-----*- Emacs: *mail* (Mail Fill)--L8--All-----*

```

“C-c C-w”组合键把签名加在邮件的末尾。

当然，如果还没有创建 “.signature” 文件，那么再怎么按 “C-c C-w” 组合键也没有用。

使用假名（别名）

如果经常要给一些地址很长的人去信，那么使用假名将会更轻松些。比如，假设想给一位名叫 Fred 的朋友写封信，而他的电子邮件地址是：

```
fred@plan9fromouterspace.galaxy.universe.com
```

如果每次写信都要输入这么一大堆字符就实在是太受罪了，但用不着！邮件假名允许给常用的地址或地址组定义一个缩略语。定义缩略语的方法是输入 “**ESC x define-mail-alias RETURN.**” 命令。Emacs 将在辅助输入区里显示一个 “Define mail alias (定义邮件假名)” 提示符；输入假名的名字（比如 **fred**），然后按下一回车键。Emacs 将给出下面这样的提示：

```
Define fred as mail alias for:
```

接着输入 Fred 的完整地址，如下所示：

```
fred@plan9fromouterspace.galaxy.universe.com
```

如果是为一个地址组定义邮件假名，顺序输入那些地址即可（地址之间要用空格隔开）。

用这种方法定义的假名只能用在当前的 Emacs 工作中，如果退出了这次 Emacs，假名就不会被保留下来。把假名永久性地定义下来当然会更有用，只需用下面这样的话句把假名命令添加到 “*.emacs*” 文件里即可：

```
(define-mail-alias 'fred "fred@plan9fromouterspace.galaxy.universe.com")
(define-mail-alias 'club "kathy ray allen fred nadine esther charles")
```

从根本上讲，邮件假名只是一个很简单的概念；但在实际应用中，它们却可能会非常复杂。问题的根源在于假名可以在很多地方进行定义：在本地的 “*.emacs*” 文件里、在其他邮件软件的定制文件里（比如 “*.mailrc*”），或者在某个系统级文件里（它通常是 “*/etc/aliases*” 文件，但也可能有别的名字）。

那么，Emacs 会到哪里去找假名呢？Emacs 会读取 “*.mailrc*” 文件寻找假名，但 “*.emacs*” 文件里的假名更优先。举个例子，如果 “*.mailrc*” 文件说 “*francis*” 是 “*francis@assisi.org*” 但 “*.emacs*” 文件却说 “*francis*” 是 “*francis@usedcars.com*”，则 Emacs 将使用后一项定义。

要想了解假名的工作原理，可以先定义一两个假名，然后用它们来发送几个邮件（别忘了把自己加到 “CC:” 栏里），看它们生成的信头是什么样的。比如，如果准备把一组地址定义为一个假名并给它起名为 “*club*”，记得要让自己是其中的一员；然后，给 “*club*” 发一封信，看看自己收到的邮件会是什么样子。收信人 “To:” 栏里根本就不会出现 “*club*” 这个词，出现的是包括在这个假名定义里的各个地址。假名可能会被扩展好几次：把 “*nadine*” 列为假名 “*club*” 的一个成员，可能会有另外一个假名把 “*nadine*” 转换为 “*nadine_fuller*”，还可能会再有一个假名把 “*nadine_fuller*” 转换为 “*nxf@orbis.com*”。

回信地址（Reply-to）栏的用法

邮件消息可以用 “*Reply-to:(回信地址)*” 栏来保证回信能够被发送到正确的地址。一般用不着关心这一栏，邮件软件基本上都能正确地生成一个供回信使用的地址。但在某些情况下，明确地设定 “*Reply-to:*” 栏是很有必要的。比如，如果系统的邮件配置有毛病，别人的回信老是莫名其妙地收不着。这种问题当然得纠正，但与邮件系统打交道并不是件容易的事情，而且这很可能根本就不在自己的管辖范围内。该怎么办呢？如果知道自己用来接收他人邮件的地址永远是 “*george@*

futons.com"，就完全可以自己动手解决这个问题——把这个地址填写在外发邮件的“Reply-to:”栏上。

再比如，如果遇到需要临时使用别人的账户（或者别人的计算机）来发送电子邮件，但想让回信还是回到你那里的情况，“Reply-to:”栏也会很有用。举个例子，假设你的计算机坏了，现在用的是一个朋友的计算机。但当你从朋友的机器上发电子邮件时，你多半还是想让回信回到自己的系统——因为在两个不同的系统上接收邮件很不方便，也许因为你估计自己的计算机不至于宕那么长的时间。

要想插入“Reply-to:”栏，可以按下“**C-c C-f C-r**”组合键或者从“**Headers**”菜单里选择执行“**Reply-To**（回信地址）”操作。如果需要经常使用这个功能，可以用设置变量**mail-default-reply-to**的方法把“Reply-to:”和回信地址自动加到外发邮件的信头里。比如，如果想把自己外发邮件里的“Reply-to:”栏全部设置为“*george@futons.com*”，就需要把下面这条语句添加到“*.emacs*”文件里：

```
(setq mail-default-reply-to "george@futons.com")
```

如果只想给某一个邮件单独加上“Reply-to:”栏，可以使用**set-variable**命令进行设置：输入“**ESC x set-variable RETURN mail-default-reply-to RETURN george@futons.com RETURN**”命令即可。

邮件发送命令汇总

表 6-1 列出了可以在 Emacs 里使用的各种邮件发送命令，“**Headers**”和“**Mail**”菜单里的有关操作选项也包括在其中。这个表里的某些命令没有在以上各小节里介绍，希望大家认真地看一看。

表 6-1：邮件发送命令速查表

键盘操作	命令名称	动作
C-x m	mail	打开“*mail*”编辑缓冲区，以邮件模板为基础填写有关资料和邮件内容
C-x 4 m	mail-other-window	在一个新窗口里打开“*mail*”编辑缓冲区

表 6-1: 邮件发送命令速查表 (续)

键盘操作	命令名称	动作
C-x 5 m	mail-other-frame	在一个新窗格里打开 “*mail*” 编辑缓冲区
C-c C-f C-t <i>Headers → To</i>	mail-to	移动到 “To:” 栏
C-c C-f C-c <i>Headers → Cc</i>	mail-cc	移动到 “CC:” 栏 (如果没有就创建之)
C-c C-f C-b <i>Headers → Bcc</i>	mail-bcc	移动到 “BCC:” 栏 (如果没有就创建之)
C-c C-f C-f <i>Headers → Fcc</i>	mail-fcc	提示输入一个文件名, 然后向这个文件发送一份邮件的副本
C-c C-f C-r <i>Headers → Reply-To</i>	mail-reply-to	指定一个地址, 而这封邮件的回信都将被发到这个地址上去
C-c C-f C-s <i>Headers → Subject</i>	mail-subject	移动到信头中的主题栏
C-c C-t <i>Headers → Text</i>	mail-text	移动到可以开始输入信体的地方去
C-c C-w <i>Mail → Insert Signature</i>	mail-signature	插入 “.signature” 文件的内容
C-c C-e <i>Mail → Send Message</i>	mail-send-and-exit	发送邮件并退出 “*mail*” 编辑缓冲区
C-c C-s <i>Mail → Send, Keep Editing</i>	mail-send	发送邮件, 但不退出 “*mail*” 编辑缓冲区
(无)	define-mail-alias	为某个名字或某个邮件表定义一个缩写的假名
(无) <i>Mail → Cancel</i>	mail-dont-send	取消正在书写的邮件消息

表 6-2 列出了可以加到邮件信头里去的各个信息栏。在默认的情况下, “*mail*” 编辑缓冲区里只显示 “To:(收信人)” 和 “Subject:(主题)” 两栏。

表 6-2：邮件信头中的信息栏

信头中的信息栏	作用
To:	收信人：此邮件将被发送给这些人
CC:	抄送：这些人将收到此邮件的一个副本
FCC:	文件抄送：此邮件的一个副本将被追加到这个文件的末尾
BCC:	密抄：这些人将收到此邮件的一个副本，但他们的名字不会出现在此邮件的信头里
Subject:	主题：此邮件信息的主题
From:	发信人：发出这封邮件的人（如果与自己的用户名不一样）；这一栏只有在用别人的账户发邮件消息时才需要填写
Reply-to:	回信地址：此邮件的回信应该被发送到的地址

用 Emacs 读取邮件

如果不能接收电子邮件，那么发出再多的电子邮件也没有什么用处。Emacs 有一个内置的编辑模式可以用来读取接收到的电子邮件。在 Emacs 里读取电子邮件的方法是：输入“**ESC x rmail RETURN**”或者从“Tools”菜单中选择执行“**Read Mail**（读取电子邮件）”操作。Emacs 将检查收信箱（inboxes）和用来保存旧邮件的文件（默认文件名是“~/RMAIL”文件）（注 2）。如果想用其他文件而不是“~/RMAIL”文件保存邮件消息，请把变量 **rmail-file-name** 设置为该文件的名字。比如，如果要把邮件消息都保存在“~/mail/inbox.rmail”文件里，就需要把下面这条语句添加到“.emacs”文件里：

```
(setq rmail-file-name "~/mail/inbox.rmail")
```

顺便告诉大家，RMAIL 文件的备份保存在“~/RMAIL~”文件里。

注 2： 收信箱（inbox）是系统用来保存新接收到的电子邮件的地方，一般用不着去操心它。Emacs 会去“/var/mail/username”，“/usr/spool/mail/username”，“/usr/mail/username”或者“/var/spool/mail/user”等地方检查有没有新邮件。如果这些文件都不存在，就说明邮件系统的设置情况与标准的不一样——问问系统管理自己的收件箱到底在什么地方。一旦找到了它，请用变量 **rmail-primary-inbox-list** 为外来邮件指定一个完整的路径名。

如果没有邮件，那么 RMAIL 会显示一条相应的信息告知。如果确实有邮件，在读完收件箱之后，Emacs 会把邮件显示在一个如图 6-1 所示的画面里。



图 6-1：用 RMAIL 读过邮件之后

从状态行上可以看出：当前编辑缓冲区叫做“RMAIL”，并且处于 RMAIL 模式中。这是一个专门用来读取邮件内容的编辑模式。上图中的“4/5”表示正在读取总共 5 条消息里的第 4 条。这个数字会根据当时有多少尚未读过的邮件而变化。最底下的那一行表示：自从上次读过邮件之后，又新收到 2 条消息，而其中的一条正显示在屏幕上。

如果没有新邮件，那么 Emacs 会把最近收到的那条消息显示在屏幕上，同时会在辅助输入区里给出一条信息“(No new mail has arrived)(没有新邮件)”。

如果收到大量的邮件，那么，当进入 RMAIL 时，就会有 10 条、20 条或者更多的新邮件在等着去读取。在这种情况下，可能需要先查看一下对各条消息进行了简单汇总的邮件清单。这方面的详细讨论请参考本章后面的“与邮件清单有关的操作”一节。

“RMAIL”编辑缓冲区除了是只读的以外，与其他 Emacs 编辑缓冲区没有什么区别。可以用“**C-x b buffername**”命令切换到其他的编辑缓冲区，或者按下“**b**”键把

“RMAIL”编辑缓冲区暂时“埋”在其他编辑缓冲区的下面。退出 RMAIL 的方法有两种：按下“q”键或者像平时那样退出 Emacs（按“C-x C-c”组合键）。

在 RMAIL 里移动

大部分标准的 Emacs 移动命令都可以用在 RMAIL 模式里，可如果需要频繁地使用 RMAIL，那么还是掌握一些用来完成面向邮件处理工作的单字符命令比较好。比如，完全能够像在其他 Emacs 文件里那样用“C-v”组合键来滚动查看一封邮件。但在 RMAIL 里，只按空格键也能达到同样的效果。

在默认情况下，RMAIL 会把所有的邮件都保存在一个文件里，其路径是“~/RMAIL”。Emacs 会把这个文件里的每条消息看做是彼此独立的事物，并且会在状态行上显示一个术语“Narrow”来表明这一点。即使所有的邮件都在该文件里，Emacs 也会把屏显画面收缩到一次只显示一条消息的程度。也就是说，即使消息的长度只有一行，它也会显示在一个独立的画面里，而信头部分将永远出现在画面的最顶部。这种设置会让我们把注意力依次集中在各条消息上。

表 6-3 列出了用在 RMAIL 里四处移动的单字符命令，以及这些命令的用处，“Move (移动)”菜单里的有关操作选项也包括在其中。

表 6-3: RMAIL 命令速查表

键盘操作	命令名称	动作
SPACE	scroll-up	卷屏，查看此消息的下一个画面
DEL	scroll-down	卷屏，查看此消息的上一个画面
.	rmail-beginning-of-message	移动到此消息的开头
n	rmail-next-undeleted-message	移动到下一条消息
<i>Move → Next</i>		
p	rmail-previous-undeleted-message	移动到上一条消息
<i>Move → Previous</i>		
<	rmail-first-message	移动到第一条消息
<i>Move → First</i>		

表 6-3: RMAIL 命令速查表 (续)

键盘操作	命令名称	动作
> <i>Move → Last</i>	rmail-last-message	移动到最后一条消息
j	rmail-show-message	如果这个命令的前面有一个数字 “n”，跳到第 n 条消息。

获取新邮件

在 RMAIL 里，可以在任何时候按下 “g” 键或者从 “Mail” 菜单里选择执行 “Get New Mail (获取新邮件)” 操作以检查自己有没有新邮件。Emacs 将检查收信箱，如果有新邮件，它就会在 RMAIL 文件存盘之后把它显示出来。

删除邮件

有用来在 RMAIL 消息之间移动的单字符命令，同样也有用来删除邮件的简单命令。

类似于删除编辑缓冲区或书签的情况(请参考前几章中的讨论)，删除邮件也要分两步走。首先、得用 “d” 或者 “C-d” 给邮件加上待删除标记 (“d” 将移动到下一条消息，而 “C-d” 将移动到前一条消息)，这一步也可以通过从 “Delete (删除)” 菜单里选择执行 “Delete” 操作来完成。这并不表示邮件消息已经无影无踪了——还来得及改变主意，用 “u” 命令去掉邮件消息上的待删除标记 (按下 “u” 键将回到上一条加有带删除标记的消息位置上，从 “Delete” 菜单里选择执行 “Undelete (撤销删除)” 操作也能到达同样的目的)。当确实想把已经加上带删除标记的邮件都删掉的时候，再只用一个操作就可以让 Emacs 把所有已经加上带删除标记的邮件都删掉。可以采取以下几种方法之一：

- 输入 “x” 命令。
- 从 “Delete” 菜单里选择执行 “Expunge (整理)” 或 “Expunge/Save (整理/保存)” 操作。
- 按下 “s” 键，保存 RMAIL 文件。
- 按下 “q” 键，退出 RMAIL。

一旦执行了上述几种操作中的任何一个，那些邮件就都将被删除，再也无法用撤销删除命令来恢复了。

表 6-4 对用来进行删除邮件操作的命令进行了汇总，“Delete”菜单里的有关操作选项也包括在其中。

表 6-4：邮件删除命令速查表

键盘操作	命令名称	动作
d <i>Delete→Delete</i>	rmail-delete-forward	给邮件加上待删除标记，然后移动到下一个
C-d	rmail-delete-backward	给邮件加上待删除标记，然后移动到上一个
ESC n <i>Move→Next</i>	rmail-next-message	移动到下一条消息；不管它是否已经加上待删除标记
ESC p <i>Move→Previous</i>	rmail-previous-message	移动到上一条消息；不管它是否已经加上待删除标记
u <i>Delete→Undelete</i>	rmail-undelete-previous-message	去掉邮件消息上的待删除标记
x <i>Delete→Expunge</i>	rmail-expunge	删除已经加有待删除标记的全部消息
s <i>Delete→Expunge/Save</i>	rmail-expunge-and-save	删除加有待删除标记的消息并保存 RMAIL 文件

回复别人的邮件

电子邮件通常被看做是电话、书信或者会议的一种替代品。它并不仅仅是一种发送给他人的备忘录，它也是一种双向的交流手段，可以有回信，甚至可以有回信的回信。知道如何对收到的邮件进行回复是很重要的。

如果想对正显示在屏幕上的消息进行回复，请按下“r”键或者从“Mail”菜单里选择执行“Reply (回信)”操作，Emacs 将打开一个“*mail*”窗口，并且会把发出回信所需要的信头信息都填写好，可以根据需要对之进行编辑以增减收信人。

在默认情况下，RMAIL 不会在回信中包括原来的信息。如果想这样做，那也很容易，只要按下“**C-c C-y**”组合键或者从“Mail”菜单里选择执行“**Cite Original**（引用原文）”操作即可。Emacs 会把正准备回信的那封邮件的内容，以段落缩进的格式插入到回信里。（段落缩进可以让人们分清哪些东西是原来的邮件内容，哪些东西是回复。）这种格式可能会使某些行超出画面，所以可能需要用“**C-c C-q**”组合键或者从“Mail”菜单里选择执行“**Fill Citation**（重排原文）”操作对引用的原文进行段落排版。如果愿意，也可以用“**C-u C-c C-y**”组合键以没有段落缩进的格式插入原邮件内容。

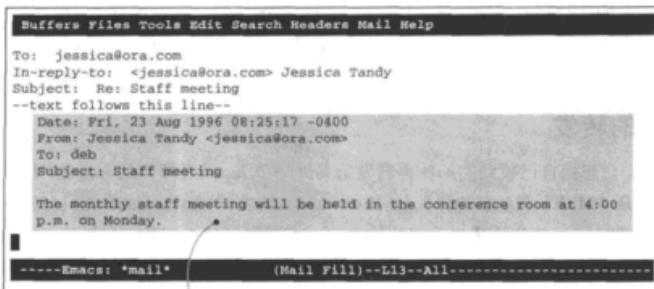
假设刚刚接收到一封关于召开员工会议的邮件，并准备发出一封回信。在读完会议通知邮件之后按下“r”键，Emacs 将打开一个如图 6-2 所示的窗口以便输入回信。



图 6-2：打开一个回信窗口

按下“**C-c C-y**”组合键插入邮件原文，如图 6-3 所示。

插入邮件原文的时候，Emacs 会删除“RMAIL”窗口，没有必要同时在屏幕上显示同一个邮件的两份副本。



按下 **C-c C-y** 组合键插入原邮件内容

图 6-3: 原邮件的内容被插入到回信里

在插入了原邮件内容之后，可以输入回信内容，编辑原来的邮件内容，或者做任何书写新邮件所需要的事情。去掉原邮件里一些不必要的内容以便回信不至于太长，是一种很好的做法。比如，如果邮件的始发者列出了 5 点意见，而你只想回复其中的 2 点，就可以把不想回复的那 3 点删掉。如果是在回复一封回信的回信（可以如此反复多次），太多层次的缩进段落反而有可能会把对方弄糊涂。只要在回信里能够把谁说过什么讲明白就足够了。引用原文的时候，可以让 Emacs 用一些更直观的东西，而不仅仅是空格来缩进正准备回复的消息。如果设置了变量 **mail-yank-prefix**，就可以让 Emacs 插入一些其他的东西来代替缩进。如果想试试这个方法，请输入 “**ESC x set-variable RETURN mail-yank-prefix RETURN**”，然后输入准备用在原文引述中的缩进字符串，比如一个大于号加一个空格之类的东西。此后，当再在回信里引用原文的时候，每一行原文的前面就都会加上 “>” 字样，从而把它们与回信的内容区别开。如果想让 **mail-yank-prefix** 在每次 Emacs 工作中都生效，就需要把下面这条语句添加到 “**.emacs**” 文件里，在引号里可以放入任何想使用的字符串：

```
(setq mail-yank-prefix "> ")
```

保存文件再重新进入 Emacs 以便这条语句生效。

写好回信之后，按下 “**C-c C-c**” 组合键，就可以把这封邮件发送出去。Emacs 将



回到原先的“RMAIL”编辑缓冲区，刚才回复的邮件仍显示在屏幕上；但此时的状态行上会出现“answered (已回复)”的字样，表示已经发出过一封回信。

邮件的转发

有时候，会想把自己收到的邮件再转发给另外一个人。转发邮件的方法是：按下“f”键或者从“Mail”菜单里选择执行“Forward (转发)”操作。Emacs会打开一个邮件编辑缓冲区，其信体是当前邮件的内容，而光标则位于“To:”栏处——直接填上准备把这份副本转发给谁即可。可以在消息里任意加上自己的评论（比如转发原因等等），然后按下“C-c C-c”组合键把这封邮件发送出去。Emacs将回到原先的“RMAIL”编辑缓冲区，并且会在这条消息的状态行上显示“forwarded (已转发)”字样。

把邮件保存到一个文件里

阅读邮件时，也许会想把某些消息保存起来以供今后参考。这件事在RMAIL里是很容易做到的，但在此之前，还得先做一个决定：是把这封邮件保存为Emacs的RMAIL格式，还是把它保存为ASCII格式。（ASCII文件就是普通的文本文件，在Emacs里编辑的大部分文件都是这种格式。）

如果只使用Emacs阅读邮件，那么RMAIL格式就是最合适的。它就是为让用RMAIL来阅读保存的邮件而设计的。比如，可以先把一封邮件保存到一个文件里，然后以后再对它进行回复。RMAIL格式能够在今后再使用RMAIL对邮件进行处理。再看ASCII格式，它更适用于那些需要把邮件转移到其他邮件系统，或者其他计算机上去的人们。其他的邮件系统也许消化不了RMAIL格式，但读取ASCII格式应该不会有太大问题。把邮件保存为RMAIL格式的方法是按下“o”键或者从“Classify (分类)”菜单里选择执行“Output (RMAIL)(输出为RMAIL格式)”操作。Emacs将提示给出一个文件名；输入文件名，再按下次回车键。如果省略了文件名，Emacs将把邮件保存到主目录里一个名为“XMAIL”的文件里。

按下: o

```
Buffers Files Tools Edit Search Minibuf Help
Date:Fri, 23 Aug 1996 08:25:17 -0400
From:Jessica Tandy <jessica@ora.com>
To:deb
Subject:Staff meeting

The monthly staff meeting will be held in the conference room at 4:00
p.m.on Monday.
----Emacs: RMAIL          (RMAIL 4/5 Narrow)--L1--All-----
Output message to Rmail file:(default XMAIL)"/
```

准备输入输出文件的名字。

把邮件保存为 ASCII 格式的方法是按下 “C-o” 键或者从 “Classify” 菜单里选择执行 “Output (inbox)(输出收信箱)” 操作。 Emacs 将提示给出一个文件名：输入文件名，再按回车键。如果省略了文件名，Emacs 将把 ASCII 格式的邮件保存到主目录里一个名为 “xmail” 的文件里。

如果是把邮件输出到一个现有的文件里，那么 Emacs 会把这条消息追加到那个文件的末尾，不会覆盖该文件的当前内容。如果因为某些原因使（比如磁盘已满） Emacs 无法保存这封邮件，它就会发出告知。

保存到文件里的任何消息都会在 RMAIL 里留一份副本。如果想让已经被保存到文件里的消息在退出 RMAIL 的时候自动删除，那么请把变量 **rmail-delete-after-output** 的值设置为 “t”。

许多 UNIX 用户会用一个特殊的子目录（通常叫做 “Mail”）来保存邮件消息。根据个人组织目录的习惯，有人可能会觉得这种做法很方便，也有人可能会觉得把邮件与其他文件混在一起更适应。

如果因为不小心按错了 “o” 和 “C-o”，而把 RMAIL 格式和 ASCII 格式弄颠倒了，那么会出什么事？不会出什么事。 Emacs 会先检查文件的格式，然后再把新邮件按与文件原有内容相同的格式追加到文件的末尾。（我们说过 Emacs 很了不起。）

此外，如果改了主意，事后又想用另外一种格式保存邮件，那么可以再把 RMAIL 格式的文件转换为 ACSII，或者再把 ASCII 格式的文件转换为 RMAIL 格式。输入 “**ESC x unrmail**” 命令就能给 RMAIL 文件另外创建一个 ASCII 版本（要为 ASCII 文件新输入一个文件名）；而用 RMAIL 读入一个 ASCII 文件（这个操作马上就要讲

到)就能把它转换为RMAIL格式。(注意:如果已经对ASCII格式的邮件文件进行过编辑,那么RMAIL能不能再次读出它来可就不能保证了。)

读取邮件文件

如果是用“o”命令把邮件保存为RMAIL格式的,那么启动RMAIL,按下“i”键,Emacs将在辅助输入区里提示:

```
Run rmail on RMAIL file:
```

输入想读入的文件名字。在读完这个文件之后,Emacs会把最后一条消息显示在屏幕上。接下来,可以用“n”和“p”来前后移动、用“r”来回复旧邮件、用“f”来转发邮件等等。

要想查看RMAIL文件的内容是很容易的。如果用“C-x C-f”组合键找到的是一个RMAIL格式的文件,Emacs会识别出它来并自动进入RMAIL模式里。

阅读以ASCII格式保存的邮件文件

如果想查看ASCII格式的邮件内容并让它保留ASCII格式,那么可以像对待任何其他文件那样用“C-x C-f”(命令名是**find-file**)组合键打开它。如果用RMAIL读入了一个ASCII格式的文件,Emacs会把它转换为RMAIL格式。如果想让它回到ASCII版本,可以使用刚才介绍的**unrmail**命令。

表6-5对文件邮件的操作命令进行了汇总,“**Classify**”菜单里的有关操作选项也包括在其中。

表6-5: 邮件文件操作命令速查表

键盘操作	命令名称	动作
o filename RETURN <i>Classify→Output (Rmail)</i>	rmail-output-to-rmail-file	把邮件消息保存为RMAIL文件格式
C-o filename RETURN <i>Classify → Output (inbox)</i>	rmail-output	把邮件消息保存为UNIX邮件文件格式(一个标准的ASCII文本文件)

表 6-5: 邮件文件操作命令速查表 (续)

键盘操作	命令名称	动作
i <i>filename</i> RETURN	rmail-input	从文件里读出邮件消息并把该文件转换为 RMAIL 格式
(无)	unrmail	创建 RMAIL 文件的 ASCII 版本

RMAIL 的查找功能

在 RMAIL 里，每个邮件都单独放在它自己的编辑缓冲区里，就好像它们是彼此分开的文件一样。在阅读邮件时，可以像平常一样使用查找命令。递增查找命令“C-s”也能正常工作，但它只能在当前邮件消息中进行查找操作，多一点也不行。

RMAIL 组件提供了一个特殊的查找命令，以便在整个邮件文件的范围内进行查找。这种扩展性的查找往往是有用的。举个例子，假设邮件文件里有 120 条消息（如果真有这么大的文件，那就应该考虑把它分割为几个小一点的文件，不过这是另外一个话题了），现在想把与“grommets”有关的消息找出来，但却想不起它是哪一条了。查看了邮件清单（我们马上就要讲到这个操作），可它并没有什么帮助——没有一条消息在它的主题栏里提到“grommets”。唯一的解决方案就剩下查找整个文件了。

于是，按下“ESC s”组合键或者从“Move (移动)”菜单里选择执行“Search (查找)”操作。Emacs 将提示输入一个正则表达式。第三章对正则表达式做过一个简要的介绍，不过即使你跳过了它或者忘了它，都没有多大关系。就这个例子而言，输入一个正常的单词就够用了。输入查找字符串，Emacs 就会把包含有这个单词的邮件消息找出来。如下所示：



按下： ESC s

```
Buffers Files Tools Edit Search Minibuf Help
Date: Fri, 23 Aug 1996 08:25:17 -0400
From: Jessica Tandy <jessicacora.com>
To: deb
Subject: Staff meeting

The monthly staff meeting will be held in the conference room at 4:00
p.m.on Monday.
---Emacs: RMAIL (RMAIL 4/5 Narrow)--L1--All
Rmail search (regexp):
```

准备输入查找字符串。

输入： grommets RETURN

```
Buffers Files Tools Edit Search Move Delete Mail Summary Classify Help
Date: Fri, 23 Aug 1996 08:29:37 -0400
From: George Jetson <george@grom.com>
To: deb
Subject: your recent order

We received your order for grommets. When shall we ship them?
---Emacs: RMAIL (RMAIL 5/5 Narrow)--L1--All
Rmail search for grommets...done
```

找到内容里有单词“grommets”的邮件。

如果再次按下“ESC s”组合键，Emacs会记得刚才的查找字符串，因而给出下面这样的提示：

```
Rmail search (regexp): (default grommets)
```

如果还想对字符串“grommets”进行查找，请直接按下回车键。Emacs会在剩下的邮件里继续查找这个单词。如果想对其他文字进行查找，输入新字符串，再按下回车键。

在开始进行查找操作之前，最好先用“j”命令跳转到第一条消息，以确保能够这次查找操作能够从头至尾地搜索整个邮件文件。但因为RMAIL里提供有逆向搜索的命令，所以这倒不是绝对必要的。如果想从当前消息开始逆向查找到文件的开始，请按下“C-u – ESC s”组合键或者从“Move”菜单里选择执行“Search Back (逆向查找)”操作。

这种查找功能有一个很大的缺点，即正向查找和逆向查找都不会对当前消息进行查找。这往往会造成一些困惑：明明看见想要找的字符串显示在屏幕上，可“ESC s”

命令就是找不到它。这可不是你的过错：“**ESC s**”总是从当前消息的结尾开始向文件末尾方向进行查找；而“**C-u - ESC s**”却又总是从当前消息的开头开始向文件头方向进行查找。不过有办法解决这个问题——递增查找（“**C-s**”）能够对当前消息进行查找，这样就不会造成遗漏了。

在第三章和第十三章里，你可以找到更多关于递增查找和正则表达式查找的内容。

与邮件清单有关的操作

与许多其他的邮件系统一样，RMAIL提供了对邮件消息清单进行操作的功能。查看邮件清单的操作方法是：利用“**h**”命令或者从“**Summary**（邮件清单）”菜单里选择执行“**All**（查看全部）”操作。Emacs 将打开一个名为“RMAIL-summary”的窗口，如图 6-4 所示。



The screenshot shows the Emacs RMAIL interface. At the top, there's a menu bar with 'Buffers', 'Files', 'Tools', 'Edit', 'Search', 'Move', 'Delete', 'Mail', 'Summary', 'Classify', and 'Help'. Below the menu is a message header:

```
Buffers Files Tools Edit Search Move Delete Mail Summary Classify Help
-----[REDACTED]-----
Date: Fri, 23 Aug 1996 08:25:17 -0400
From: Jessica Tandy <jessica@ora.com>
To: deb
Subject: Staff meeting

The monthly staff meeting will be held in the conference room at 4:00
p.m. on Friday.
```

Below the message is a list of messages from the RMAIL buffer:

```
----Emacs: RMAIL
-----[REDACTED]-----
1 23-Aug      land@ora.com Photo wanted
2 23-Aug      mark@ora.com PO number
3 23-Aug      sue@ora.com Corporate policy
4 23-Aug      jessica@ora.com Staff meeting
5 23-Aug      george@ora.com your recent order
-----[REDACTED]-----
```

At the bottom, there's another buffer labeled 'RMAIL-summary' showing a summary of the messages:

```
----Emacs: RMAIL-summary
-----[REDACTED]-----
Computing summary lines...done
-----[REDACTED]-----
```

Emacs给出的邮件消息汇总清单

图 6-4: RMAIL 的邮件清单画面

从这份清单开始，只需利用几个单字符命令就能完成对邮件消息进行查看、删除和撤销删除，以及在邮件之间移动等操作。（如果你没有跳过第四章里对编辑缓冲区清单上的操作命令的讨论，对这些命令就不应该感到陌生。）比如，如果输入用来移动到下一条消息的“**n**”命令，Emacs 就会把光标移动到邮件清单的下一行，并且会

把选中的消息显示在“RMAIL”窗口里。类似地，“**p**”将使光标移动到邮件清单的上一行，并把选中的消息自动显示出来。如果想同时退出邮件清单和RMAIL功能，请按下“**q**”键。

最新版本的Emacs在这些现有单字符命令的基础上又前进了一步，使光标移动命令基本上都能用在邮件清单上。更方便的是，递增查找命令“**C-s**”也能用在邮件清单里。这样，如果需要在一个很长的邮件清单里进行查找，那么用标准的命令就能对各种地址名、人名和邮件主题进行递增查找了。

按下“**d**”键给消息加上待删除标记，邮件清单里该条消息的前面会出现一个大写字母“**D**”，表示已经给这条消息加上了待删除标记。但这条消息并不会立刻被删除，还有机会用“**u**”来撤销它身上的待删除标记。Emacs将在按下“**x**”键时才会真正删除那些加有待删除标记的消息。按“**q**”键退出RMAIL时也会删除已加上标记的邮件。当按下“**q**”键时，Emacs将退出邮件清单模式和RMAIL；而RMAIL会在退出时把加有待删除标记的邮件都删掉。这种先给邮件加上待删除标记然后再进行删除的做法，与在Dired中从编辑缓冲区清单里删除编辑缓冲区，或者从书签表里删除书签的情况是相同的，这种不同功能操作之间的一致性使Emacs更容易学习和掌握。

在邮件清单和邮件内容之间跳来跳去可不方便，因此，Emacs允许在不离开邮件清单的情况下查看邮件的内容。如果在邮件清单里按一下“**SPACE**”键，Emacs就会把显示在“RMAIL”窗口里的邮件内容卷到下一页；如果按下的是“**DEL**”键，Emacs就会把邮件内容卷到上一页。记住：这些命令对邮件清单窗口本身不会有任何影响。在邮件消息之间移动要使用“**n**”和“**p**”键，在邮件消息的内容里前后移动要使用“**SPACE**”键和“**DEL**”键。总之，邮件清单模式有阅读邮件所需要的全部东西。

表 6-6 列出了能够在邮件清单窗口里使用的命令。

表 6-6：邮件清单操作命令速查表

键盘操作	命令名称	动作
SPACE	rmail-summary-scroll-msg-up	向前卷动 RMAIL 窗口里的邮件消息
DEL	rmail-summary-scroll-msg-down	向后卷动 RMAIL 窗口里的邮件消息

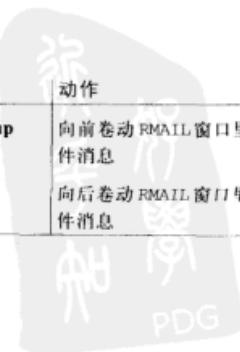


表 6-6：邮件清单操作命令速查表（续）

键盘操作	命令名称	动作
d <i>Delete→Delete</i>	rmail-summary-delete-forward	给消息加上待删除标记（在消息序号前出现字母“D”标记）
e	rmail-summary-edit-current-message	编辑当前消息（完成时要按下“C-e C-c”组合键）
u <i>Delete→Undelete</i>	rmail-summary-undelete	去掉当前消息上的待删除标记
n	rmail-summary-next-msg	移动到下一条消息并把它显示在 RMAIL 窗口里
p	rmail-summary-previous-msg	移动到上一条消息并把它显示在 RMAIL 窗口里
x <i>Delete→Expunge</i>	rmail-summary-expunge	删除所有加有待删除标记的消息
q	rmail-summary-quit	退出 RMAIL
w	rmail-summary-wipe	删除 RMAIL 邮件清单窗口

邮件的整理归类

如果邮件很多，那么就可以利用分类标签（label）对它们进行整理和归类。要想给一封邮件加上分类标签，需要输入：

```
a label RETURN
```

要想删除一个分类标签，请输入：

```
k label RETURN
```

“Classify”菜单里的“Add labels（添加标签）”和“Kill labels（删除标签）”操作也能完成同样的工作。我们来看一个例子：给员工会议通知邮件加上分类标签“meetings”。如下所示：



按下: a

```
Buffers Files Tools Edit Search Minibuf Help
Date:Fri,23 Aug 1996 08:25:17 -0400
From:Jessica Tandy <jessica@ora.com>
To:deb
Subject:Staff meeting

The monthly staff meeting will be held in the conference room at 4:00
p.m.on Monday.
---Emacs: RMAIL          (RMAIL 4/5 Narrow)--Ll--All-----
Add label: [ ]
```

准备添加一个分类标签。

看到辅助输入区里的提示信息后,输入一个分类标签作为响应。如果此前使用过“a”命令,那么 Emacs 会记得以前的分类标签,并询问是否想用它做为默认的设置值。这很适合用来给一组邮件消息加上分类标签。

添加好分类标签之后, Emacs 将把它显示在状态行上。如下所示:

输入: **meetings RETURN**

```
Buffers Files Tools Edit Search Move Delete Mail Summary Classify Help
Date: Fri,23 Aug 1996 08:25:17 -0400
From: Jessica Tandy <jessica@ora.com>
To: deb
Subject: Staff meeting

The monthly staff meeting will be held in the conference room at 4:00
p.m.on Monday.
---Emacs: RMAIL          (RMAIL 4/5:meetings'Narrow)--Ll--All-----
```

这条邮件消息加上了“meetings”分类标签。

如果现在查看 RMAIL 的邮件清单,就会发现这个标签已经在那了。如下所示:



按下: h

```
Buffers Files Tools Edit Search Move Delete Mail Summary Classify Help
Date: Fri, 23 Aug 1996 08:25:17 -0400
From: Jessica Tandy <jessica@ora.com>
To: deb
Subject: Staff meeting

The monthly staff meeting will be held in the conference room at 4:00
p.m. on Monday.
--- Emacs: RMAIL (RMAIL 4/5; meetings Narrow)--L1--All-----
1 23-Aug      land@ora.com Photo wanted
2 23-Aug      mark@ora.com PO number
3 23-Aug      sue@ora.com Corporate policy
4 23-Aug      jessica@ora.com { meetings,} Staff meeting
5 23-Aug      george@rom.com your recent order
--%*-Emacs: RMAIL-summary (RMAIL Summary)--L1--All-----
```

邮件清单已经给第 4 条消息加上了“meetings”分类标签。

可以创建一个特殊的邮件清单，让它只列出带某些特殊分类标签的邮件消息。与此对应的命令是“l”，后面必须跟着打算在邮件清单里列出的分类标签（可以有好几个）。如下所示：

按下: l

```
Buffers File* Tools Edit Search Minibuf Help
Date: Fri, 23 Aug 1996 08:25:17 -0400
From: Jessica Tandy <jessica@ora.com>
To: deb
Subject: Staff meeting

The monthly staff meeting will be held in the conference room at 4:00
p.m. on Monday.
--- Emacs: RMAIL (RMAIL 4/5; meetings Narrow)--L1--All-----
1 23-Aug      land@ora.com Photo wanted
2 23-Aug      mark@ora.com PO number
3 23-Aug      sue@ora.com Corporate policy
4 23-Aug      jessica@ora.com {meetings,}Staff meeting
5 23-Aug      george@rom.com your recent order
--%*-Emacs: RMAIL-summary (RMAIL Summary)--L1--All-----
Labels to summarize by: |
```

可以创建一个信头清单，让它里面只列出带有某些特定分类标签的邮件。



输入: **meetings RETURN**

```

Buffers Files Tools Edit Search Move Delete Mail Summary Classify Help
Date: Fri, 23 Aug 1996 08:25:17 -0400
From: Jessica Tandy <jessica@ora.com>
To: deb
Subject: Staff meeting

The monthly staff meeting will be held in the conference room at 4:00
p.m. on Monday.
--- Emacs: RMAIL      (RMAIL 4/5; meetings Narrow)--L1--All-----
| 23-Aug      jessica {meetings }Staff
10 1-Sep      george {meetings }Can't make it
15 1-Sep      deb {meetings }You lazy bum
--- Emacs: RMAIL-summary      (RMAIL Summary)--L1--All-----

```

得到的邮件清单里只列出了那些带有“meetings”分类标签的邮件消息。

这个清单与其他邮件清单没有什么区别——可以在清单里上下移动、删除邮件等等。用不着去担心那些没有列在这份清单里的邮件。也就是说，分类标签功能能够方便地对邮件消息进行归类，并且能够方便地对邮件消息组进行处理。可以依次查看所有与会议有关的消息，决定对它们的取舍。而这一切对邮箱里的其他东西不会有任何影响。

可以给一条消息加上任意多个分类标签。有时候，如果一条消息上的分类标签太多，就可能无法把它们都显示在状态行上；但只要愿意，Emacs 并不会因此而限制使用更多的分类标签。

如果想去掉某个邮件消息上的分类标签，请使用“k”命令。比如，为了去掉某个邮件消息上的“meetings”分类标签，先要用 RMAIL 把那条邮件消息显示在屏幕上，然后输入命令“**k meetings RETURN**”。

有几个内建的分类标签是由 Emacs 自动管理的，它们都列在表 6-7 里。

表 6-7：内置的邮件消息分类标签

分类标签	含义
filed	此消息已经被保存到一个邮件文件里
unseen	还没有读过这条消息
answered	已经（用“r”命令）给这条邮件消息发了一封回信
forwarded	已经把这条消息转发给别人
deleted	已经给这条消息加上了待删除标记

虽然这些内置的分类标签不会出现在信头清单里，但它们会出现在状态行上，大家可能已经注意到了。它们与其他分类标签并没有什么不同，只是不必费心去设置和删除它们而已。Emacs 会根据对邮件消息所做（或者没有做）的操作而自动设置好它们——比如，新收到的邮件消息会自动加上“unseen”分类标签，直到对它们进行了读取；而一旦读取过，Emacs 就会自动删除其“unseen”分类标签。把邮件消息保存到每个文件里时，它们会自动加上“filed”分类标签。删除“deleted”分类标签与去掉邮件消息上的待删除标记效果完全相同。“d”和“u”命令所做的事情其实就是添加和去除分类标签。（不信可以自己去试试。）类似地，给一条消息加上“unseen”分类标签，等于是告诉 Emacs 你还没有阅读过这条消息，以此类推。

对邮件清单进行排序

分类标签提供了一种对邮件进行整理和归类的办法，但 Emacs 还提供了其他一些办法。可以只查看邮件清单的一个子集，比如按收信人、正则表达式、主题或分类标签等分别查看邮件清单的某个部分。还可以用不同的办法对邮件文件本身进行排序。把这些技术组合在一起，就能轻松地对爆满的邮箱进行“减肥”。

我们刚刚讲过，按下“l”键（命令名是 **rmail-summary-by-labels**）就能按分类标签来查看邮件清单。还可以按主题（实际就是一个关键词或关键短语）来查看邮件清单。要想找出与某个主题有关的全部邮件，请按下“**ESC C-t**”（命令名是 **rmail-summary-by-topic**）组合键或者从“Summary”菜单里选择执行“**By Topic**（按主题）”操作。Emacs 将提示输入一个主题，把出现在邮件“Subject:”行里的某个单词或者短语做为关键字提供给它。Emacs 将查看全部邮件的“Subject:”行，把里面有那个关键字的邮件都挑出来列成一个邮件清单。（它不搜索邮件消息的内容，它还没那么能干。）

使用正则表达式筛选到的邮件清单就更有用了（对正则表达式的简单介绍请参考第三章）。如果想用正则表达式来查看邮件清单，可以按作者（比如所有来自 John 的邮件）、按日期（比如所有十月份的邮件），或者按主题行里的某个字符串等等来进行查找匹配。用正则表达式来查看邮件清单的方法是：按下“**ESC C-s**”（命令名是 **rmail-summary-by-regexp**）组合键或者从“Summary”菜单里选择执行“**By Regexp**（按正则表达式）”操作。按照 Emacs 的提示输入一个正则表达式（单词也

可以看做是正则表达式), 然后按下回车键。Emacs 将把信头匹配上此正则表达式的所有邮件都挑出来列成一个邮件清单。这个清单会把无法从邮件清单里查看的某些信头信息也显示出来, 比如有哪些用户收到过该邮件的抄送副本等。也就是说, 如果对与 John 有关的邮件进行查找, 除了 John 发送给的邮件外, 向 John 发送过抄送副本的邮件也会被找出来。

这几个命令虽然能够看到 RMAIL 邮件清单的某个子集, 却不能改变 RMAIL 文件的排序情况。可有时候, 确实需要对 RMAIL 文件本身进行排序。在默认情况下, 它是按照邮件到达的先后顺序来排序的, 最新收到的排在最后一个。但可以按行数、作者或者主题等项目重新进行排序。按不同顺序对邮件进行排序听起来似乎很吓人, 其实不然; 即使按其他顺序对 RMAIL 文件本身进行了排序, 也可以通过按日期重新排序的办法随时让它恢复原状。注意: 即使按其他顺序对 RMAIL 文件进行排序之后放在那里不管, Emacs 也仍然会按自己的办法对它进行管理和维护——它仍然会把最新收到的邮件追加在这个文件的末尾, 不管是否按其他顺序对该文件进行过排序。当然, 最好是在完成操作之后用 “**ESC x rmail-sort-by-date**” 命令对它重新进行排序。

假设 RMAIL 文件大得有点过头, 它里面有一个很大的邮件——比如别人邮来的一本书, 它和其他邮件混在一起。如果想把最长的邮件找出来, 请输入 “**ESC x rmail-sort-by-lines**” 命令, Emacs 将按从小到大的顺序对邮件进行排序。(这个顺序也很容易倒过来, 只要在任何排序命令前加上一个 “**C-u**”, 就可以按逆序进行排序。在这种情况下, 最长的邮件将位于邮件清单最顶部。)

还可以按作者或按主题进行排序。按主题排序的命令是 “**ESC x rmail-sort-by-subject**”; 按作者排序的命令是 “**ESC x rmail-sort-by-author**”。Emacs 将按字母表顺序对邮件清单进行排序; 先是 Fred 发来的所有邮件, 然后是 Maureen 发来的所有邮件, 再后面是 Zane 发来的所有邮件等等。

表 6-8 对用来列出邮件清单的各种命令进行了汇总, “**Summary**” 菜单里的有关操作选项也包括在其中。

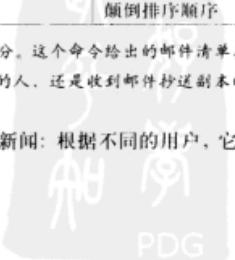


表 6-8: 邮件排序命令速查表

键盘操作	命令名称	动作
h <i>Summary→All</i>	rmail-summary	显示邮件清单
ESC C-t <i>Summary→By Topic</i>	rmail-summary-by-topic	按指定主题列出一个邮件清单；相应的查找操作是对“Subject:”栏进行的
ESC C-s <i>Summary→By Regexp</i>	rmail-summary-by-regexp	按指定正则表达式列出一个邮件清单；相应的查找操作是对信头的所有信息栏进行的
ESC C-r <i>Summary→By Recipients</i>	rmail-summary-by-recipients^a	按用户列出一个邮件清单
l <i>Summary→By Labels</i>	rmail-summary-by-labels	按分类标签列出一个邮件清单
(无)	rmail-sort-by-date	按日期对 RMAIL 文件进行排序
(无)	rmail-sort-by-subject	按主题对 RMAIL 文件进行排序
(无)	rmail-sort-by-author	按作者对 RMAIL 文件进行排序
(无)	rmail-sort-by-recipient	按收信人对 RMAIL 文件进行排序
(无)	rmail-sort-by-lines	按消息长度对 RMAIL 文件进行排序（从短到长）
(无)	rmail-sort-by-correspondents	按对应关系对 RMAIL 文件进行排序
C-u	universal-argument	颠倒排序顺序

a. 这里用“*recipient* (收信人)”一词有些误导成分。这个命令给出的邮件清单其实是按用户名排列的，不管他是发出邮件的人、收到邮件的人，还是收到邮件抄送副本的人。

熟悉邮件的使用方法之后，下一步就是 Usenet 新闻：根据不同的用户，它可能是一



种信息财富，也可能是一种时间和磁盘空间的巨大浪费。Emacs自带一个名为Gnus的新闻阅读器，它就是下面将要介绍的东西。

用 Gnus 读取 Usenet 新闻

Usenet最初是作为美国北卡罗莱纳州两所大学之间的一个公告板系统出现的。随着时间的推移，Usenet在运营站点数量和讨论组数量两方面都有非常巨大的增长。人们通常把Usenet中的讨论组称为“新闻组（newsgroup）”，它们现在的数量要以千来计算，讨论的话题从超人到超级计算机，可以说是五花八门、应有尽有。

新闻在某些方面与邮件很相似，但它们彼此之间有一些重要的区别。首先，它们用的术语就不一样：电子邮件说的是消息，新闻则说的是文章（国人经常称之为“帖子”）。发送的是邮件消息，张贴的则是自己的文章（“张贴”暗含着“这是一个公告板系统”的意思，用来描述Usenet正合适）。其次，新闻的受众和传播形式与电子邮件是截然不同的。电子邮件只会发到发信人指定的那个人手里，而Usenet新闻却会被世界各地众多的因特网站所接收——它是一个属于大众的讲坛。一篇文章哪怕只张贴在某一个新闻组里，它也可能会被成千上万的人读到。（这可能会让人们犹豫该不该向Usenet张贴稿件，但它确实是这样的。）

人们经常把新闻说成是Usenet新闻，但严格说来，这两种叫法都不准确。与Usenet没有关系却依然属于新闻范畴的新闻组是存在的。基于这个原因，有些人把Usenet新闻组更准确地称为网络新闻。“新闻”这个词本身含义就比较含糊。大多数新闻组其实都是一些讨论组，其内容与人们心目中像《华尔街日报》或《晚间新闻》之类传统意义上的新闻没有什么瓜葛。确实有少数新闻组会提供传统意义上的新闻——比如Clarinet新闻组，只能在支付服务费用的站点上才能获得。

在Emacs里就能通过Gnus阅读和张贴新闻。在Emacs 19.30里，GNUS 4.x已经升级为Gnus 5.x。有改动的地方还真不少——至少它名字里的大小写改了。此外，Gnus仍在进一步地开发当中。它是一个完备的新闻阅读器软件。如果你还在使用GNUS 4.x——那么别管什么原因，请找系统管理员讨论升级问题。

这一小节里将要介绍Gnus的基本用法。详细资料请参考Gnus的在线手册——它的主页地址是http://www.ifi.uio.no/~larsi/ding-manual/gnus_toc.html

(注 3)，或者去查阅由 Mark Harrison 撰写，O'Reilly & Associates 出版公司出版的《The USENET Handbook》(USENET 手册)一书。

启动 Gnus

Gnus 5.x 的重大改进之一是它的首次启动容易了许多。Gnus 的早期版本在首次启动时会给用户订阅所有可订阅的新闻组（光是这项工作就得用掉一个多小时的时间），而如今的 Gnus 会从先给用户订阅少量的新闻组开始，不再像以前那样让人等得着急了。如果以前阅读过新闻，那么 Gnus 将使用原有的新闻初始化文件“.newsrsrc”启动，这个文件存放在主目录里（也就是说，它的完整路径是“~/.newsrsrc”）。

启动 Gnus 的方法是输入“**ESC x gnus RETURN**”命令。Gnus 将尝试读取“.newsrsrc”文件。如果这个文件不存在，那么 Gnus 就会创建它。如下所示：

输入： **ESC x gnus RETURN**



Emacs 显示的 Gnus 初始画面。

注 3： 这堆字符串是一个 URL (Uniform Resource Locator, 统一资源定位器) 地址，这是 World Wide Web 所使用的地址形式。用我们将在第七章介绍的 Emacs 的 W3 模式就可以查阅这个手册。

画面会自动前进、用不着按任何键。可能还会出现一条信息“Garbage collecting”。这条信息并不意味着有什么新闻审查软件出现，它的意思是说内部清理工作正在进行，而这些事是用不着用户去操心的。

```

Buffers Files Tools Search Misc Groups Group Help
 4: news.announce.newusers
 111: news.groups.questions
 13: gnu.emacs.gnus
-- Gnus List of groups  {nntp:news.ora.com}  (Group)- Ll--All-----
Checking new news...done

```

Emacs给出的“Group”编辑缓冲区。

“Group”编辑缓冲区列出订阅的（如果是一位新用户，那么就是 Gnus 挑选的）所有新闻组。如果已经读过订阅的所有新闻组里的所有新闻，那么 Emacs 就会显示一条谚语“*No news is good news*”做为提示。

这个编辑缓冲区给出各新闻组的一些情况。新闻组名称前面的数字是这个新闻组里尚未阅读过的新闻条数。有些新闻组里会有成千上万篇文章，而有些却连 50 篇也到不到了。

选读一个新闻组

在“Group”编辑缓冲区里，首先得移动到想要阅读的新闻组。有以下几种方法。如果想选读的新闻组就显示在屏幕上，那么把光标移动到这一行再按下行键即可。如果知道想要选读的新闻组的名字，那么请按下“J”键（命令名是 **gnus-group-jump-to-group**）。Emacs 提示输入准备阅读的新闻组的名称。可以逐字输入它完整的名称，也可以按 TAB 键以利用 Emacs 的自动补足功能。它允许用户挑选任何一个新闻组，即使以前并没有订阅也不要紧。下面以阅读 *gnu.emacs.help* 新闻组为例来进行说明。

按下：j

```

Buffers Files Tools Edit Search Minibuf Help
 4: news.announce.newusers
 111: news.groups.questions
 13: gnu.emacs.gnus
-- Gnus List of groups  {nntp:news.ora.com}  (Group)--Ll--All-----
Groups:

```

Emacs 提示输入一个新闻组名称。

输入: gnu.emacs.help RETURN

```

Buffers Files Tools Search Misc Groups Group Help
 4: news.announce.newusers
 111: news.groups.questions
 13: gnu.emacs.gnus
K 40: gnu.emacs.help
-- Gnus List of groups  {nntp:news.ora.com}  (Group)--L4--All-----

```

Emacs 移动到 gnu.emacs.help 新闻组。

注意新闻组名称旁边的大写字母“K”，它意味着这个新闻组已经被某个初始化文件排除（注 4）；这是为了保证屏幕画面不至于被过多的新闻组弄得乱七八糟。如果想订阅这个新闻组，请使用“u”（命令名是gnus-group-unsubscribe-current-group）命令。这个命令其实是一个切换开关（如果尚未订阅，它就进行订阅；如果已经订阅，就取消订阅）。虽然“u”命令看上去不那么直观，但用起来却没问题。请继续往下看：

按下: u

```

Buffers Files Tools Search Misc Groups Group Help
 4: news.announce.newusers
 111: news.groups.questions
 13: gnu.emacs.gnus
 40: gnu.emacs.help
-- Gnus List of groups  {nntp:news.ora.com}  (Group)--L4--All-----

```

Emacs 订阅这个新闻组，并去掉它旁边的大写字母“K”标记。

要想阅读新闻组里的新闻，请按“SPACE”（命令名是gnus-group-read-group）键。如果新闻组里的新闻很多，Emacs 会让用户来决定想阅读多少篇文章。

Emacs 现在会进入“Summary”编辑缓冲（这个编辑缓冲区将在下一小节介绍）。表 6-9 对“Group”编辑缓冲区里使用的部分操作命令进行了汇总。

注 4： Gnus 的启动过程是这样的：如果你有一个“.newsr”文件，它就会读入；如果你没有“.newsr”文件，就说明你可能没有在这台机器上阅读过新闻。Gnus 现在是不会去创建“.newsr”文件的（因为这可能要花上好几个小时的时间），它会先在一个名为“.newsr.eld”的辅助启动文件里把其中的大部分新闻组排除掉，只保留少数几个。然后再把很少的几个新闻组放到你的“.newsr”文件里——你今后可以按自己的想法随意增加。用户自己是不能直接修改“.newsr.eld”文件的，有些命令能自动对它做出修改。

表 6-9: “Group” 编辑缓冲区操作命令速查表

键盘操作	命令名称	动作
(无) <i>Tools→Read Net News</i>	gnus	启动 Gnus
SPACE <i>Group→Read</i>	gnus-group-read-group	阅读光标位置处的新闻组里的文章
j <i>Groups→</i> <i>Jump to group</i>	gnus-group-jump-to-group	提示输入一个新闻组名称以转到它那里去（可以转到未曾订阅的新闻组）
n	gnus-group-next-unread-group	移动到下一个有尚未阅读过的新闻的新闻组
p	gnus-group-prev-unread-group	移动到上一个有尚未阅读过的新闻的新闻组
N	gnus-group-next-group	移动到下一个新闻组
P	gnus-group-prev-group	移动到上一个新闻组
<	beginning-of-buffer	移动到编辑缓冲区的开始
>	end-of-buffer	移动到编辑缓冲区的末尾
u <i>Group→</i> <i>Toggle subscription</i>	gnus-group-unsubscribe-current-group	订阅或者撤销订阅这个新闻组
U	gnus-group-unsubscribe-group	订阅或者撤销订阅一个指定的新闻组
c	gnus-group-catchup-current	给这个新闻组里的所有文章都加上已阅读标记并删除它们
C <i>Group→</i> <i>Catch up all articles</i>	gnus-group-catchup-current-all	给这个新闻组里的所有文章都加上已阅读标记并删除它们，包括那些带惊叹号标记（表示文章已保存）的文章
A k	gnus-group-list-killed	列出那些被“.newsr.eld”文件里的语句所排除掉的新闻组
I	gnus-group-list-groups	列出已订阅并且有新闻可读的新闻组

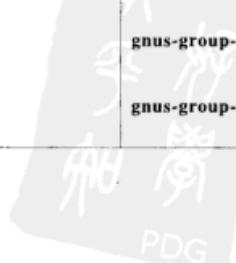
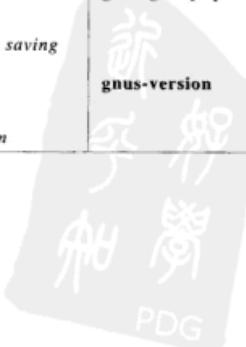


表 6-9: “Group” 编辑缓冲区操作命令速查表 (续)

键盘操作	命令名称	动作
L	gnus-group-list-all-groups	列出此服务器上的全体新闻组
g <i>Misc → Check for new news</i>	gnus-group-get-new-news	取回启动 Gnus 后新收到的新闻
R <i>Misc → Restart Gnus</i>	gnus-group-restart	读 “.newsrsrc” 文件并重新启动 Gnus (同时取回最新的新闻)
b <i>Misc → Delete bogus groups</i>	gnus-group-check-bogus-groups	找出不存在的新闻组并删除它们
a <i>Misc → Post an article</i>	gnus-group-post-news	为这个新闻组写一篇新文章
C-x C-t	gnus-group transpose-groups	交换当前行和上一行的位置
s <i>Misc → Save .newsrsrc files</i>	gnus-group-save-newsrsrc	保存 “.newsrsrc” 文件
z <i>Misc → Suspend Gnus</i>	gnus-group-suspend	临时性地挂起 Emacs 和 Gnus
q <i>Misc → Exit from Gnus</i>	gnus-group-exit	退出新闻功能并刷新 “.newsrsrc” 文件
Q <i>Misc → Exit without saving</i>	gnus-group-quit	退出新闻功能但不刷新 “.newsrsrc” 文件
V <i>Misc → Gnus version</i>	gnus-version	显示 Gnus 的版本号



阅读新闻消息

选定一个新闻组之后，Emacs 屏幕画面也将随之改变。画面的上半部分是一个“Summary”编辑缓冲区，下半部分则是一个“Article”编辑缓冲区。如下图所示：

```

Buffers Files Tools Search Post Threads Article Score Misc Help
R [ 50: Greg Kedge           | Re: Strip signature
    <29: Richard Pieri      |
    [ 33: Kevin K. Lewis     | Re: ksh-mode.el
    < 29: Zach Leber        |
- Gnu gnu.emacs.help/31001 (39 more)          (Summary)-L-
From: Greg Kedge<gkedge@ycc.Kodak.COM>
Subject: Re: Strip signature
Newsgroups: gnu.emacs.help,comp.emacs
Date: 23 Jan 1996 12:05:32 -0500
Organization: Eastman Kodak Co.Photo CD
Reply-To: gkedge@ycc.kodak.com

Greg>When I reply to a message in which I am going to yank(C-c C-y)
Greg>the original into the reply, I am interested in first stripping
- Gnu gnu.emacs.help/31001 Re: Strip signature          (Article)-L-
Loading gnus-cite...done

```

本画面分为一个“Summary”编辑缓冲区和一个“Article”编辑缓冲区。

“Summary”编辑缓冲区用来给出各篇文章的基本情况。已经读过的文章的左边会出现一个大写的字母“R”标记，表示已经读过（读过的文章会在这次新闻操作结束时从未阅读文章清单里删除）。方括号里给出的是文章里的文本行数和（如果有）作者的名字（有时候只有作者的电子邮件地址）。接着是文章的标题。在上面这个画面里，当前文章的标题是“Re: Strip signature”。类似于电子邮件方面的情况，这个“Re:”表示这篇文章是对此前一篇同名文章的回复。新闻组的名字会显示在状态行上，它后面跟着一个用斜线字符隔开的数字；这个数字是当前文章的编号，其作用相当于每一篇 Usenet 帖子的身份代码。

为什么有些行是缩进的？因为它们是“线索 (thread)”，也就是回复别人文章的文章。新闻是一场永无休止的对话，事实上，它是许多场同时进行的对话。阅读不带线索的新闻就好像是在一间非常大的屋子里听很多人同时在交谈一样，没有头绪；而沿着一条交谈线索去听就方便多了。Gnus 是一个线索化的新闻阅读器，线索是被默认启用的——可以用“**ESC C-t**”（命令名是 **gnus-summary-toggle-threads**）组合键关闭它们。因为线索的缘故，某篇文章的所有回复都会附在该文章的后面。有

时候，甚至会有针对回复文章的回复。在上面的例子中，Richard Pieri 针对 Greg Kedge 的帖子写了一篇回复文章。

用不着移动到“Article”编辑缓冲区就可以读到新闻，而事实是也不应该那样做。Emacs 为“Summary”编辑缓冲区准备的命令，可以不用移动到“Article”编辑缓冲区，就能看到新闻的内容。“Article”编辑缓冲区里只有很少几条命令可供使用（比如，用来后翻新闻内容的 **SPACE** 键，或用来前翻新闻内容的 **DEL** 键等），而这些命令在“Summary”编辑缓冲区里使用时也能达到同样的效果。

在正常情况下，按“**n**”键（命令名是 **gnus-summary-next-unread-article**）移动到下一篇新闻，按“**p**”键（命令名是 **gnus-summary-prev-unread-article**）移动到上一篇新闻；这两个命令的作用是在未阅读过的文章中间移动。如果因“**n**”键按得太快而跑过了头，就需要向回移动去阅读此前的文章，哪怕 Emacs 认为已经读过它了。返回并阅读前一篇文章的方法是按下“**P**”键（命令名是 **gnus-summary-prev-article**）；这条命令将不管是否已经阅读过前一篇文章而返回到那里。另外一个方法是按下“**I**”键（小写的字母“L”，命令名是 **gnus-summary-goto-last-article**），它的作用是返回到刚才读过的最后一篇文章。现在，我们准备阅读下一篇新闻。

按下： **n**

```
Buffers Files Tools Search Post Threads Article Score Misc Help

R [ 50: Greg Kedge ] Re: Strip signature
R < 29: [Richard Pieri >
[ 33: Kevin K.Lewis ] Re: ksh-mode.el
< 29: Zach Leber >
- Gnus gnu.emacs.help/31013 (38 more) (Summary)-L2-
From: Richard Pieri <ratinox@unilab.dfc.harvard.edu>
Subject: Re: Strip signature
Newsgroups: gnu.emacs.help,comp.emacs
Date: 24 Jan 1996 11:27:34 -0500
Organization: Dana-Farber Cancer Institute
-----BEGIN PGP SIGNED MESSAGE-----
>>>>"g" --- gkedge <Greg> writes:
g>This better should look for "~- $$" (note the space after the
g>dashes).
I'd have to double-check the RFC (yes, there *IS* an RFC that defines
the signature delimiter) but I believe that "~- 0{rq is a better regular
expression to match on.
- Gnus gnu.emacs.help/31013 Re: Strip signature (Article)-L20-
```

Emacs 把下一篇新闻显示到屏幕上。

现在，第一篇和第二篇文章都已经加上已阅读标记“R”；离开新闻功能的时候它们将会被删除。光标则向下移动到当前文章处。

读过几篇文章之后，你可能会认为其余的文章都不值得一读了。这时，可以按下“e”键（命令名是 **gnus-summary-catchup-and-exit**）结束这次阅读新闻工作。Emacs 将提问：

```
Mark all unread articles as read?
```

如果回答“y”，Emacs 就将给清单里所有还未曾阅读过的文章（仅限于这个新闻组）都加上已阅读标记，然后回到“Group”编辑缓冲区。

另外一种情况是发现有些文章很有意思，值得保存下来。可是，Gnus 会给阅读过的文章都加上已阅读标记，并且会在退出时删除它们。要想保存某篇文章，必须明确地给它加上未阅读标记——即告诉 Gnus 要把它留在文章清单里等待阅读。保存当前文章的方法是按下“u”键（命令名是 **gnus-summary-tick-article-forward**）。Emacs 将把字母“R”标记替换为一个惊叹号。这样，当下次进入这个新闻组时，这篇文章就还会保留在那里。如果想把文章保存到一个文件里（而不是让它留在新闻组里），请按下“C-o”组合键把它保存到一个 ASCII 文件里，或者按下“o”键把它保存为 RMAIL 格式。

如果已经把某个新闻组里的文章都读过了，Emacs 会询问是否还想前进到下一个新闻组；如果已经没有别的新闻组可读，Emacs 就会询问是否想退出。可以用“q”键（命令名是 **gnus-summary-exit**）随时返回到“Group”编辑缓冲区。

下一小节将讨论怎样才能查看到所有可供阅读的新闻组，过滤（kill）文件又该怎样使用，以及如何向新闻组投稿等事项。表 6-10 列出了一些可以在“Summary”编辑缓冲区里使用的命令。

表 6-10：“Summary” 编辑缓冲区操作命令速查表

键盘操作	命令名称	动作
SPACE	gnus-summary-next-page	文章前卷
DEL	gnus-summary-prev-page	文章后卷
RETURN	gnus-summary-scroll-up	文章前卷，每次一行

表 6-10: “Summary” 编辑缓冲区操作命令速查表 (续)

键盘操作	命令名称	动作
.	gnus-summary-first-unread-article	移动到此新闻组里尚未读过的第一篇文章处
< Article → <i>Beginning of the article</i>	gnus-summary-beginning-of-article	移动到当前文章的开始
> Article → <i>End of the article</i>	gnus-summary-end-of-article	移动到当前文章的末尾
n	gnus-summary-next-unread-article	移动到下一篇文章
N	gnus-summary-next-article	移动到下一篇文章 (即使已经读过)
p	gnus-summary-prev-unread-article	移动到上一篇文章
P	gnus-summary-prev-article	移动到上一篇文章 (即使已经读过)
I	gnus-summary-goto-last-article	移动到刚读过的最后一篇文章 (如果移动得太快, 可以用这条命令返回去)
H f Misc → Fetch group FAQ	gnus-summary-fetch-faq	取回这个新闻组的常见问题答疑文件
ESC C-t Threads → Toggle threading	gnus-summary-toggle-threads	打开 / 关闭线索功能
ESC C-k Threads → Mark thread as read	gnus-summary-kill-thread	排除当前线索, 包括其子线索
ESC C-d Threads → Go down thread	gnus-summary-down-thread	移动到这个线索的下一篇文章处



表 6-10: “Summary” 编辑缓冲区操作命令速查表 (续)

键盘操作	命令名称	动作
ESC C-u <i>Threads→</i> <i>Go up thread</i>	gnus-summary-up-thread	移动到这个线索的上一篇文章处
ESC C-f <i>Threads→</i> <i>Go to next thread</i>	gnus-summary-next-thread	移动到下一个线索 (即移动到另外一个主题)
ESC C-b <i>Threads→</i> <i>Go to previous thread</i>	gnus-summary-prev-thread	移动到上一个线索 (即移动到另外一个主题)
ESC C-h <i>Threads→Hide thread</i>	gnus-summary-hide-thread	隐藏当前线索
ESC C-s <i>Threads→</i> <i>Display hidden thread</i>	gnus-summary-show-thread	显示当前线索
q	gnus-summary-exit	返回“Group”编辑缓冲区
Q	gnus-summary-exit-no-update	返回“Group”编辑缓冲区，但不删除读过的文章
c <i>Group→Catch up</i>	gnus-summary-catchup-and-exit	给此新闻组里的文章都加上已阅读标记并返回“Newsgroup”编辑缓冲区
u	gnus-summary-tick-article-forward	给当前文章加上未阅读标记，在它旁边放上一个惊叹号以保留它供今后阅读。如果重复输入此命令，则保存下一篇文章
U	gnus-summary-tick-article-backward	给当前文章加上未阅读标记，在它旁边放上一个惊叹号以保留它供今后阅读。如果重复使用此命令，则保存上一篇文章
C-o	gnus-summary-save-article-mail	以 UNIX 格式保存当前文章

表 6-10: “Summary” 编辑缓冲区操作命令速查表 (续)

键盘操作	命令名称	动作
o	gnus-summary-save-article	以RMAIL格式保存当前文章
d	gnus-summary-mark-as-read-forward	给当前文章加上已阅读标记，从当前行开始向下移动
D	gnus-summary-mark-as-read-backward	给当前文章加上已阅读标记，从当前行开始向回移动
j	gnus-summary-goto-subject	要求指定在屏幕上显示的文章篇数
ESC n	gnus-summary-next-unread-subject	移动到下一个尚未读过的主题
ESC p	gnus-summary-prev-unread-subject	移动到上一个尚未读过的主题
ESC C-n	gnus-summary-next-same-subject	移动到同一主题的下一篇文 章
ESC C-p	gnus-summary-prev-same-subject	移动到同一主题的上一篇文 章
m <i>Post→Send a mail</i>	gnus-summary-mail-other-window	打开一个电子邮件消息的编 辑缓冲区
C-c C-f	gnus-summary-mail-forward	把这篇文章的副本发送给某 人
=	gnus-summary-expand-window	扩展“Summary”窗口，让 它充满整个屏幕
g <i>Article→Redisplay</i>	gnus-summary-show-article	显示当前文章（特别适用于 扩展了的“Summary”窗口或 者需要在这个窗口里移动的 情况）
s <i>Article→Isearch article</i>	gnus-summary-isearch-article	用isearch功能查找当前文章 中的文本
ESC s <i>Article→Search all articles</i>	gnus-summary-search-article-forward	对当前文章进行正方向的正 则表达式查找



表 6-10: “Summary” 编辑缓冲区操作命令速查表 (续)

键盘操作	命令名称	动作
ESC r	<code>gnus-summary-search-article-backward</code>	对当前文章进行反方向的正则表达式查找
t	<code>gnus-summary-toggle-header</code>	打开/关闭文章信头的详细显示状态
w	<code>gnus-summary-stop-page-breaking</code>	不在文章中出现换页符的地方进行分页
x	<code>gnus-summary-remove-lines-marked-as-read</code>	把带已阅读标记的文章全部删除
C-c TAB	<code>gnus-info-find-node</code>	显示 Gnus 的 Info 帮助信息
C-c C-r	<code>gnus-summary-caesar-message</code>	加密/解密这篇文章
C-x C-s	<code>gnus-summary-reselect-current-group</code>	重启这个新闻组, 不给文章加上已阅读标记
ESC t	<code>gnus-summary-toggle-mime</code>	进入 MIME 模式
ESC U	<code>gnus-summary-clear-mark-backward</code>	清除当前行上所有标记, 包括连字符、排除标记“K”、待删除标记“D”。如果重复输入这个命令, 则清除前一行上的标记
ESC u	<code>gnus-summary-clear-mark-forward</code>	清除当前行上所有标记。如果重复使用这个命令, 则清除后一行上的标记
C-t <i>Misc → Toggle line truncation</i>	<code>gnus-summary-toggle-truncation</code>	打开/关闭针对长文本行的截断功能
& <i>Misc → Filter articles</i>	<code>gnus-summary-execute-command</code>	在信头部分里查找指定的正则表达式
	<code>gnus-summary-pipe-output</code>	把这篇文章经管道输出到一个子进程去, 比如送去打印等

表 6-10: “Summary” 编辑缓冲区操作命令速查表 (续)

键盘操作	命令名称	动作
C-d Article→ Enter digest buffer	gnus-summary-rmail-digest	如果这篇文章是一份文摘，分析并阅读之

列出所有的新闻组

在默认情况下, Gnus 会把新闻组的名单从 (典型的) 10,000 个以上裁减为只剩下几个。可好奇的用户可能想去看看到底有多少新闻组是可用的。为此, 你必须在“Group”编辑缓冲区里 (如果还待在“Summary”编辑缓冲区里, 请输入“**C-x b *Group***”命令进入“Group”编辑缓冲区) 进行操作。查看全部新闻组名单的方法是连续输入“A k”(命令名是 **gnus-group-list-killed**)。如下所示:

输入: A k

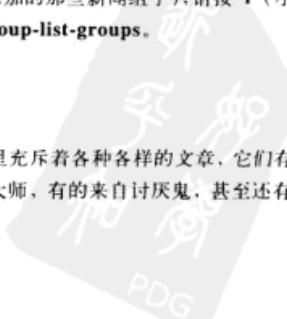
```
Buffers' Files Tools Search Misc Groups Group Help
K   *: alt.Cajun.info
K   *: alt.abortion.inequity
K   *: alt.abuse.offender.recovery
K   *: alt.abuse.recovery
K   *: alt.activism
K   *: alt.activism.d
K   *: alt.activism.death-penalty
K   *: alt.adoption
-- Gnus List of groups  (nntp:news.ora.com)  (Group)--L1--Top-----
A k
```

Gnus 把全部可用的新闻组都列出来。

要想找出自己感兴趣的新闻组, 可以用“C-s”组合键对之进行查找, 或者就老老实实地翻看这份名单。如果想订阅某个新闻组, 请按下“u”键。如果想返回自己已经订阅的新闻组名单(那份名单现在已经包括刚刚添加的那些新闻组了), 请按“l”(小写的“L”字母) 键, 它对应的命令名是 **gnus-group-list-groups**。

过滤文件 (kill file) 的使用

Usenet 里的帖子反映了各种各样的人性。新闻组里充斥着各种各样的文章, 它们有的来自智者, 有的来自普通大众, 有的来自幽默大师, 有的来自讨厌鬼, 甚至还有



些是来自心理阴暗的人。几乎每个新闻组里都有那么一两个臭名昭著的人，他们不停地张贴文章，一会儿是想卖些谁也不想买的东西，一会儿是散布一些与这个新闻组里大多数人观点对立的言论。这就是为什么会有过滤文件的原因——它们能有选择地对某些人或话题说“到此为止！”。过滤文件是用LISP命令写成的，它会告诉Gnus哪些东西是不想看到的。Gnus会自动排除那些不想看见的文章。因为不是每个人都知道如何编写LISP语句，所以Gnus提供了一些命令来帮你完成这件事情。

阅读文章的时候（在“Summary”编辑缓冲区里才能阅读新闻），即使没有使用过滤文件（我们马上就要对它进行详细的讨论），也有一条快速命令可以选用。如果关于某个话题的文章来得非常之多并且对它们却都不感兴趣，就需要在其中某篇文章显示在屏幕上时按下“C-k”（命令名是`gnus-summary-kill-same-subject`）组合键排除掉它们。如果只输入了一个字母“k”（命令名是`gnus-summary-kill-same-subject-and-select`），Emacs将排除掉与这个主题有关的全部文章，并选择下一个主题以供阅读。在“k”或“C-k”的前面加上“C-u”等于是进行相应的反操作，并且会把各种标记清除掉。Emacs会给用这种方法排除掉的文章旁边加上一个“K”标记，然后在用户阅读其他新闻的时候越过它们。

这些命令只对本次工作有效，而这往往也正是希望达到的效果。过滤文件是一种更具永久性的解决方案；只要不去修改它们，它们就会一直起作用。可以先用“C-k”命令在某次阅读新闻时试试，如果“垃圾”文章还是源源不断地涌现出来，就应该考虑在某个过滤文件里增加一条语句以彻底地解决它们。

可以创建一个局部过滤文件专门来对付某个新闻组、也可以创建一个全局过滤文件，在全部新闻组里把来自某个作者或者关于某个主题的帖子都排除干净。下面来看一个给局部过滤文件添加语句的例子。编辑一个局部过滤文件的方法是按下“ESC k”（命令名是`gnus-summary-edit-local-kill`）组合键；创建一个全局过滤文件的方法是按下“ESC K”（命令名是`gnus-summary-edit-global-kill`）组合键。

当想排除的文章出现在屏幕上时，按下“ESC k”组合键。Emacs将在屏幕上打开一个新的编辑缓冲区，它（显示在状态行上）的名字是那个新闻组的名字再加上一个“.KILL”后缀。（全局过滤文件的名字就简简单单地是“KILL”）。请看下图：

按下：ESC键

```
Buffers Files Tools Search Post Threads Article Score Misc Help
R      [ 18:Tom "Tom " Harrington ]
R      [ 24: Joe Btsfphbl ] Great Deals on Grommets!
[ 7: Sidney Schwartz ] NG charter archives?
[ 15: jules.verne@ai.nl ] Why the HD-led lights w/no W/R-action?
- Gnus news groups.questions/15927 [64 more] (Summary) -L44
```

Emacs 打开一个新的编辑缓冲区，它的名字是新闻组名加一个 “.KILL” 后缀。

如果想把这位作者的所有文章都排除掉, 请按下 “C-c C-k C-a” (命令名是 `gnus-kill-file-kill-by-author`) 组合键。如下所示:

按下： C-c C-k C-a

```
Buffers Files Tools Search Post Threads Article Score Misc Help
R [ 18: Tom "Tom" Harrington ]
R [ 24: Joe Btsfphbl ] Great Deals on Grommets!
R [ 7: Sidney Schwartz ] NG charter archives?
R [ 15: jules.verne@al.nl ] Why the HD-led lights w/no W/R-action?
--Gnus news.groups.questions.15927 (64 more) . (Summary)--L44-
(gnus-kill "From " "joe@grom \.com (Joe Btsfphbl)")
```

Emacs把用来在这个新闻组里排除这位作者全部文章的LISP代码，插入到编辑缓冲区里。

如果想按主题而不是按作者进行过滤,请按下“**C-c C-k C-s**”(命令名是**gnus-kill-file-kill-by-subject**)组合键。Emacs将把与这个主题有关的所有文章都排除掉,但也仅限于这个主题。如果真正想排除的是包含有某些特定单词(比如“*grommets*”)的全部文章,就需要使用正则表达式,而且还得有点LISP编程经验才行(详细讨论请参考第十二章)。

永久保存这些设置并退出“kill”编辑缓冲区的方法是按下“**C-e C-e**”（命令名是**gnus-kill-file-exit**）组合键。以后，每次进入Gnus时，Emacs都会自动加载这个过滤文件，并给那些被排除的文章加上一个大写字母“X”标记。如果没有用“**ESC u**”（命令名是**gnus-summary-clear-mark-forward**）组合键去掉“X”标记，Gnus就会自动越过这些文章。

另外，如果用其他新闻阅读器创建过滤文件，那么 Gnus 也会使用它们。表 6-11 列出了与过滤文件有关的操作命令。

表 6-11：与过滤文件有关的操作命令速查表

键盘操作	命令名称	动作
C-k <i>Group→Kill</i>	gnus-summary-kill-same-subject	排除这个主题下的所有文章；其反操作是“C-u C-k”组合键
k	gnus-summary-kill-same-subject-and-select	排除这个主题下的所有文章并选择下一个主题；其反操作是“C-u k”组合键
ESC k <i>Group→Edit kill file</i>	gnus-summary-edit-local-kill	编辑一个局部过滤文件（只对当前新闻组有影响）
ESC K <i>Misc→Edit global kill file</i>	gnus-summary-edit-global-kill	编辑一个全局过滤文件（对所有的新闻组都有影响）
C-c C-k C-a	gnus-kill-file-kill-by-author	插入排除此作者全部文章的 LISP 代码
C-c C-k C-s	gnus-kill-file-kill-by-subject	插入排除此主题全部文章的 LISP 代码
C-c C-c	gnus-kill-file-exit	保存过滤文件并退出“kill”编辑缓冲区

获取一份常见问题答疑（FAQ）文件

在读了一些新闻之后，有人可能想更积极地参与到其中，比如张贴一篇新文章，给文章的作者发一封邮件，或者把回应文章张贴到新闻组里等等。在向 Usenet 投稿之前，应该先阅读一下那个新闻组的常见问题答疑（frequently asked questions，简称 FAQ）文件。Gnus 为此准备了一个快速命令，只要连续按下“H f”（命令名是 **gnus-summary-fetch-faq**）两键即可。Gnus 会自动建立一条 FTP 协议（file transfer protocol，FTP）通路连接到存有该新闻组 FAQ 文件的某个服务器上。在 Emacs 里使用 FTP 功能的详细讨论请参考第七章。

在“Summary”编辑缓冲区按下“H f”两键，如下所示：

按下: H f

```

Buffers Files Tools Search Operate Mark Regexp Immediate Subdir Help
R [ 12: Teddy Hogenborn ] Re: "Couldn't open server" only on first t
< 20: Michael K.Sanders >
[ 49: Glenn Barry ] Re: xemacs using xv to display xface insts
< 19: Pete Forman >
-Gnus gnu.emacs.gnus/6641 (10 more) (Summary)-L-
/ftp@mirrors.aol.com:/pub/rftm/Usenet/gnu.emacs.gnus:
total 292
drwxr-xr-x 2 root sys 512 Jan 22 02:38 .
drwxr-xr-x 1498 root sys 44544 Jan 30 02:13 ..
-r--r--r-- 1 root sys 28617 Jan 20 00:36 Gnus_(Emacs_Newsrea
der).FAQ
--4%-Dired: gnu.emacs.gnus (Dired by name)--L5--All-
Reading directory /ftp@mirrors.aol.com:/pub/rftm/usetnet/gnu.emacs.gnus/
...done

```

Gnus 自动建立一条通路连接到存有该新闻组 FAQ 文件的某个服务器上。

如果那个FTP服务器因为有很多用户在登录而不能再接纳登录用户，那么就会显示一条说明这一情况的信息，请过几分钟再试试。按下“H f”键会进入一个 Dired 编辑缓冲区里，我们在第五章里已经解释过这一问题。它与平常使用的其他 Dired 编辑缓冲区没什么两样，但实际列在其中的却是某个远程服务器上的...些 FAQ 文件。(进入 Dired 编辑缓冲区里而非显示 FAQ 文件内容的原因是：很多 FAQ 都是由多个相对小一点的文件组成的，必须根据需要进行选择。)用 Dired 命令查看那些文件，移动到想要的 FAQ 文件所在的那一行，按“f”键。如下所示：

按下: f

```

Buffers Files Edit Search Help
R [ 12: Teddy Hogenborn ] Re: "Couldn't open server" only on first t
< 20: Michael K.Sanders >
[ 49: Glenn Barry ] Re: xemacs using xv to display xface insts
newsserver n.mit.edu! < 19: Pete Forman >
-Gnus gnu.emacs.gnus/6641 (10 more) (Summary)-L-
.....: senator-bedford.mit.edu!b.pixel.kodak.com
From: Steven L Baur <steve@miranova.com>
Newsgroups: gnu.emacs.gnus
Subject: Gnu (Emacs Newsreader)FAQ
Supersedes: <m291k8acaq.fsf@diana.miranova.com>
Followup-To: poster
Date: 19 Jan 1996 11:26:09 -0800
Organization: Miranova Systems, Inc.
Lines: 841
Kugtq m2ag3Approved: news-answers-request@MIT.EDU
Expires: 21 Feb 1995 00:00:00 GMT
Message-.fsf@miranova.com>
NNTP-Posting-Host: 204.212.162.100
Mime-Version: 1.0 (generated by tm-edit 7.40)
----- Emacs: Gnus_(Emacs_Newsreader)_FAQ (Text Fill)--L1-Top-----

```

Emacs 把该 FAQ 文件的内容显示在屏幕上。

接下来，可以在线阅读这份FAQ，也可以把它保存到一个本地文件上以后再看。本地保存这份文件的方法是先按下“**C-x C-w**”组合键，再给出一个本地系统上的文件名。(如果需要从头开始输入一个新的路径名，先用“**C-a**”命令移动到路径名的最开始，再用“**C-k**”命令删除它。) 输入一个本地文件名，就可以把那份FAQ复制下来。这只是查看和在本地保存远程文件的方法之一，用Dired的复制命令(“**C**”键)也可以同样容易地把它复制下来。有关操作请参考第五章中对Dired模式的介绍。既然知道获取FAQ文件是这么的容易，那还不赶快去下载并阅读它们。可以把它看做是一所Usenet大学——远程教育和学习的终极途径。

用 Gnus 向 Usenet 投稿

读过FAQ之后，有人可能还想往Usenet投稿。但考虑到Usenet的看客是那么的多，如果对是否有必要投稿没把握，那最好先忍一忍，或者请周围的人给文章把把关。Usenet的看客实在是太多了，多加小心总是应该的。

给文章加上签名

许多用户会把自己的文章加上一个签名，本章前半部分已经介绍过给电子邮件加上签名的办法。签名保存在一个名为“*.signature*”的文件里。对电子邮件而言，“*.signature*”文件在默认情况下是不会被包括在其中的。但对Gnus来说，它们却会被默认地包括在稿件里。因此，如果在练习使用电子邮件时创建了一个可笑的“*.signature*”文件，千万要记得删除它或者对它进行修改，否则它就会被默认地夹在稿件里发送到Usenet上——这可是我们的经验之谈。

在输入文章内容的时候，“*.signature*”文件是不会出现屏幕上的，但它实际上已经被追加在文章末尾——如果从新闻组里调出自己张贴的文章，就会看到情况确实是这样的。如果不想要“*.signature*”文件被自动地包括进去，请把下面这条语句添加到“*.emacs*”文件里：

```
(setq gnus-signature-file nil)
```

张贴一篇新文章

往 Usenet 张贴一篇新文章的方法是按下 “a” 键 (命令名是 **gnus-group-post-news**)。如下所示：

按下： a

```
Buffers Files Tools Search Misc Groups Group Help
 64: news.groups.questions
 12: gnu.emacs.gnus
 39: gnu.emacs.help
 6: alt.religion.emacs
-- Gnus List of groups  (nntp:news.ora.com)  (Group)--L3--All-----
Are you sure you want to post to all of Usenet?(y or n)?
```

Emacs 要求确认是否想向 Usenet 投稿。

回答 “y” 是指投出稿件，回答 “n” 是指取消它。如果回答的是 “y”， Emacs 将提示给文章输入一个主题。如下所示：

按下： y

```
Buffers Files Tools Search Misc Groups Group Help
 64: news.groups.questions
 12: gnu.emacs.gnus
 39: gnu.emacs.help
 6: alt.religion.emacs
-- Gnus List of groups  (nntp:news.ora.com)  (Group)--L3--All-----
Subject: [ ]
```

Emacs 提示给文章输入一个主题。

输入： Peace summit for vi and emacs users RETURN

```
Buffers Files Edit Search News Fields Help
Newsgroups: gnu.emacs.help
Subject: Peace summit for emacs and vi users
-text follows this line -
[ ]
--***-Emacs: *post-news*  (News Reply Fill)--L2--All-----
```

可以开始写文章了。

请注意： Emacs 已经把 “Newsgroups:(新闻组)” 栏默认地设置好了，这一栏的作用有点像电子邮件中的 “To:(收信人)” 栏。经过考虑，这篇文章还是投给 *alt.*

religion.emacs 新闻组比较好一些——它在那里可能会引起比较大的反响；因此，应该把新闻组那一栏改过来。

不要删除“*-text follows this line-*”这一行，它是不会出现在稿件里的。把文章内容写出来，或者，如果它已经在另外一个文件里，那么先按下“**C-x i**”组合键，再输入那个文件名，然后按回车键。

做好发送文章的准备工作之后，按下“**C-c C-c**”（命令名是 **news-news**）组合键。Emacs 将给文章加上签名（如果有“*.signature*”文件）并把它张贴到 Usenet 上。

回应他人的文章

对一篇文章的回应有两种：一种是把回应张贴到 Usenet，另一种是通过电子邮件直接与文章的作者进行交流。如果没有必要让全世界都知道，那么还是与文章的作者做一对一的交流比较好。直接回应文章作者的方法是按下“**r**”键（命令名是 **gnus-summary-reply**）。如果还想在自己的回应里引用那篇文章的原文，就需要按下“**R**”键。Emacs 将打开一个“mail”编辑缓冲区，写好回应后按“**C-c C-c**”就可以把它发送出去。

如果想把回应张贴到新闻组里，请按下“**f**”键（命令名是 **gnus-summary-followup**）。Emacs 会先问是否真的想把回应文章投给整个 Usenet，然后它会打开一个编辑缓冲区，那里面已经把必要的地址信息都填写好了。如果还想在自己的回应里引用那篇文章的原文，就需要按下“**F**”键（命令名是 **gnus-summary-followup-with-original**）；Emacs 将会把原文插入到回应文章里。可能需要对原文进行一些删节，好让它只包括准备回应的要点。写好回应之后，请按下“**C-c C-c**”组合键把它发送出去。

撤回投稿

如果在投出稿件之后又改变了主意，就得赶快动手。撤回投稿的方法是在“Summary”编辑缓冲区里按下“**C**”键（命令名是 **gnus-summary-cancel-article**）；Emacs 将尝试撤回这篇稿件。能否成功地撤回这篇文章在很大程度上要看动作有多快。即使是立刻就进行了操作，也可能会有人读到这篇文章了。

表 6-12 对向 Usenet 投稿用的命令进行了汇总。

表 6-12：投稿命令速查表

键盘操作	命令名称	动作
a <i>Post→Post an article</i>	gnus-summary-post-news	张贴一篇新文章
r	gnus-summary-reply	直接回复文章的作者
R <i>Post→Reply and yank</i>	gnus-summary-reply-with-original	直接回复文章的作者并附带一份原文的副本
f	gnus-summary-followup	把对当前文章的回应张贴到 Usenet 上
F <i>Post→Followup and yank</i>	gnus-summary-followup-with-original	把对当前文章的回应张贴到 Usenet 上并附带一份原文的副本
C-c C-f C-n	news-reply-newsgroups	移动到“News”编辑缓冲区的“Newsgroups:”栏；如果它不存在，就创建它
C-c C-f C-s	mail-subject	移动到“News”编辑缓冲区的“Subject:”栏；如果它不存在，就创建它
C-c C-f C-f	news-reply-followup-to	移动到“News”编辑缓冲区的“Followup-To:”栏；如果它不存在，就创建它
C-c C-f C-k	news-reply-keywords	移动到“News”编辑缓冲区的“Keywords:”栏；如果它不存在，就创建它
C-c C-f C-d	news-reply-distribution	移到“News”编辑缓冲区的“Distribution:”栏；如果它不存在，就创建它
C-c C-f C-a	news-reply-summary	移到“News”编辑缓冲区的“Summary:”栏；如果它不存在，就创建它
C-c C-y	news-reply-yank-original	在“News”编辑缓冲区里插入原始文章

表 6-12：投稿命令速查表（续）

键盘操作	命令名称	动作
C-c C-q	mail-fill-yanked-message	对插入到“News”编辑缓冲区里的原始文章进行段落重排（以便它们有统一的文本行长度）
C-c C-r	gnus-summary-caesar-message	在“News”编辑缓冲区里对文章进行简单的加密（采用rot13算法）
C-c C-e	news-inews	把文章张贴到Usenet上（如果在“News”编辑缓冲区里）或把它作为电子邮件发出去（如果在“mail”编辑缓冲区里）
C	gnus-summary-cancel-article	在“Summary”编辑缓冲区里，撤回投稿

退出 Gnus

退出 Gnus 请按下“q”键；根据当时所处的位置情况，可能得按两次。如果处在“Summary”编辑缓冲区里，按一次“q”键将退回到“Group”编辑缓冲区里。如果想完全离开 Gnus，你还得在“Group”编辑缓冲区里再按一次“q”键。Emacs 将提问：

```
Are you sure you want to quit reading news? (y or n)
```

回答“y”将退出 Gnus。Emacs 将保存“.newsr”文件，也就是把你这次的阅读情况记录下来。

电子邮件和新闻方面的基本操作到这里就介绍得差不多了。下一章将对诸如 Telnet、FTP 和 WWW 之类的 Emacs 因特网组件进行讨论。

疑难解答

- 对电子邮件进行排序的时候，画面内容没有变化。使用的可能是 Emacs 19 的一个早期版本。请按“**h**”键让 Emacs 刷新 RMAIL 的“Summary”画面。
- **Gnus不工作**。可能是系统没有连接到因特网，或者是没有本地新闻服务器。另一种可能是使用着 Emacs 一个比较早期的版本（比如 Emacs 18）——它们没有支持 Gnus 的功能。
- **Gnus 的工作情况与这里介绍的完全不一样**。使用的可能是 Gnus 4.x，它是 Emacs 19.29 或更早的版本里配备的新闻组件。这一章里介绍的是 Gnus 5.1（配备在 Emacs 19.30 或更高级的版本里）。



第七章

Emacs 的 因特网工具箱

本章内容：

- Emacs 的 Telnet 模式
- Emacs 的 Ange-ftp 模式
- 用 W3 模式浏览 Web 主页

有许多工具可以用来访问因特网。事实上，这些工具的数量每天都在增加。那么，我们凭什么说 Emacs 是 Telnet、FTP 和 World Wide Web 等因特网应用方面一种值得拥有的工具呢？虽然我们一直在说它并不是最好的工具，但 Emacs 提供的集成性却无与伦比。如果你正使用着 Netscape 之类的图形化浏览器，我们并不打算让你放弃它们。可一旦你学会了如何通过 Emacs 来访问因特网资源，就肯定会把它添加到你的百宝箱里去。

即使不离开 Emacs，也能 telnet 到另外一台计算机、浏览 Web 或者使用 FTP 功能下载并编辑文件。如果在网上发现了一些很不错的因特网资源，会不会把它的 URL (uniform resource locator，统一资源定位器) 地址发送给朋友？像这样的电子邮件你又收到过多少？可以把它们用笔和纸记下来，或者直接把它们存到某个文件里去。但更好的办法是去快速访问一下那个站点，如果它值得保留，就把它添加到自己的收藏夹里。Emacs 提供给我们的正是这些东西。

本章主要介绍几种因特网工具：

- Telnet 模式 —— 访问远程计算机

- Ange-ftp 模式 —— 用 FTP 协议 (file transfer protocol, 文件传输协议) 传输文件
- W3 模式 —— 访问 World Wide Web

Telnet 模式和 ange-ftp 模式是 Emacs 的内置编辑模式, 而 W3 模式则是一个需要用户自己从因特网上下载和安装的LISP软件包。有的站点上可能已经有人把它们下载下来了。如果还没有, 请继续阅读, 我们会在后面说明在因特网上的什么地方可以找到它。

本章将介绍一些用 Emacs 来访问因特网的基本操作。如果希望多学一些关于因特网自身的知识, 那么《The Whole Internet User's Guide & Catalog》会是一个很好的开端, 这本书作者是 Ed Krol, 由 O'Reilly & Associates 出版公司出版。

Emacs 的 Telnet 模式

Telnet 提供一种在网上使用其他计算机的方法。在 Emacs 里, 完全能用 Telnet 访问本地网络或者因特网上的任何一台主机并登录到该系统上——简单地输入“**ESC x telnet**”命令即可。我们现在就尝试用 Telnet 访问一个文档服务器试试。(文档服务器能够帮助在因特网上寻找程序和文件。找到之后, 再用 ange-ftp 模式把它们复制下来。具体操作将在后面介绍。)

输入: **ESC x telnet**

```
Buffers Files Tools Edit Search Minibuf Help
It was the best of times,it was the worst of times,it
was the age of wisdom,it was the age of foolishness,
it was the epoch of belief,it was the epoch
of incredulity,it was the season of Light,it was
the season of Darkness,it was the spring of hope,it
was the winter of despair,we had everything before us,
-----Emacs:dickens          (Text Fill)--L1--80%-----
Open telnet connection to host:
```

Emacs 询问想连接到哪台计算机上去。

输入: archie.unl.edu

```
Buffers Files Tools Edit Search Complete In/Out Signals Help
telnet>Trying 128.167.254.195 ...
Trying 129.93.1.14 ...
Connected to crcnis2.unl.edu.
Escape character is ].

SunOS UNIX (crcnis2)
login:[

--**-Emacs:*archie.unl.edu-telnet*      (Telnet:run)--LB--All-----
```

文档服务器提示输入一个登录名。

输入: archie RETURN

```
Buffers Files Tools Edit Search Complete In/Out Signals Help
Trying 129.93.1.14 ...
Connected to crcnis2.unl.edu.
Escape character is ].

SunOS UNIX (crcnis2)

login:archie
Password:[

--**-Emacs:*archie.unl.edu-telnet*      (Telnet:run)--L9----All-----
```

文档服务器继续提示输入一个口令。

这个文档服务器要求输入一个口令。许多服务器不要求输入口令，但这台需要（如果某个公共服务器要求输入一个口令——比如现在，那么请把电子邮件地址当做口令输入进去）。进入 Telnet 模式的这一步骤存在着这样一个问题（如果各位没有跳过本书前面对 shell 模式进行讨论的部分，就应该已经见过这个问题）：输入的口令会显示在屏幕上，而这是不安全的。如果想让 Emacs 把输入的口令隐藏起来——不管在 Emacs 里的什么地方——请把下面这两行语句添加到 “.emacs” 文件里：

```
(add-hook 'comint-output-filter-functions
          'comint-watch-for-password-prompt)
```

把这两行语句添加到 “.emacs” 文件里之后，保存该文件，退出 Emacs，然后再重新进入。现在，只要屏幕上再出现要求输入口令提示信息，Emacs 就会到辅助输入区里去输入口令。不管输入的是什么，Emacs 都将显示为一个星号。在这个例子里，我们将把 Joe Btsphbl 的电子邮件地址用做口令。如下所示：

输入: joe@grom.com RETURN

```

Buffers Files Tools Edit Search Complete In/Out Signals Help
# # # # # # ##### # # # # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# # ##### ##### # # # # # # # # # # # # # # # # # # # # #
Welcome to the ARCHIE server at the University of Nebraska -Lincoln
If you need further instructions, type help at the uni-archie>prompt.
#####
# Bunyip Information Systems, 1993
# Terminal type 'emacs' is unknown to this system.
# 'erase' character is '^?'.
# 'search' (type string) has the value 'sub'.
uni-archie>
--**-Emacs: *archie.uni.edu-telnet*      (Telnet:run)--L34--Bot-----

```

已经登录到这台文档服务器上。

虽然这台文档服务器不知道“emacs”代表的是哪一种终端类型，但却允许继续工作。（如果遇到因无法把 Emacs 识别为一种终端类型，而不允许以 Telnet 模式登录上机的服务器，就只能到 shell 下去使用 Telnet 了。）现在，我们来查找 GNU 压缩程序 gzip。如下所示：

输入: prog gzip

```

Buffers Files Tools Edit Search Complete In/Out Signals Help
Host ftp.cc.rochester.edu      (128.151.224.6)
Last updated 09:10  8 Sep 1994
|   Location: /pub/Solaris-2.1
|       FILE  -rwxr-xr-x  139264 bytes  01:00 25 Mar 1993  gzip
Host primost.cs.wisc.edu     (128.105.36.61)
Last updated 11:06  5 Aug 1994
|   Location: /bin
|       FILE  --x--x--x  90112 bytes  12:42 16 Mar 1994  gzip
uni-archie>
--**-Emacs: *archie.uni.edu-telnet*      (Telnet:run)--L87--Bot-----

```

稍等一会儿，文档服务器返回一份清单，里面列出可以找到 gzip 程序的各个地点。

既然清单已经显示在自己的屏幕上，那么就可以结束这次 Telnet 会话了。



输入: bye

```

Buffers Files Tools Edit Search Complete In/Out Signals Help
Host ftp.cc.rochester.edu      (128.151.224.6)
Last updated 09:10  8 Sep 1994
Location:/pub/Solaris-2.1
FILE -rwxr-xr-x 139264 bytes 01:00 25 Mar 1993 gzip
Host primost.cs.wisc.edu (128.105.36.61)
Last updated 11:06 5 Aug 1994
Location:/bin
FILE ----x--x--x 90112 bytes 12:42 16 Mar 1994 gzip
unl-archie>bye
*Bye.
Connection closed by foreign host.

-- Emacs: *archie.unl.edu-telnet*  (Telnet: no process) --L102-Bot--

```

Emacs 关闭这个 Telnet 连接，但这次会话的输出内容仍留在屏幕上。

文档服务器的输出内容会留在这个编辑缓冲区里，既可以把它保存到一个文件里去，也可以把它留在那里。学会使用 ange-ftp 模式之后，甚至可以马上把 FTP 地址复制到辅助输入区里并下载那个文件。

Telnet 模式就是这么简单；不过还必须再学几个键盘命令。如果想发送一个“C-c”字符到 Telnet 服务器去，就必须按下“C-c C-c”。如果想发送一个“C-z”字符，就必须也给它加上一个“C-c”前缀。如果各位没有跳过 shell 模式部分的学习内容，就应该还记得与此类似的 shell 命令也必须加上“C-c”前缀。表 7-1 列出了 Telnet 模式下的各种命令。

表 7-1：Telnet 命令速查表

键盘操作	命令名称	动作
(无)	telnet	进入 Telnet 模式
C-d	comint-delchar-or-maybe-eof	根据上下文、发送 EOF 字符或删除光标位置下的字符
RETURN	telnet-send-input	处理 Telnet 输入
C-c C-c	telnet-interrupt-subjob	中断当前作业，相当于 shell 中的“C-c”组合键
C-c C-q	send-process-next-char	发送紧随其后的控制字符，相当于 shell 中的“C-q”组合键
C-c C-d	comint-send-eof	发送 EOF（文件尾）字符

表 7-1: Telnet 命令速查表 (续)

键盘操作	命令名称	动作
C-c C-r	comint-show-output	让输出内容的第一行显示在窗口的顶部
ESC C-l	comint-show-output	让输出内容的第一行显示在窗口的顶部
C-c C-e	comint-show-maximum-output	让输出内容的最后一行显示在窗口的底部
C-c C-o	comint-kill-output	删除上一个命令的输出内容
C-c C-z	telnet-c-z	挂起或暂停一个作业，相当于 shell 中的“C-z”组合键
C-c C-w	backward-kill-word	删除前一个单词
C-c C-u	comint-kill-input	删除当前行，相当于 shell 中的“C-u”组合键
ESC n	comint-next-input	查看此后输入的命令（重复时可查看到更靠后的命令）
ESC P	comint-previous-input	查看此前输入的命令（重复时可查看到更靠前的命令）

Emacs 的 Ange-ftp 模式

FTP 是一种在因特网上或者在任何一个 TCP/IP 网络上挪动文件的标准办法。因特网上有无数“匿名的”FTP 服务器允许从它们那里复制文件。可以先利用刚才介绍的档案服务器找出某个特定文件的保存地点，然后再用 FTP 把文件复制到本地服务器上。

Ange-ftp 模式使得不用再学 FTP 命令了。就操作过程而言，先用“C-x C-f”组合键“找到”某远程系统上的文件，然后再用 Dired 命令（参见第五章）把它们复制下来即可（参见第五章）。Ange-ftp 模式是一个透明的 FTP 操作界面，并且是内置在 Emacs 里的。

用 Ange-ftp 模式复制文件

从用户的角度看，ange-ftp 模式实际上是个增加了一点语法、扩展了一些功能的 **find-file** 命令（键盘操作仍是按下“**C-x C-f**”组合键）。那么，当你按下“**C-x C-f**”组合键时，Emacs 又是怎样知道你想启动的是 Ange-ftp 模式呢？如果同时满足下面 3 个条件，Emacs 就会启动 ange-ftp 模式：

1. 文件名以一个斜线字符（/）打头
2. 斜线字符后紧跟着 *username@systemname*
3. 系统名之后是一个冒号（：），然后是子目录名或文件名（如果有）。例如 */anonymous@rtfm.mit.edu:/pub*

比如，输入“*/anonymous@rtfm.mit.edu:/pub*”将打开一个到主机 *rtfm.mit.edu* 的 FTP 连接，并把 */pub* 子目录里的文件清单显示在屏幕上。注意：容易忘记打头的斜线字符，而系统名和文件路径名之间的冒号就更容易漏掉。如果 Emacs 提示：

New file

或者

Use M-x make-dir RET RET to create a directory

就得用“**C-x C-v**”组合键（命令名是 **find-alternate-file**）来纠正这个错误。当然也有可能是把文件名、子目录名或系统名拼写错了，请注意检查它们。另外，当访问某个因特网资源的时候，如果它被更新过或者发生了其他变化，你可以去掉路径名中的一两个元素再多试几次。比如，如果在“*/anonymous@rtfm.mit.edu:/pub/usenet-by-group*”位置上没有找到自己想要的东西，可以去试试“*/anonymous@rtfm.mit.edu:/pub/*”；如果这样还是不行，可以再去试试“*/anonymous@rtfm.mit.edu:*”。

要想下载本章前面提到的 **gzip** 程序，请输入“**C-x C-f/anonymous@ftp.cc.rochester.edu:/pub/Solaris-2.1/gzip**”命令。如果给出了文件名，Emacs 就将把那个文件检索出来；如果省略了文件名，Emacs 就会把远程子目录里的文件清单显示在屏幕上，可以看看里面有没有自己感兴趣的其他东西。



输入: C-x C-f /anonymous@ftp.cc.rochester.edu:/pub /Solaris-2.1

```
Buffers Files Tools Search Operate Mark Regexp Immediate Subdir Help
/anonymous@ftp.cc.rochester.edu:/pub/Solaris-2.1:
total 1005
drwxrwsr-x 4 0          512 Mar 25 1993 .
drwxr-sr-x 20 0          512 Jun 14 20:54 ..
drwxr-sr-x 6 1142         512 Mar 25 1993 Migration_Kit
-rw-r r 1 1142           760 Mar 25 1993 README
drwxr-sr-x 2 1142         512 Mar 25 1993 Solaris_Info
-rw-r-xr-x 1 1142        139264 Mar 25 1993 gzip
-rw-r r 1 1142           866103 Mar 25 1993 migration.tar.z
--%--Dired:Solaris-2.1          (Dired by name)--L5--All-----
Reading directory /anonymous@ftp.cc.rochester.edu:/pub/Solaris-2.1/
...done
```

Emacs 把远程系统上的子目录显示在屏幕上。

把光标移动到 gzip 条目上, 然后按下: C

```
Buffers Files Tools Search Operate Mark Regexp Immediate Subdir Help
/anonymous@ftp.cc.rochester.edu:/pub/Solaris-2.1:
total 1005
drwxrwsr-x 4 0          512 Mar 25 1993 .
drwxr-sr-x 19 0          1024 Apr  5 11:25 ..
drwxr-sr-x 6 1142         512 Mar 25 1993 Migration_Kit
-rw-r r 1 1142           760 Mar 25 1993 README
drwxr-sr-x 2 1142         512 Mar 25 1993 Solaris_Info
-rw-r-xr-x 1 1142        139264 Mar 25 1993 gzip
-rw-r r 1 1142           866103 Mar 25 1993 migration.tar.z
--%--Dired:Solaris-2.1          (Dired by name)--L8--All-----
Copy [-p] gzip to: /anonymous@ftp.cc.rochester.edu:/pub/Solaris-2.1/
```

Emacs 询问想把文件复制到何处; 默认设置是复制到那个远程系统上。

Emacs 认为想把文件复制到远程系统上的某个地方。删除这个路径 (先用“C-a”组合键移动到路径的开始, 再用“C-k”组合键删除它), 然后输入想把文件复制过去的路径。就这个例子来说, 可以输入 “~/gzip”。

文件复制: 二进制格式与 ASCII 格式

FTP 用户常犯的错误之一是试图以 ASCII 格式来复制一个二进制文件。Ange-ftp 模式会尽量以正确的格式来下载文件——如果它发现必须以二进制格式来进行下载, 就会自动进行切换。它使用一个正则表达式来判断是否需要把文件作为一个二进制文件来进行传输 (程序、图形和 PostScript 文件等都属于二进制文件)。Ange-ftp 会

把以“.Z”、“.lzh”、“.arc”、“.zip”、“.zoo”、“.tar”、“.dvi”、“.ps”、“.elc”、“.gif”和“.EXE”等后缀结尾的文件看做是二进制文件。这些扩展名在变量 **ange-ftp-binary-file-name-regexp** 的默认值里已经设置好。如果想增加其他的文件扩展名，就需要自己去修改这个正则表达式。

想由自己来直接发出FTP命令也可以，但必须先进入一个 ange-ftp 通常不显示的编辑缓冲区才行。每个 FTP 连接都有它自己的一个名为“***ftp systemname***”的编辑缓冲区，按下“**C-x C-b**”组合键就能在编辑缓冲区清单里看到此缓冲区。如果在另外一个窗口里把这个编辑缓冲区显示出来，就能观察到 ange-ftp 模式的工作情况。如果在 shell 提示符下使用 FTP，就可以自己来发出FTP命令（比如切换到二进制模式的 **bin** 命令等）。

用 Ange-ftp 模式查找文本文件

FTP 最普通的用途就是下载文件。在访问FTP 站点的时候，我们经常会看到一些名为“*README*”或者“*Index*”的文件。如果想查看这些文件的内容，在命令行方式的FTP 中是很麻烦的。要先把这些文本文件下载回来，关闭FTP 连接，用 **more** 命令或者 Emacs 阅读这些文本文件的内容，然后再启动FTP 连接到刚才的地方，才能下载自己想要的文件。

但在 ange-ftp 模式里，这一切只需一个操作步骤就能完成——只要像对待任何其他文件那样用 ange-ftp 模式找到并打开 *README* 文件即可。Emacs 将把它显示在屏幕上。下面来看一个查找并打开某个文本文件的例子，准备查找的是 *gnu.emacs.help* 新闻组的 FAQ 文件（注 1）。

输入：**C-x C-f /anonymous@rtfm.mit.edu:/pub/usenet-by-hierarchy/gnu/emacs/help/GNU_Emacs_Frequently_Asked_Questions_(FAQ).part_1_5**（注 2）

注 1： 我们当然知道现在可以通过“Help”菜单来查阅这份 FAQ 文件的某个版本，我们只是用它来举个例子。再说，这个地址上的版本总是最新的。

注 2： 就这个例子来说，其实用不着输入这么长的文件名。可以先打开它的子目录，移到到这个文件，然后利用“F”命令打开它。

```
Buffers Files Tools Edit Search Help
  M-x: senator-bedfellow.mit.edu!the-tech!reuven
From: Reuven M. Lerner <reuven@the-tech.mit.edu>
Newsgroups: gnu.emacs.help.comp.emacs.comp.answers.news.answers
Subject: GNU Emacs Frequently Asked Questions (FAQ), part 1/5
Supersedes: <GNU-Emacs-FAQ-1_802998241@the-tech.mit.edu>
Followup-To: gnu.emacs.help
Date: 12 Jun 1995 23:02:50 GMT
Summary: Questions and answers having to do with GNU Emacs
        GNU Emacs FAQ: Introduction

Emacs:GNU_Emacs_Frequently_Asked_Questions_(FAQ)_part_1_5 (Text) Top
```

Emacs 检索出这个文件并把它显示在屏幕上。

如果想保留一份这个文件的副本，则可以用“**C-x C-w**”组合键加文件名的方法把它写到一个本地文件里。如果只是随便看看，并不想留一份副本，就没有必要把它存到本地系统上。给它设置一个书签（参见第四章中的讨论内容），以后再需要查阅它时，只需用一个简单的命令就能再次找到并打开它了，磁盘空间可以留给更重要的东西（比如从 Internet Talk Radio 网站下载的一个很大的音乐文件）。

用 W3 模式浏览 Web 主页

如果有 William Perry 编写的基于 Emacs 的浏览器 W3（注 3），那么即便是浏览 World Wide Web，也不需要离开 Emacs。（它不是 Emacs 软件的默认配备，但要想下载到它也非常容易，稍后再对此做详细介绍。）W3 的最新版本现在还没有图形化操作界面（但听说它正在朝这方面努力）。与 Emacs 的其他功能组件一样，我们看中的是 W3 与其他工作的集成性。假设收到的电子邮件里有一个指向某个 Web 站点，或 FTP 站点的 URL 地址，于是想去那里看一看（注 4）。移动到该 URL 地址处并输入“**ESC**

注 3： 在本书付印期间，有消息说 W3 将改名为 GnuScape。与其他优秀的 Web 浏览器一样，W3 也在不断地改进，所以 W3 现在的功能很有可能已经超越这本书里介绍的情况了。

注 4： URL 地址的格式是 “*protocol://resourcename*”。比如，Gnu Bulletin 的 URL 地址就是 “<http://www.cs.pdx.edu/~trent/gnu/gnu.html>”。W3 的 FTP 站点的 URL 地址是 “<ftp://cs.indiana.edu:/pub/elisp/w3>”。任何想让 W3 去识别的因特网地址都必须是这种格式，因此必须把 “<ftp.microsoft.com>” 写成 “<ftp://ftp.microsoft.com>”。而如果你遇到的是像 “URL: <http://www.openmarket.com>” 这样的 URL 地址时，则又必须把最前面的 “URL: /” 去掉。

x w3-follow-URL-at-point，就会转到那个站点上去。如果对那个站点上的东西感兴趣，则可以把该站点添加到收藏夹里，或者用“**C-x C-w**”组合键把该主页上的内容保存到一个文件里去。Emacs在保存这类文件时会先去掉主页上的HTML(Hypertext Markup Language, 超文本标记语言)语言标记。

也可以在不退出 Emacs 的情况下浏览自己编写的 HTML 文档，输入“**ESC x w3-open-local RETURN filename RETURN**”命令即可。Emacs 会用 W3 把这个 HTML 文件显示在屏幕上。(第九章将介绍如何利用 html-helper 模式编写 HTML 文档。)

有了 W3，基本上用不着再自己去输入 URL 地址了。可以把 URL 地址复制到删除环(kill ring, 见第二章)、粘贴它们插入到编辑缓冲区里，把它们添加到收藏夹里，或者去浏览它们，而这一切都不需要输入工作。此外，想下载 URL 地址指示的资源时，还可以在辅助输入区里按下 **TAB** 键，以利用 Emacs 的自动补足功能。如果想看看某个给定的站点上都有什么好东西，这不失为一个好方法。

W3 支持以下几种 URL 地址类型：

- Usenet 新闻
- HTTP .9 或 HTTP 1.0
- Gopher 或 Gopher+
- FTP
- 本地文件
- Telnet
- TN3270
- 电子邮件

虽然，在大多数情况下——至少是我们写这本书的时候，W3 还不能显示多媒体对象，但用户可以对 W3 进行设置，让它在选中一个多媒体超链接时，启动其他的浏览器进行浏览。比如，X 窗口系统的用户就能指定各种各样的外部程序，以便让 Emacs 显示图形。

让 W3 成为启动工作的一部分

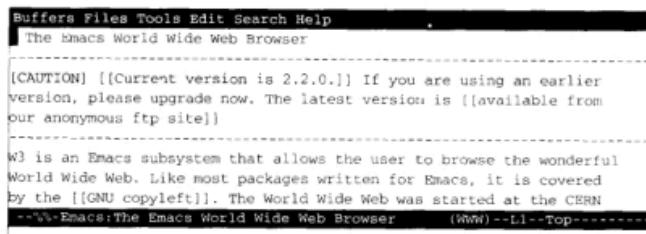
如果还没有安装 W3，那么可以从 “<ftp://cs.indiana.edu/pub/elisp/w3>” 处下载。如果想让它成为 Emacs 启动工作的一部分，请把下面这些语句添加到 “.emacs” 文件里，别忘了把第一行里的 “FULLPATHNAME FOR W3” 替换为真正的路径名。

```
(setq load-path (cons 'FULLPATHNAME FOR W3' load-path))
(autoload 'w3-preview-this-buffer 'w3 "WWW Previewer" t)
(autoload 'w3-follow-url-at-point 'w3 "Find document at pt" t)
(autoload 'w3 "w3" "WWW Browser" t)
(autoload 'w3-open-local 'w3 "Open local file for WWW browsing" t)
(autoload 'w3-fetch 'w3 "Open remote file for WWW browsing" t)
(autoload 'w3-use-hotlist 'w3 "Use shortcuts to view WWW docs" t)
(autoload 'gopher-dispatch-object 'gopher "Fetch gopher dir" t)
```

默认启始主页

输入 “**ESC x w3 RETURN**” 命令就能启动 W3。状态行上将出现 “WWW” 字样，而 W3 将会把默认的启始主页 “<http://www.cs.indiana.edu/elisp/w3/docs.html>”（注 5）加载到屏幕上。

输入： **ESC x w3 RETURN**



Emacs 进入 W3 模式并加载默认的启始主页。

按 **SPACE** 键向下翻页、按 **DEL** 键向上翻页。如果想移动到页面的开始，按 “<” 键，

注 5： 如果想另外指定默认启始主页，请把下面这条语句添加到 “.emacs” 文件里：

```
(setq w3-default-homepage "http:// your URL here")
```

比如，如果想用 Global Network Navigator 做为主页，就得用下面这条语句进行设置：

```
(setq w3-default-homepage "http://www.gnn.com/")
```

如果想移动到页面的结尾，按“>”键。如果想按超链接在页面中做跳跃式前进而不是一页一页地翻阅，请按“f”或TAB制表键（命令名是w3-forward-link），Emacs将前进到下一个超链接处，并把它的URL地址显示在辅助输入区里。再按TAB制表键则Emacs将前进到再下一个链接处。按下“b”键将使Emacs返回到上一个链接。在很多情况下，按照超链接移动要比逐页翻卷有用得多。

在WWW页面之间移动

可以用输入URL地址的方法来漫游WWW，但更典型的方法是选择超链接，这些超链接可能有各种各样的显示手段，比如呈高亮反显状态等。在下面给出的画面里，超链接显示在双方括号里。要想跟随超链接在WWW页面之间进行转移，需要把光标移动到超链接上再按回车键。如下所示：

按下：TAB TAB TAB

```
Buffers Files Tools Edit Search Help
The Emacs World Wide Web Browser

[CAUTION] [[Current version is 2.2.0.]] If you are using an earlier
version, please upgrade now. The latest version is [[available from
our anonymous ftp site]]

W3 is an Emacs subsystem that allows the user to browse the wonderful
World Wide Web. Like most packages written for Emacs, it is covered
by the [[GNU copyleft]]. The World Wide Web was started
--% Emacs:The Emacs World Wide Web Browser (WWW)--L1--Top-----
```

Emacs前进到“GNU copyleft”超链接处。

按下：RETURN

```
Buffers Files Tools Edit Search Help
*****
PREAMBLE
*****
the license agreements of most software companies try to keep users at
the mercy of those companies. By contrast, our General Public License is
intended to guarantee your freedom to share and change free software--
to make sure the software is free for all its users. The General Public
License applies to the Free Software Foundation's software and to any
other program whose authors commit to using it. You can use it for your
programs, too.
--% Emacs:GNU Copyleft Preamble (WWW)--L1--Top-----
```

Emacs进入“GNU Copyleft”Web主页。

那么，当转到另外一个主页之后，怎样才能再返回来呢？按“**I**”（小写的字母“L”代表“上一个”）键或者直接切换编辑缓冲区。W3会把每一个页面分别放在它们各自的编辑缓冲区里，用“**C-x b**”组合键就能在它们之间进行切换。按下“**C-x C-b**”组合键将调出编辑缓冲区清单，里面会把访问过的所有主页都列出来——其他的编辑缓冲区也包括在内。在某个主页的显示画面里按“**q**”键将删除那个主页，用来删除编辑缓冲区的“**C-x k**”组合键也能完成同样的事情。

除用来对 Emacs 编辑缓冲区进行操作的命令以外，还可以使用历史记录进行操作。历史记录不仅记录了这次上网所访问过的地方，此前的 W3 会话中访问过的站点也会被记在里面。在默认情况下，W3 不保存历史记录。如果想让它这样做，就需要把下面这条语句添加到“*.emacs*”文件里：

```
(setq url-keep-history t)
```

Emacs 会把访问过的所有 URL 地址都添加到一个 history（历史记录）编辑缓冲区里。如果想返回到前一页，请按下“**B**”键（命令名是 **w3-backward-in-history**）；如果想前进到后一页，请按下“**F**”键。要想查看本次会话的历史记录，请按下“**C-c C-b**”（命令名是 **w3-show-history-list**）组合键；Emacs 将以超链接的形式把这次会话的历史记录清单列出来，如果想访问其中的某个链接，把光标移到那儿再按下回车键即可。退出的时候，Emacs 将把这次会话的历史记录以追加方式写到一个名为“*.mosaic-global-history*”的文件里。（根据个人兴趣和磁盘空间的情况，有的用户可能不想让开关变量 **url-keep-history** 总是处于“开”的状态。可以定期删除这个文件或者从“*.emacs*”文件里删除这条语句。）

当返回到前一页的时候，会发现刚访问过的那个超链接的显示情况又发生了变化。括在访问过的超链接两端的将不再是方括号，而是花括号就像“`{(GNU copyleft)}`”这样。括号方面的这种变化能帮助回忆已经访问过哪些链接。

W3 还有一个很有用的功能，就是它不仅能记住已经访问过哪些主页，还能记住在这些主页上的位置，就像 Emacs 能够记住在各个编辑缓冲区里的编辑位置一样。因此，如果在阅读某个主页时去其他地方转了一圈，再利用历史记录功能回到这个主页上时还能接着刚才的地方继续往下看。



用 W3 模式查找 URL 地址

W3最让人喜欢的功能是它允许在Emacs里随时去访问任何一个URL地址。假设你在阅读自己用Emacs的ange-ftp模式取回来的Gnu Emacs FAQ时，看到了一个URL并且想到那里去看看，那么你根本不需要退出Emacs去启动一个浏览器。只要把光标移动到那个URL地址处，然后再输入“**ESC x w3-follow-URL-at-point RETURN**”命令就能让Emacs把该URL指示的资源取回来。（如果你经常使用W3，就应该考虑把这个命令绑定到键盘的某个按键上。具体步骤请参考第十一章内容。）

初始化状态：

```
Buffers Files Tools Edit Search Help
A collection of past GNU's Bulletins (as well as other GNU-related
information) is available at

  http://www.cs.pdx.edu/~trent/gnu/gnu.html

21:Where can I get help in installing Emacs?

See question 81 for some basic installation hints, and question 83 if
you have problems with the installation.

Emacs: gnu.emacs.Frequently_Asked_Questions_(FAQ).part_1.5 (Text)
```

在阅读 *gnu.emacs.help* 新闻组的FAQ时，看到了一个有意思的URL。

把光标移动到那个URL地址处并输入：**ESC x w3-follow-URL-at-point RETURN**

```
Buffers Files Tools Edit Search Help
*****
[IMAGE(gnu.xbm)] GNU'S NOT UNIX!
*****
```

This is an unofficial archive of GNU information. It is mainly a collection of information that I have collected over the years. Much of it is either historical information or information that FSF does not keep in the software archives.

- o [[GNU's Bulletins]]. Or you can skip the index and get the [[latest]] edition.
- o [[GNU HURD information]]
- o RMS paper on [[Why you should not use TCL]]

```
--% Emacs:GNU's Not Unix! (WWW)--L1--Top-----
```

Emacs将自动启动W3，并把该URL指示的资源取回来。

这种取回 URL 所指资源的功能可以随时使用。在阅读文件或者电子邮件的时候，我们经常会遇到一些有意思的因特网地址。有了 W3，只需把光标移过去，再输入“**ESC x w3-follow-URL-at-point RETURN**”命令就可以满足自己的好奇心了。如果你经常上网，就会发现几乎遍地都是 URL；如果先得记下它们，过后才能进行访问，就未免太麻烦了。有了 W3，随时都能到其他站点上去溜达一圈，等转够了再回到手里的工作上来。

我们在第六章介绍了 Emacs 的新闻阅读器 Gnus，在它里面取 URL 所指资源就更容易了。只要把光标移动到文章里的某个 URL 地址上，再按下 **RETURN** 回车键，Gnus 就会启动 W3 把那个 URL 指示的资源给取回来。

收藏夹的使用

收藏夹是想多次访问的 URL 地址，为了能够方便今后的使用，它们都被保存为超链接的形式。在默认情况下，Emacs 将使用 Mosaic 的收藏夹，它的文件名是“*~/.mosaic-hotlist-default*”。如果想导入 Netscape 的收藏夹，请输入“**ESC x w3-import-netscape-bookmarks**”（注 6）命令。Emacs 会复制那个收藏夹文件，把它转换为 W3 格式，然后把它保存为现有收藏夹的一部分。

请注意：收藏夹与 Emacs 书签是不一样的。Emacs 书签可以设置在普通文件或者远程文件里（通过 FTP），只需用一个简单的操作就能到达文件中书签的特定位置。收藏夹虽然在效果上与此类似，但它们只能工作在 W3 模式下，并且必须指向 URL 地址。

要想把当前浏览页面的 URL 添加到收藏夹里，请按下“**a**”键（命令名是 **w3-hotlist-add-document**），W3 将把该 URL 保存到收藏夹里，而它的名字就是当前主页的标题名称。如果不喜欢这个名字，可以用“**ESC x w3-rename-hotlist-entry**”命令给它另外取个名字。

在没有取回某个 URL 的情况下也能把它添加到收藏夹里去。只要移动到该 URL 地址上，再输入“**ESC x w3-hotlist-add-document-at-point**”命令即可。（如果已经在 W3 模式下，按“**A**”键也能完成同样的事情。）

注 6： 如果使用的是 W3 的一个早期版本，这个命令可能不管用。但可以把 HTML 文件 “*.netscape-bookmarks.html*” 当做一个本地的 Web 主页来打开。

要想访问收藏夹里的某个站点, 请按下 “**h**” 键 (命令名是 **w3-use-hotlist**)。Emacs 将提示给出想访问的收藏夹条目的名称。可以在输入该条目的第一个或者前几个字符之后按 **TAB** 键, Emacs 的自动补足功能将把该名称的剩余部分填写出来。

要想查看收藏夹的内容, 请按下 “**H**” 键 (命令名是 **w3-show-hotlist**)。Emacs 会把收藏夹中各个主页的标题列出来。当移动到某个标题并按下回车键时, Emacs 就会把那个主页取回来。

W3 与多媒体对象

根据系统的软、硬件配置情况, 有的用户也许能够用 W3 浏览多媒体数据对象。如果在某个图片链接上按下 “**ESC RETURN**” 组合键 (命令名是 **w3-follow-inlined-image**), Emacs 将会尝试启动系统上的某个看图器程序。Emacs 会检查系统里是否有一个把该图形数据对象, 与某个能对其进行正确处理的程序关联起来的定义。Emacs 先检查环境变量 **MIMETYPES** 是否存在, 如果存在, 就使用该变量所做的定义; 如果不存在, Emacs 将再去检查主目录里是否有一个名为 “**~/.mime-types**” 的文件。如果这个文件也不存在, Emacs 将依次检查以下几个系统文件是否存在, 它们是 **“/etc/mime-types”** 文件、**“/usr/etc/mime-types”** 文件、**“/usr/local/mime-types”** 文件以及 **“/usr/local/www/conf/mime-types”** 文件。

查看 URL

许多浏览器——例如 Mosaic 及其同系列产品, 会把当前主页的 URL 显示在屏幕顶部的某个地方。可 W3 却不一定会把 URL 显示在屏幕画面里。如果想查看自己正在浏览的主页的 URL, 请按下 “**v**” 键 (命令名是 **url-view-url**)。如果想查看某个给定超链接的 URL, 就需要移动到该超链接上再按下 “**V**” 键 (命令名是 **w3-view-this-url**)。Emacs 将把该 URL 地址显示在辅助输入区里。

复制 URL

自从出现了 WWW, 人们不再需要先学习一大堆关于因特网的知识, 就能去自己感兴趣的地方进行访问。但这也给 Web 带来这样一个缺陷: URL 地址必须输入正确。为了避免不必要的错误, 最好的办法是尽可能不亲自去输入它们。

我们已经介绍过一个可以避免亲手输入 URL 的方法：移动到 URL 地址上再输入“**ESC x w3-follow-URL-at-point**”命令。可如果想把 URL 地址发送给自己的朋友该怎么办呢？或者，如果遇到需要在自己设计的主页里插入一个指向某 URL 的交叉引用的情况时又该怎么办呢？方法有两种：如果想把当前主页的 URL 地址保存到删除环（kill ring，参见第二章）里，请按下“**k**”键或“**C-k**”组合键（命令名是 **w3-save-url**）；如果是想把光标位置上的 URL 地址复制到删除环里，请按下“**K**”键（命令名是 **w3-save-this-url**）。粘贴 URL 地址的键盘操作是按下“**C-y**”组合键。

按下“**ESC TAB**”组合键（命令名是 **w3-insert-this-url**）可以节省一个步骤。Emacs 要求指定一个用来放置该 URL 的编辑缓冲区，然后把该 URL 地址插到该编辑缓冲区的光标位置处。

查看页面的 HTML 源代码

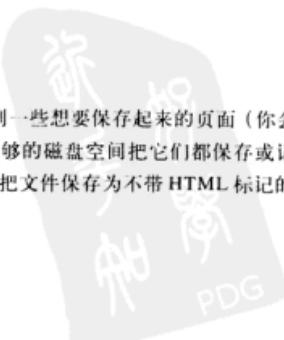
大部分 WWW 主页都是 HTML 文档。在编写 HTML 文档时，如果能够以其他各类主页的源代码做参考，借鉴他人安排 HTML 元素的经验，那么肯定有很大的帮助。如果想查看某个主页的 HTML 源代码，请按下“**ESC s**”或“**s**”键（命令名是 **w3-source-document**）。如果想查看某个 URL 所指主页的源代码，就得移动到那个 URL 地址上再按下“**S**”键（命令名是 **w3-source-document-at-point**）。Emacs 将把 HTML 源代码文件显示在屏幕上。

打开一个本地文件

如果想检查自己编写的 HTML 文件是否正确，可以在 Emacs 的 W3 模式里打开它。在 W3 模式里打开一个本地文件的方法是按下“**o**”键（命令名是 **w3-open-local**）；Emacs 将提示输入文件名。输入文件名再按下回车键，W3 就将对该 HTML 文件进行检查，它会把其中的错误报告出来，并且会把该页面显示在屏幕上。

W3 模式下的文件存盘操作

在浏览 World Wide Web 的时候，可能会遇到一些想要保存起来的页面（你会发现值得保存的页面实在是太多了，不见得有足够的磁盘空间把它们都保存或记录下来——不过这是另外一个问题了）。如果想把文件保存为不带 HTML 标记的形式



(以便在 W3 以外也能顺利地阅读它们), 请输入 “**C-x C-w filename RETURN**” 命令。Emacs 将把文件保存为普通的 ASCII 文本, 把其中的 HTML 标记 (以及各种超链接) 都去掉。

如果想在保存文件时保留 HTML 标记, 需要先按下 “s” 键查看它的源代码, 然后再输入 “**C-x C-w filename RETURN**” 命令保存它。Emacs 会认为要把文件写到主目录里, 但你可以根据自己的具体情况对路径进行修改。

表 7-2 对 W3 模式下的各种操作命令进行了汇总。

表 7-2: W3 命令速查表

键盘操作	命令名称	动作
(无)	w3-follow-URL-at-point	收回光标位置处的 URL 并进入 W3 (在 Emacs 的任何组件里都能使用这个命令进入 W3)
(无)	w3	进入 W3, 并把默认起始页面显示在屏幕上
C-o	w3-fetch	提示输入一个 URL, 然后把它取回来
SPACE	scroll-up	向后滚动屏显内容
DEL	scroll-down	向前滚动屏显内容
RETURN	w3-follow-link	选中一个超链接
<	w3-start-of-document	前进到当前页的最开始
>	w3-end-of-document	前进到当前页的最末尾
TAB 或 n	w3-forward-link	移动到下一个超链接
b	w3-back-link	移动到前一个超链接
m	w3-complete-link	提示输入一个要移动到的超链接
I	w3-goto-last-buffer	前进到访问过的最后页
B	w3-backward-in-history^a	退回到上一页
F	w3-forward-in-history	前进到历史记录里的下一页
C-c C-b	w3-show-history-list	把历史记录显示为超链接

表 7-2: W3 命令速查表 (续)

键盘操作	命令名称	动作
a	w3-hotlist-add-document	把当前页面添加到收藏夹里
A	w3-hotlist-add-document-at-point	把光标位置上的 URL 添加到收藏夹里
d	w3-hotlist-delete	提示指定一个收藏夹条目以删除它
H	w3-show-hotlist	把收藏夹显示为超文本
h	w3-use-hotlist	提示指定一个收藏夹条目以显示它
(无)	w3-rename-hotlist-entry	提示指定一个收藏夹条目以重新命名
ESC RETURN	w3-follow-inlined-image	尝试显示插入点处的链接图片
v	url-view-url	显示当前页的 URL 地址
V	w3-view-this-url	显示光标位置处超链接的 URL 地址
k 或 C-k	w3-save-url	把当前页的 URL 地址复制到删除环里
K	w3-save-this-url	把光标位置处的 URL 地址复制到删除环里
s	w3-source-document	显示这一页的 HTML 源代码
S	w3-source-document-at-point	显示光标位置处 URL 的 HTML 源代码
o	w3-open-local	在 W3 中打开一个本地 HTML 文件
U	w3-use-links	前进到下一个 <LINK> 标记处
?	w3-help	查阅 W3 的在线手册
Q 或 u	w3-leave-buffer	退出 W3, 但不删除当前编辑缓冲区
q	w3-quit	退出 W3, 并删除当前编辑缓冲区
g 或 r	w3-reload-document	删除编辑缓冲区, 然后再重新取回这一页
I	w3-goto-last-buffer	前进到刚才访问过的前一个编辑缓冲区

表 7-2: W3 命令速查表 (续)

键盘操作	命令名称	动作
P	w3-print-url-under-point	以 HTML 或 L ^A T _E X 格式把光标位置处 URL 指示的主页打印出来
p	w3-print-this-url	以 HTML 或 L ^A T _E X 格式把当前主页打印出来
w	w3-submit-bug	提交一份在 W3 里发现的程序漏洞报告
C-c C-v	w3-version	显示 W3 的版本号
ESC s	w3-search	查找 (如果这是一个可查找数据项)。用 Emacs 的查找命令进行本地查找
ESC m	w3-mail-current-document	把这个主页以电子邮件方式发送给某人，具体格式由用户自己选定
ESC M	w3-mail-document-under-point	取回光标位置处 URL 指示的主页并把它发送给某人
ESC TAB	w3-insert-this-url	提示指定一个用来放置此 URL 的编辑缓冲区；如果带有“C-u”前缀，就把 URL 插入到光标位置处
(无)	w3-import-netscape-bookmarks	导入 Netscape 的收藏夹文件并把它转换为 W3 格式 (原始文件不会被修改)

a. 只有在变量 `url-keep-history` 被设置为“t”时，与历史记录有关的命令才能用。要想对该变量做它永久性的设置，请把下面这条语句添加到 “.emacs” 文件里：

```
(setq url-keep-history t)
```

疑难解答

- **Telnet、Ange-ftp** 模式和 **W3** 不工作。造成这种问题的原因可能是计算机没有连接在一个 TCP/IP 网络上。如果情况确是如此，任何 TCP/IP 功能都将无法使用。第二种可能是 Emacs 在编译时没有把 TCP/IP 支持功能包括在内。请向系统管理员做进一步查询。顺便说一句，Telnet 和 Ange-ftp 模式是 Emacs 自带的功能组件，而 W3 则必须安装为一个 LISP 增值软件包。

第八章

简单的文字排版 和特效编辑

本章内容：

- 文本的缩进
- 文本的居中
- 插入分页符
- 矩形编辑
- 绘制简单的图形
- Emacs 的大纲模式

Emacs 在本质上是文件编辑器而不是字处理器。它是一种用来创建文件以容纳屏幕上显示内容的工具，而不是一种用来美化文本文件打印效果的工具。但 Emacs 仍具备以下几种排版方面的功能：

- 以各种方式对文本进行缩进。
- 使单词、文本行和文本段落居中。
- 绘制简单的框图、草图和图片，并且能够把它们插入文件或者电子邮件（其他字处理器和图形工具包大都不具备这项功能）。
- 按列而不是按行来进行编辑（特别适合用来创建和修改表格）。在 Emacs 里，这项功能被称为矩形编辑（rectangle editing）。
- 用大纲模式（outline mode）隐藏和显示文档的某些部分，以便能够对文档的整体布局进行安排。大纲模式能够简化按大标题、小标题、草稿和正式稿的步骤完成文件的起草和编辑的工作。

除了以上这些将在本章介绍的功能以外，如果显示器还具备显示不同字体和颜色的能力，那么 Emacs 也允许在编辑缓冲区里使用它们。关于这方面的详细讨论请参考

第十四章。Emacs 还为 troff、TeX 和 Scribe 等排版程序准备了专用的编辑模式，这些编辑模式将在第九章里介绍。

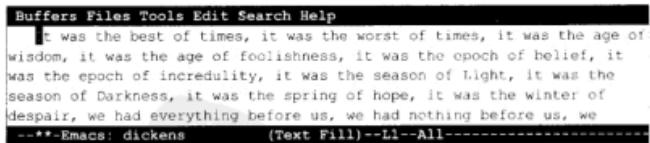
文本的缩进

在编写大纲、正式文件、技术文件或者其他任何东西的时候，经常需要对描写文本进行缩进处理。Emacs 为此准备了好几 种解决方法。我们将从制表位讲起——它是最简单的文本缩进方法。然后我们将介绍缩进前导字符串 (fill prefix)，它既可以用来缩进文本，也可以用来给每行文本都增加一个前导字符串（特别适用于非正式文档中的注解或者程序中的注释）。Emacs 里最精巧的文本缩进工具是文本缩进模式 (indented text mode)。如果文档里有各种各样的缩进层次，那么用它来写是最合适不过的了。

制表位的使用

在 Emacs 里，**TAB** 键的使用情况与各位预期的情况相当吻合，即把光标移动到下一个制表位处。光标后的文本也将向文件尾方向移动，文本就像是被推向了右方，结果是在这一行留出了一些空格——即使处在改写模式 (overwrite mode) (注 1) 也会如此。比如，如果想缩进某个段落的第一行，就可以先把段落的内容输入进去，然后移动到它的第一行按下 **TAB** 键。如下所示：

按下： **TAB**



TAB 键缩进的段落的第一行，其他文本也做相应的移动。

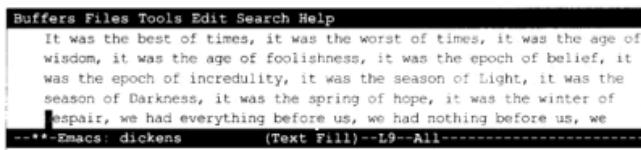
注 1：之所以要强调这一点，是因为某些编辑器中的制表位与它在打字机上的工作情况完全一样：按下 **TAB** 键将把光标向后移，但不改变文本的排列情况。

如果想手动缩进整个段落，当然没有问题：在每一行的开头按 **TAB** 键就能做到。不过 Emacs 给我们准备了一些简单点的法子，我们稍后再对它们进行介绍。现在，还是假设将手动完成缩进排版工作。

缩进整个段落

可以用在每行开头按 **TAB** 键的方法，来对某个段落做整体上的缩进。如下所示：

在段落中每一行的开始按下 **TAB** 键：



The screenshot shows a portion of the Dickens' A Christmas Carol text in Emacs. The menu bar at the top includes 'Buffers', 'Files', 'Tools', 'Edit', 'Search', and 'Help'. The main window displays the first few lines of the text, with tabs added at the start of each line. The status bar at the bottom indicates the buffer name 'dickens' and the mode '(Text Fill) -L9--All-'.

```
Buffers Files Tools Edit Search Help
It was the best of times, it was the worst of times, it was the age of
wisdom, it was the age of foolishness, it was the epoch of belief, it
was the epoch of incredulity, it was the season of Light, it was the
season of Darkness, it was the spring of hope, it was the winter of
despair, we had everything before us, we had nothing before us, we
--*** Emacs: dickens (Text Fill) -L9--All--
```

整个段落都被缩进。

但对段落进行这样的编辑之后，它会变得参差不齐，还能用 “**ESC q**”（命令名是 **fill-paragraph**）组合键对它进行段落重排吗？如果只对段落的第一行进行了缩进，这个命令就还能用——它会重新安排文本，在它们和右页边之间留出一个适当的间隔。它不会改变原来的缩进设置，也不会影响到此前的段落，第一行文本前面的缩进让 Emacs 明白这里开始了一个新的段落。

可如果对整个段落进行了缩进，“**ESC q**” 命令就不奏效了。由于它们的前面都有一些空格，所以 Emacs 会把每一行文本当做是彼此独立的段落。要想对整体缩进了的段落进行重排，必须先把这个段落定义为一个文本块，然后输入 “**ESC x fill-individual-paragraphs RETURN**” 命令。**fill-individual-paragraphs** 命令作用于一个文本块而不是一个单纯的段落，它会根据缩进情况重新安排文本块的段落（缩进程度相同的文本行将组成一个段落，缩进程度不同的文本行将被划分为不同的段落）；然后再在保留其缩进设置的前提下，对文本块中的各个段落进行重排。请看下面这个屏幕画面里的文本。

初始状态：

Buffers Files Tools Edit Search Help
My second correspondent, also a woman, sends me the following statement:
Life seemed difficult to me at one time. I was always breaking down, and had several attacks of what is called nervous prostration, with terrible insomnia...
Here is another case, more concrete, also that of a woman. I read you these cases without comment -they express so many varieties of the state of mind we are studying.
-----Emacs: hjames (Text Fill) --L9-- All-----

上图中的文本由3个段落组成，因为其中有3个缩进程度不同的区块。如果需要对这类文本进行段落重排，就需要先在它的一端（用“C-@”组合键）设置好文本块标记，然后移动到它的另外一端输入“**ESC x fill-individual-paragraphs RETURN**”命令。

设置制表位

在默认情况下，制表位之间的间隔是 8 个字符。但 Emacs 允许改变制表位的位置。要想改变制表位，请输入“**ESC x edit-tab-stops**”命令。屏幕上将出现一个“*Tab Stops*”编辑缓冲区。如下所示：

输入: ESC x edit-tab-stops

屏幕上显示一个制表位标尺，制表位的位置用冒号表示。

第一行上的冒号代表制表位的当前设置情况。随后两行构成了一把标尺，给出文本行上每一个字符的显示位置。要想插入一个制表位，需要用“**C-F**”组合键移动到预定的字符列位置，然后输入一个冒号字符（：）。要想删除一个制表位，需要先移动到预定的制表位位置，然后按下空格键。“***Tab Stops***”编辑缓冲区处于改写模式下，所以所做的修改不会影响到其他的制表位。记住：一定要在画面的第一行进行编辑修改，在其他行上做的修改不会生效。

对制表位的设置情况感到满意之后，按下“**C-e C-e**”组合键让它们生效。如果没进行任何修改，按“**C-e C-e**”组合键将退出这个编辑缓冲区。如果做了一些修改但又不想让它们生效，可以用“**C-x k RETURN**”命令删除这个编辑缓冲区；表示继续使用默认的制表位设置。

所设置的制表位只能在这次 Emacs 会话里起作用，新设置的制表位将影响到此后创建的一切编辑缓冲区。

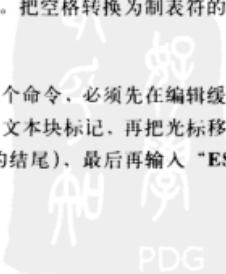
把制表位转换为空格

如果有人不明白“把制表位转换为空格”的原因，那么可以不去管它，直接跳过这一小节应该也没什么影响。这部分内容主要与那些需要把文件或软件包从一个系统挪到另外一个系统上的人们，或者与那些遇到打印问题的人们有关。

制表位提供了一种对文本进行某些简单排版的方法。但它们可能会引起大问题——因为它们不是“可移植的”。对于制表位之间的标准间隔设置没有统一的规定。制表位通常是用一个特殊的字符（ASCII 9 或者“**C-l**”）来表示的。当某个终端、编辑器或者打印机看到这个字符的时候，就会把这个字符解释为“移动到下一个制表位”——就像在打字机上看到的情况一样。刚刚介绍过 Emacs 允许把制表位设置在任意位置上，但要想重新设置打印机上的制表位可就没有这么简单了。虽然文件在屏幕上看起来完美无缺，但要是打印机的制表位设置得与 Emacs 的制表位不一致，就无异于一场灾难。对 troff 和 TeX 文档来说，制表位更是一个特别容易出问题的因素。有不少用户为了弄明白“为什么 Emacs 显示的是一个样，可打印出来的东西却又是另外一个样”而花费了大量的时间——根源全在几个隐藏着的制表位身上。

GNU Emacs 特意准备了一种既能去掉文件里的制表符，又能用制表位进行简便排版的方法。这样，可以在编辑工作中使用制表位，最后把各制表位统一转换为正确个数的空格，而文件的版面布局丝毫不变。与制表符不同，对空格的定义总是很明确的。去掉制表符的命令是“**ESC x untabify**”。把空格转换为制表符的相应命令是“**tabify**”，但我们建议大家不要记这个命令。

untabify 命令作用于文本块。也就是说，要想使用这个命令，必须先在编辑缓冲区里的某个地方（最好是编辑缓冲区的开头）设置一个文本块标记，再把光标移动到编辑缓冲区里的另外一个地方（最好是编辑缓冲区的结尾），最后再输入“**ESC x**



untabify RETURN”命令。“C-x h”组合键（命令名是 **mark-whole-buffer**）能够自动把光标放到编辑缓冲区的开头、把文本块标记放到其结尾。这将使去掉制表位的工作更容易进行一些，因为只需用一个简单的按键组合“**C-x h ESC x untabify RETURN**”就能把事情做好。

但我们必须提醒大家有一种文件是不能进行去除制表位操作的，它就是制作文件（**makefile**）。制作文件里的制表位都有着特殊的含义和作用，把它们转换为空格会引起一系列错误。（作为UNIX最有用的程序之一，出现这样一个设计缺陷真让人觉得遗憾。）假使不知道什么是制作文件也没有关系，只要记住不要对任何名字是“*makefile*”或“*Makefile*”的文件进行去除制表位的操作即可。

表 8-1 对与制表位有关的命令进行了汇总。

表 8-1：制表位命令速查表

键盘操作	命令名称	动作
(无)	edit-tab-stops	打开一个名为“*Tab Stops*”的编辑缓冲区以便改变制表位的设置情况
(无)	tabify	在不影响文本版面的情况下把 3 个及 3 个以上的空格视需要转换为制表位间隔
(无)	untabify	把全部制表位都转换为对应个数的空格
(无)	fill-individual-paragraphs	根据段落的缩进情况对它们进行段落重排

缩进前导字符串的使用

有好几种方法可以用来在段落前面自动增加一些空白。其中之一就是使用缩进前导字符串（*fill prefix*）——由 Emacs 自动放到每行输入文本开头的一个字符串。之所以会使用“*fill prefix*”这个术语，是因为 Emacs 把字换行（*word wrap*）功能称为“*auto-fill mode*（自动换行模式）”。换句话说，所谓缩进前导字符串就是这样一种字符串：当 Emacs 进行字环绕处理的时候，它会把这个字符串插入到每个文本行的开始（即给文本行加上一个“前缀”）。

缩进前导字符串最好是在自动换行模式下使用。如果状态行上有“*Fill*”字样，就

说明已经在自动换行模式中了。如果不是这样，请输入“**ESC x auto-fill-mode RETURN**”命令。

下面假设对一个备忘录进行缩进处理。在备忘录的第一行，手动输入缩进值——比如8个空格。然后按下“**C-x .**”组合键（命令名是**set-fill-prefix**）。Emacs将在辅助输入区里显示这样一条信息：“fill prefix “ ””。接下来，像平时一样开始输入。当输入文本超出右边界而使Emacs进行自动换行时，它会把8个空格的缩进设置插入到文本行的开头。

下面是一个更有意思的例子。没有理由让缩进前导字符串永远都是些空格，它们可以是任何东西。请看下面的例子：假设正在给同事发电子邮件，并且想给它加上一个抢眼的缩进前导字符串。

输入：**ELEPHANT RIDING PARTY!!! C-x .**

```
Buffers Files Tools Edit Search Help
ELEPHANT RIDING PARTY!!! |
```



```
----- Emacs: elephant (Text Fill) --L1--All-----
fill prefix:"ELEPHANT RIDING PARTY!!!"
```

先输入一个字符串，再用“**C-x .**”组合键把它设置为缩进前导字符串。

设置好缩进前导字符串之后，像平时那样输入邮件消息的内容。

输入：**The time . . . the zoo.**

```
Buffers Files Tools Edit Search Help
ELEPHANT RIDING PARTY!!!The time for the annual elephant riding party
ELEPHANT RIDING PARTY!!!!is nearly upon us. Plan to be at the San
ELEPHANT RIDING PARTY!!!!Diego Zoo at 3:30 on Saturday, June 10. A
ELEPHANT RIDING PARTY!!!!$5.00 donation per elephant ride is requested
ELEPHANT RIDING PARTY!!!!to benefit the zoo.|
```



```
----- Emacs: elephant (Text Fill) --L5--All-----
```

Emacs将把缩进前导字符串插入到邮件消息每一行的开头。

只需输入一遍“**ELEPHANT RIDING PARTY!!!**”就足够了，Emacs会自动把其余的插入进来。下面是一些在使用缩进前导字符串时需要了解的注意事项：



- Emacs 不会在段落的第一行插入缩进前导字符串。很明显，不可能让 Emacs 从第一段的第一行就开始插入缩进前导字符串（总得在什么地方亲自输入这个字符串吧）。但 Emacs 在任何一段的第一行都不会插入缩进前导字符串。这么说吧，如果上面这个备忘录有两个段落，那你就必须在第二段的开头再输入（或者复制）一遍短语“ELEPHANT RIDING PARTY!!!”。

不过，缩进前导字符串不用重新设置。Emacs 会给后续段落除第一行之外的所有文本行都加上缩进前导字符串——它只在段落的第一行上犯迷糊。

- 一旦开始使用缩进前导字符串，怎样才能让它停下来呢？没有专门为这准备的命令。得把光标放在左边界上，然后用“**C-x .**”组合键再定义一个新的缩进前导字符串。可这次的缩进前导字符串应该被定义为空字符串，即什么也没有。
- 最后，如果以后又想对这些段落重新进行排版，需先去掉以前的缩进前导字符串，重新定义一个新的，然后再按下“**ESC q**”组合键。这个命令虽然简单，却正是我们需要的，因为**fill-individual-paragraphs**之类的命令对付不了缩进前导字符串。

文本缩进模式

如果会大量使用缩进，就应该考虑使用 Emacs 的文本缩进模式 (*indented text mode*)。这个特殊的编辑模式额外提供了一些对缩进功能的支持。它很适合用来拟订大纲（也可以考虑使用本章后面将要介绍的大纲模式“outline mode”）。如果书写的东西里需要大段的引用，那么文本缩进模式可以帮助把引用部分和其他文字区分开。

进入文本缩进模式的方法是输入“**ESC x indented-text-mode RETURN**”命令。状态行上将出现“*Indented*”字样。另外，如果状态行上没有“*Fill*”字样，还得再输入“**ESC x auto-fill-mode RETURN**”命令。如果自动换行模式没有启用，那么文本缩进模式是无法工作的。

知道如何进入文本缩进模式之后，我们再来看看如何使用它。先从屏幕画面的左边界开始用**TAB**键或者空格键设置出足够的缩进，然后开始录入内容。到达文本行末尾的时候，Emacs 会自动对下一行进行缩进，使它与前一行匹配。下面是一个用文本缩进模式来排版一段引文的例子。

输入: **ESC x indented-text-mode RETURN**

Buffers Files Tools Edit Search Help

-----Emacs: dickensarticle (Indented Text Fill)--L1--All-----

状态行上将出现“Indented”字样。

输入文本的第一段。

Buffers Files Tools Edit Search Help

At the beginning of A Tale of Two Cities, Dickens penned the most
familiar opening lines found in any novel:

---**-Emacs: dickensarticle * (Indented Text Fill)--L2--All-----

接下来将以缩进格式输入一段引文。

输入: **RETURN TAB**, 然后输入一段引文:

Buffers Files Tools Edit Search Help

At the beginning of A Tale of Two Cities, Dickens penned the most
familiar opening lines found in any novel:

It was the best of times, it was the worst of times, it was
the age of wisdom, it was the age of foolishness, it was the
epoch of belief, it was the epoch of incredulity, it was the

---**-Emacs: dickensarticle (Indented Text Fill)--L5--All-----

只要把第一行缩进, Emacs 就能把整段引文都自动地缩进。

Emacs 会认为想对所有的段落都做同样的缩进处理。如果想开始一个新的段落, 需要连接两次回车键(留出一个空白行), 然后再按一次 **TAB** 键(用来设置下一个段落的缩进量)。在设置好最初的缩进层次之后, 再按一次 **TAB** 键将得到与前一段相同的缩进效果。在文本缩进模式里按一次 **TAB** 键会自动回到文本的第一列, 不管制表位是如何设置的, 也不管刚才设置的缩进距离是多大。

如果想减小缩进的距离, 可以用 **DEL** 键删除一些制表位或空格(把它们全删除也行); 如果想加大缩进的距离, 可以接着按 **TAB** 键或空格键。

知道了如何在文本缩进模式里录入文字之后, 我们再来看看怎样才能对它们进行编辑。随着编辑工作的进行, 文本将逐渐变得参差不齐, 需要时不时地把它们重排为



整齐的段落。具体操作方法已经在前面介绍过：把打算重排的那部分内容标记为一个文本块，然后输入“**ESC x fill-individual-paragraphs**”命令。

退出文本缩进模式的方法是进入另外一个主模式，比如文本模式等。进入文本模式的方法是输入“**ESC x text-mode**”命令，状态行上的“*Indented Text*”字样将改变为“Text”字样。

文本缩进模式中的制表位

文本缩进模式不使用 Emacs 预先定义的制表位。该模式下的制表位情况是这样的：首先，Emacs 会找到上一个非空白行，然后，第一个制表位将对应于该行第一个非空白字符的那一列，第二个制表位则对应于该行下一个单词的第一个字母所在的那一列，以此类推。该行结束之后，正常的制表位将重新“占据”这一行的剩余部分。图 8-1 给出了 Emacs 在文本缩进模式下的制表位定义情况。

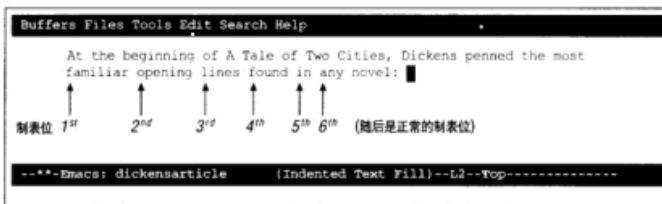


图 8-1：设定制表位

第一个制表位对应字母“f”的那一列，第二个对应字母“o”的那一列，以此类推。注意：这可能会带来很多的不便——如果在文本缩进模式下拟订一份大纲，那么，按两次（或者更多次）**TAB** 键并不一定会把你带到文本行上的同一个位置。好在制表位并不是惟一的选择，可以用空格键对缩进距离做任意调整。

对文本块进行缩进

如果已经在没有进行缩进的情况下录入了文本，现在又想对它们进行缩进，该怎么办呢？请按以下步骤操作：

1. 进入文本缩进模式（输入“**ESC x indented-text-mode RETURN**”命令）。
2. 移动到准备缩进的文本块的末尾，设置文本块标记。
3. 移动到准备缩进的文本块的开头设置好缩进距离，用空格或制表位来设置都行。
4. 按下“**ESC C-v**”组合键（命令名是 **indent-region**）。Emacs 将按第一行的格式对整个文本块进行缩进。
5. 按下“**ESC q**”组合键对文本块进行段落重排。

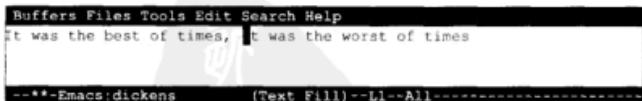
如果你打算缩进的区域有好几个缩进层次，就必须把不同缩进层次的每一块文本定义为不同的文本块，然后再按上述步骤进行设置。

其他缩进技巧

只要正使用着缩进（不管你是否在文本缩进模式里），那么随时可以用“**ESC m**”组合键（命令名是 **back-to-indentation**）移动到文本行的第一个非空白字符上。在没有被缩进的文本行上，这个命令将把光标移动到这一行的开头。也就是说，“**ESC m**”命令将把光标移动到文本行的“逻辑”开始位置，也就是平时按下“**C-a**”组合键时到达的位置。

另一个缩进命令是“**ESC C-o**”（命令名是 **split-line**）。可以用这个命令达到一种使文字呈台阶状排列的效果。这个命令也可以用在文本缩进模式以外的其他编辑模式里。先把光标移动到准备分拆到下一行去的文本处，然后按下“**ESC C-o**”组合键即可。如下所示：

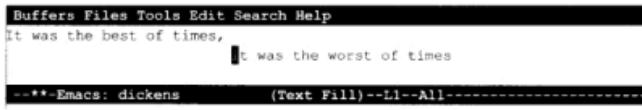
初始状态：



The screenshot shows a terminal window with the Emacs menu bar at the top. The menu items are: Buffers, Files, Tools, Edit, Search, Help. Below the menu, there is a single line of text: "It was the best of times, It was the worst of times". At the bottom of the window, there is a status bar with the text: "--*-Emacs:dickens-- (Text Fill) --L1--All-----".

我们想分拆这一行。

按下: **ESC C-o**



“**ESC C-o**”在光标位置把这一行分拆为呈台阶状排列的两行。

虽然这个命令在文本模式下也能制造出这种有趣的效果——可能适合书写诗歌之类的东西，但它在本章后面介绍的图形模式（picture mode）里会更有用。

表 8-2 对前面介绍的缩进命令进行了汇总。

表 8-2: 缩进命令速查表

键盘操作	命令名称	动作
C-x .	set-fill-prefix	把左边界到光标位置之间的字符串设置为加在段落每一行之前的缩进前导字符串；在第一列输入这个命令，表示此后将不再插入缩进前导字符串
(无)	indented-text-mode	进入文本缩进模式。在这个编辑模式里，按下 TAB 键将为某段落的后续文本行定义一个新的缩进层次
(无)	text-mode	退出文本缩进模式，让编辑缓冲区回到文本模式（进入其他的主编辑模式也能退出文本缩进模式）
ESC C-\	indent-region	按第一行的缩进设置把文本块作整体上的缩进
ESC m	back-to-indentation	把光标移动到文本行的第一个字符
ESC C-o	split-line	在光标位置处分拆文本行，并把它的后半截缩进到光标所在的一列
(无)	fill-individual-paragraphs	对缩进的段落进行重排，但保留缩进效果

文本的居中

另外一项常见的排版工作就是文本居中。比如，可能想让文档标题或者文档中的小标题居中。Emacs 提供有用来对文本行、段落和文本块等进行居中排版的命令。

首先要确认是在文本模式里。查看一下状态行，如果上面显示有“Text”字样，就说明已经在文本模式里；如果看不到这个单词，请输入“**ESC x text-mode RETURN**”命令以进入文本模式。接下来，录入想居中的文字（或者把光标移到某现有文本行的任意位置）之后，按下“**ESC s**”组合键。如下所示：

输入： **Annual Report**

```
Buffers Files Tools Edit Search Help
Annual Report

----- Emacs: annualreport (Text Fill) --L1--All-----
```

按下： **ESC s**

```
Buffers Files Tools Edit Search Help
Annual Report

----- Emacs: annualreport (Text Fill) --L1--All-----
```

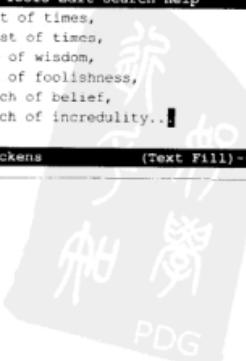
Emacs 把这一行放到居中的位置。

还可以对段落或者文本块进行居中排版。Emacs 将对段落或文本块中的每一行分别做居中处理，而不是把它们当做一个整体进行居中。要想使段落居中摆放，需要使用“**ESC S**”（命令名是 **center-paragraph**）组合键；要想使文本块居中，需要使用“**ESC x center-region**”命令。请看下面这个示例：让一段引文居中。

初始状态：

```
Buffers Files Tools Edit Search Help
It was the best of times,
it was the worst of times,
it was the age of wisdom,
it was the age of foolishness,
it was the epoch of belief,
it was the epoch of incredulity..| 

----- Emacs: dickens (Text Fill) --L6--All-----
```



按下: ESC S

The screenshot shows a window titled "Buffers" with a menu bar containing "Buffers", "Files", "Tools", "Edit", "Search", and "Help". The main buffer displays a poem by Dickens:

```

    It was the best of times,
    it was the worst of times,
    it was the age of wisdom,
    it was the age of foolishness,
    it was the epoch of belief,
    it was the epoch of incredulity ...
  
```

At the bottom of the window, the status bar shows "---- Emacs: dickens (Text Fill) --L6 All ----".

文本已居中。

在上面的例子里，逐行居中的效果看起来相当有艺术味。但有时会遇到需要让 Emacs 把文本块作为一个整体来对进行居中排版的情况。可以用本章前面介绍的 **indent-region** 命令来实现这种排版效果。为了达到视觉上的居中排版效果，一定要把文本块的缩进量安排好。

表 8-3 对文本居中命令进行了汇总。

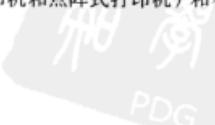
表 8-3: 居中命令速查表

键盘操作	命令名称	动作
ESC s	center-line	把光标所在的文本行居中
ESC S	center-paragraph	把光标所在的段落居中
(无)	center-region	把当前定义的文本块居中

插入分页符

虽然 Emacs 并没有提供什么页面排版功能，但用户可以在自己的文件里插入分页符。首先，输入引用命令（“C-q”组合键），它的作用是告诉 Emacs 不管随后的字符是什么，都把它插入到文本里（而不要把这个字符解释为一个编辑命令）。然后，在文本行的第一列输入“C-l”字符。这个字符在屏幕上会显示为“^L”。虽然这个字符看起来好像是两个字符，但其实却是一个，删除它试试就会明白了。

“C-l”代表着一个名为进纸符 (formfeed) 的特殊字符。许多打印机（特别是行式打印机、菊花轮打印机和点阵式打印机）和不少 UNIX 程序（比如 **more**）都会把进



纸符解释为分页符 (page break)。对 troff 和 TeX 等排版软件来说，“C-I”字符是无含义的 (或者更糟，可能引起问题)。对大部分激光打印机来说，“C-I”字符也是无含义的 (但很多激光打印机都有一种能够仿真行式打印机的工作模式，该模式也许会对进纸符做出正确的解释)。

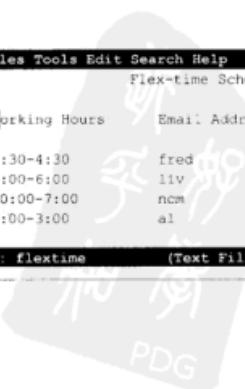
使用这种分页符的弊端是它们都是一些“硬”分页符。随着编辑工作的进行，如果某个页被加长或者缩短了，那么就必须重新设置它的分页符，这样才能让它在打印时实现正确地分页。我们认为比较好的方法是不在工作稿里添加这种分页符，等到想打印这个文件的时候，先复制它，然后再用一个简单的宏 (macro) 来每隔 “n” 行插入一个分页符 (“n” 代表每页的行数)。关于宏的编写方法，请参考第十章。

矩形编辑

当为移动或删除操作设定文本块的时候，文本块总是覆盖整个画面的宽度。Emacs 里有很多工作很适合用文本块来完成。可要是用文本块来编辑一个表格就不好办了吧？由于文本块会覆盖整个画面的宽度，所以不适合用来解决这类问题。Emacs 专门为这准备了另外一种用来定义文本区域的方法——矩形 (rectangle)。Emacs 中的矩形是用特殊的矩形编辑命令定义和处理的长方形区域。如果需要移动或者删除垂直排列的信息块，采用矩形编辑的办法将是非常有效的。比如需要移动表格的某列数据的情况，又比如在编写程序时需要添加或者删除代码注释的情况等。

先来看一个例子。假设正在对下图中的表格进行编辑，现准备把“Working Hours”栏移动到最右边去。文本块是解决不了这个事情的，可要是掌握了矩形编辑命令，那么这样的工作可以说是轻而易举。如下所示：

初始状态：



Flex-time Schedule				
Name	Working Hours	Email Address	Home Phone	
Fred	7:30-4:30	fred	(814) 722-3940	
Olivia	9:00-6:00	liv	(814) 827-3344	
Nicole	10:00-7:00	ncm	(412) 734-0492	
Alvin	6:00-3:00	al	(814) 437-9320	

--***- Emacs: flextime (Text Fill)--L5--All-----

定义矩形区域与定义文本块的方法是一样的，它们之间的区别体现在设定文本区域之后发出的编辑命令，这些命令将告诉Emacs打算对文本块进行操作，还是打算对矩形进行操作。

定义矩形的方法是：先把光标移动到它的左上角，然后按下“C-@”组合键或“C-SPACE”组合键设置文本块标记；然后再把光标移动到矩形的右下角。在把光标移动到矩形的右下角之后，还要再向右移动一个字符。为什么还要移动一个字符的距离呢？记住，在定义文本块的时候，文本块标记上的字符是文本块的一部分，但光标位置上的字符却不是文本块的一部分。定义矩形时也要遵守同样的规则，只是它得在水平方向和垂直方向上都满足这一要求。现在来定义一个覆盖表格第二列的矩形。如下所示：

按下：C-@

Buffers Files Tools Edit Search Help				
Flex-time Schedule				
Name	Working Hours	Email Address	Home Phone	
Fred	7:30~4:30	fred	(814)722-3940	
Olivia	9:00~6:00	liv	(814)827-3344	
Nicole	10:00~7:00	ncm	(412)734-0492	
Alvin	6:00~3:00	al	(814)437-9320	

--***-Emacs: flextime (Text Fill)--L5--All--
Mark set

文本块标记设置在准备移动的矩形区域的左上角。

把光标移动到这个矩形区域的右下角再往右一个空格的位置，即单词“al”中的字母“a”处。

Buffers Files Tools Edit Search Help				
Flex-time Schedule				
Name	Working Hours	Email Address	Home Phone	
Fred	7:30~4:30	fred	(814)722-3940	
Olivia	9:00~6:00	liv	(814)827-3344	
Nicole	10:00~7:00	ncm	(412)734-0492	
Alvin	6:00~3:00	al	(814)437-9320	

--***-Emacs: flextime (Text Fill)--L8--All--

光标位于矩形区域右下角再往右一个空格的位置。



这样，这个矩形区域就定义好了。下一步是删除它，然后把它移动到需要的位置。以粘贴为最终目的的矩形删除命令是“**C-x r k**”（命令名是**kill-rectangle**）。

按下： **C-x r k**



The screenshot shows a table of contacts in an Emacs buffer. The table has columns for Name, Email, Address, Home, and Phone. A rectangular selection is made around the entire table. The status bar at the bottom displays the message: "----Emacs: flextime (Text Fill)--L8--All---- Loading rect . . . done".

Name	Email	Address	Home	Phone
Fred	fred		(814) 722-3940	
Olivia	liv		(814) 827-3344	
Nicole	ncm		(412) 734-0492	
Alvin	al		(814) 437-9320	

矩形区域被删除，它被放到一个特殊的矩形删除缓冲区（rectangle kill buffer）里。

再重复一遍：在选取矩形的时候，必须先把光标放在它的左上角，设置上文本块标记；然后再移动到矩形右下角再往右一个字符的位置。Emacs会认为矩形就是一个规规矩矩的长方形。如有必要，它会在文本右侧添加空格，以使选取到的文本区域能够正确地围成一个矩形。

在屏幕上做任意的移动，然后用“**C-x r y**”组合键（命令名是**yank-rectangle**）把最后一次删除的矩形重新插入到光标位置。为了把“Working Hours”栏移动到表格的最右边，需要把光标移动到“Home Phone”的后面去。如下所示：

把光标移动到“Home Phone”的后面，然后按一次 **TAB** 键：



The screenshot shows the same contact table as before, but the "Home" column now contains tab characters instead of phone numbers. The status bar at the bottom displays the message: "----Emacs: flextime (Text Fill)--L5--All----".

Name	Email	Address	Home	Phone
Fred	fred		(814) 722-3940	
Olivia	liv		(814) 827-3344	
Nicole	ncm		(412) 734-0492	
Alvin	al		(814) 437-9320	

把光标移动到想重新插入矩形区域的地方。



按下：C-x r y

Buffers Files Tools Edit Search Help				
Flex-time Schedule				
Name	Email Address	Home Phone	Working Hours	
Fred	fred	(814) 722-3940	7:30-4:30	
Olivia	liv	(814) 827-3344	9:00-6:00	
Nicole	ncm	(412) 734-0492	10:00-7:00	
Alvin	al	(814) 437-9320	6:00-3:00	

--***Emacs: flextime (Text Fill)--L8--All-----

Mark set

Emacs 插入刚删除的矩形区域。

Emacs 按照操作精确地插入了矩形。把光标移到“Home Phone”的后面，然后再按一次 TAB 键的目的是为了在“Home Phone”和“Working Hours”两栏之间留出一些空间。要不然，Emacs 插入的“Working Hours”栏内容就会混杂在“Home Phone”栏的内容当中。需要提醒大家注意的是矩形没有相应的删除环（kill ring，参见第二章），但下一小节讨论的图形模式（picture mode）是个例外。只能把最近一次删除的矩形恢复回来。

矩形区域的剪切和粘贴操作需要多加练习才能掌握。一旦熟悉了这一流程，表格或者其他依赖于列的文字编辑工作就不在话下。

还有其他一些用来在屏幕上创建空白矩形的命令。举个例子，假设想把“Name”栏和“Email Address”栏之间的间隔加大两个空格。用“C-x r o”组合键（命令名是 open-rectangle）可完成这一工作。这个命令的作用是在屏幕上的任意位置插入一个空白的矩形，原有的文本将被推向右侧。首先定义一个矩形，然后再用“C-x r o”组合键在表格的第一栏和第二栏之间插入一些空白间隔。如下所示：

把光标移动到单词“Email”中的字母“E”上，然后按下“C-@”组合键

Buffers Files Tools Edit Search Help				
Flex-time Schedule				
Name	Email Address	Home Phone	Working Hours	
Fred	fred	(814) 722-3940	7:30-4:30	
Olivia	liv	(814) 827-3344	9:00-6:00	
Nicole	ncm	(412) 734-0492	10:00-7:00	
Alvin	al	(814) 437-9320	6:00-3:00	

--***Emacs: flextime (Text Fill)--L5--All-----

Mark set

Emacs 把文本块标记设置在矩形区域的左上角。

现在，需要定义准备插入到两栏之间的间隔量。移动到矩形的底部（即人名“Alvin”所在的那一行），然后再向右移动两个字符的距离。最后，按下“**C-x r o**”组合键给表格增加新的空格间隔。

把光标移动到单词“al”的后面，然后按下“**C-x r o**”组合键

Buffers Files Tools Edit Search Help					
Flex-time Schedule					
Name	Email	Address	Home	Phone	Working Hours
Fred	fred		(814) 722-3940		7:30-4:30
Olivia	liv		(814) 827-3344		9:00-6:00
Nicole	ncm		(412) 734-0492		10:00-7:00
Alvin	al		(814) 437-9320		6:00-3:00

--**- Emacs: flextime (Text Fill)--L8--All-----

Emacs 将插入一个两个空格宽的空白矩形。它会把表格中的其他内容推向右边。

clear-rectangle 命令的作用是清除矩形中的文本，在它原来的位置留下一个空白的矩形。这就像在黑板上擦掉一列板书一样。类似于黑板板书的情况，被清除掉的文本列将永久消失，不会被保存在矩形删除缓冲区（rectangle kill buffer）里。还是用刚才的例子进行说明。在传阅了表格“Flex-time Schedule”之后，有关人员都不同意把他们的家庭电话列在表里。于是：

把光标移动到单词“Home”中的字母“H”上，然后按下“**C-@**”组合键

Buffers Files Tools Edit Search Help					
Flex-time Schedule					
Name	Email	Address	Home	Phone	Working Hours
Fred	fred		(814) 722-3940		7:30-4:30
Olivia	liv		(814) 827-3344		9:00-6:00
Nicole	ncm		(412) 734-0492		10:00-7:00
Alvin	al		(814) 437-9320		6:00-3:00

--**- Emacs: flextime (Text Fill)--L5--All-----

Mark set

文本块标记被设置在准备清除的矩形的左上角。



把光标移动到最后一个电话号码之后的那个空格上，然后按下“C-x r c”组合键

Buffers Files Tools Edit Search Help		
Flex-time Schedule		
Name	Email Address	Working Hours
Fred	fred	7:30~4:30
Olivia	liv	9:00~6:00
Nicole	ncm	10:00~7:00
Alvin	al	6:00~3:00

--***-Emacs: flextime (Text Fill)--L8--All-----

clear-rectangle 命令删除了“Home Phone”栏，原先的位置上留下了一块空白。

正如大家所看到的，表格间隔还是不够完美；还需要用**delete-rectangle**命令把第二栏和第三栏之间多余的空白间隔去掉一些。类似于 Emacs 里的其他删除命令，**delete-rectangle**命令也不会把删除的东西保存到删除环（kill ring，参见第二章）里。要想删除那些多余的空白间隔并且不保存它们，得先把光标移动到“Email Address”后面的空格处、按“C-@”组合键设置上文本块标记；再沿着对角线方向移动到待删除矩形的另外一端；最后，按下“C-x r d”组合键。

把光标移动到“Email Address”的尾部，然后按下“C-@”组合键

Buffers Files Tools Edit Search Help		
Flex-time Schedule		
Name	Email Address	Working Hours
Fred	fred	7:30~4:30
Olivia	liv	9:00~6:00
Nicole	ncm	10:00~7:00
Alvin	al	6:00~3:00

--***-Emacs: flextime (Text Fill)--L5--All-----

Mark set

文本块标记被设置在准备删除的矩形的左上角。



把光标移动到最后一行里“6:00”再往前几个空格的某个位置上，然后按下“**C-x r d**”组合键

Name	Email Address	Working Hours
Fred	fred	7:30-4:30
olivia	liv	9:00-6:00
Nicole	ncm	10:00-7:00
Alvin	al	1:00-3:00

delete-rectangle 命令删除了那个空白间隔。

如果正在做一些精彩的表格编辑工作，那么能同时保存多个矩形，当然会有很大的帮助。这样，就可以把表格的每一栏都保存为一个矩形，再专门用一个矩形来保存各栏之间的空白间隔量。如果是在下面将要介绍的图形模式里使用矩形编辑命令，就可以不受限制地保存多个矩形。

表 8-4 对矩形编辑命令进行了汇总。

表 8-4：矩形编辑命令速查表

键盘操作	命令名称	动作
C-x r k	kill-rectangle	删除一个矩形并把它保存起来
C-x r d	delete-rectangle	删除一个矩形但不把它保存起来
C-x r y	yank-rectangle	在光标位置插入最后一次删除的矩形
C-x r c	clear-rectangle	清除矩形区域的内容，并且不保存它
C-x r o	open-rectangle	在矩形区域里插入一个空白矩形

绘制简单的图形

无论从哪个方面看，Emacs 都算不上是一个图形软件包，可它确实提供了一些有限的绘图功能。Emacs 有一个图形模式（picture mode），它允许用键盘字符绘制简单的图形。它很适合用来在电子邮件消息之类的文字性内容里快速插入一个草图——

这一功能连大多数图形软件包都不一定具备，这个编辑模式也很适合用来绘制框图、（电子工程师用的）时序图和时间表等各种简单的图形。

千万不要小看这个简单的组件！我们见过许多用 troff 或 TeX 精心排版的文件，里面总是有一两个用星号和短划线字符绘制出来的草图。当然可以用图形处理软件创建出更好看的图形来；但在现实世界里，pic 或者 TeX 系列的图形处理软件用起来既费时间又痛苦。用星号和短划线字符绘制出来的草图虽然不那么美观，但简便性却是谁也比不了的。

图形模式相当于把被编辑区域变成一块按行、列划分的绘图板。在图形模式里，可以用键盘字符创建出简单的图形（比如图 8-2 中的那个），并且不让它们被自动换行模式（auto-fill mode）的字环绕功能“重新排版”。

进入图形模式的命令是 “**ESC x edit-picture**”。状态行上将出现 “Picture” 字样，跟在它后面的是默认绘制方向（我们马上就要讲到它们）。按下 “**C-c C-c**” 组合键将退出图形模式，并返回到进入此编辑模式之前所在的一种主编辑模式里。

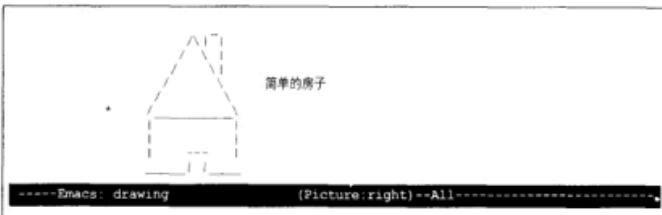


图 8-2：在图形模式中绘制图形

在图形模式中绘制图形

在图形模式里，可以沿 8 个方向中的任何一个用任何一种字符“画图”。不过，虽然可以沿 8 个方向绘图，但每次只有一个方向是可用的，这个方向就叫做“默认绘制方向（default direction）”。刚进入图形模式的时候，默认绘制方向是“右”。也就是说，如果连续按 4 次短划线键，就会向右画出一条“----”这样的直线。默认绘制方向会显示在状态行上，比如下面这样：

(Picture:right)

通过输入可以改变默认绘制方向的特殊命令，就可以沿另外7个方向画图。比如，“C-c ^”命令将把默认绘制方向设置为“右下”（译注1）；状态行上将显示“(Picture:se)”字样。如果沿这个方向连接4次短划线键，它们将排列为台阶的样子，如下所示：

图 8-3 给出了用来在图形模式里设置各种默认绘制方向的命令。

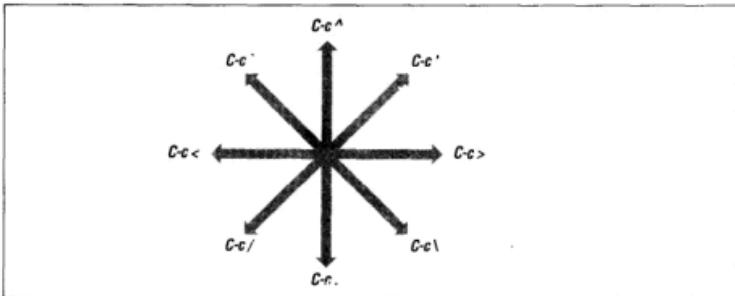


图 8-3：图形模式中的方向移动命令

要想记住在Emacs图形模式里设置默认绘制方向的命令并不困难，比如“C-c ^”命令中的“^”指向上方，而“C-c `”命令中的“`”指向左上方等等。如果你想出了一个记忆“C-c .”命令的好方法，别忘了告诉我们！也许你会把它记为“往下点儿”。

设置好默认绘制方向之后，重复按下任何键都会沿该方向画出一条字符线。可以在草稿编辑缓冲区（即“*scratch*”编辑缓冲区）里，用上图中的命令改变一下默认绘制方向试试。我们先一起画个长方形试试。

译者1：southeast，东南。外国人习惯用东、南、西、北及其组合来表示方向，而中国人习惯用上、下、左、右及其组合来表示方向；两者的对应关系是上北、下南、左西、右东。书中将采用中国人习惯的说法。

输入： **ESC x edit-picture**

把编辑缓冲区放到图形模式下，默认绘制方向是“右”。

输入： **TAB ESC 20 -**

Emacs 向右画出一条直线。接下来，把默认绘制方向改为“下”，再用“l”字符画出长方形的右边线。

输入： **C-c . ESC 5 |**

Emacs 向下画出一条直线。接下来，把默认绘制方向改为“左”，再画出长方形的底边线。

输入： C-c < ESC 20 -



Emacs 向左画出一条直线。接下来，把默认绘制方向改为“上”，再用“l”字符画出长方形的左边线并回到起点。

输入： C-c ^ ESC 5 |



Emacs 向上画出一条直线，完成了长方形的绘制工作。

图形模式中的编辑操作

说了这么多，大家应该对图形模式的功能和作用有一些基本认识了吧。它算得上是一种比较复杂的副编辑模式，而原因就在于它对许多主要编辑操作的键盘命令都重新进行了定义——而且还有很好的理由。针对普通 ASCII 文件的编辑技巧在图形上可就没那么好用了。在图形模式里，一般不会考虑使用文本的插入模式，这是因为标准的插入模式下所输入的字符会弄乱文本行的其余部分，因而会降低编辑图形的效率。因此，图形模式将明确地变为改写模式。其他一些功能也重新进行了定义，有些变化不那么明显，有些变化则相当显著。

因此，为了把图形模式讲清楚，我们就不得不对基础性编辑概念中的大部分重新进行审视。如果各位对图形不感兴趣，那么现在就可以跳过这一节内容；要不然，就得容忍我们继续唠叨下去。我们将从最基本的光标移动操作讲起。

图形模式中的光标移动操作

图形模式对基本的光标移动操作做了一些细微却又非常重要的改变。这些改变用一句话就能概括：图形模式里的“方向”概念说的是二维空间里的方向，而不是只相对于文件首尾的一维概念。比如，在其他编辑模式里，“C-f”组合键的作用大都是朝文件尾方向以每次一个字符的方式移动光标，连续按下“C-f”组合键等于是让光标做横越屏幕的移动；而到达文本行的末尾时，它将跳到下一行的第一个字符处，最终将到达文件的结尾。但在图形模式里，“C-f”组合键的含义是“向右移动”。当在图形模式里移动到行尾时，“C-f”组合键不会回绕到下一行去；它将继续向当前文本行添加字符。也就是说，一旦越过行尾，继续按“C-f”组合键就等同于按空格键。

“C-b”组合键也有类似的改变。在其他编辑模式里，“C-b”意味着向回移动到文件的最开始。但在图形模式里，它却意味着向回移动一位；而一旦到达行首，它就将停在那里不再继续向前移动。

“C-p”和“C-n”组合键则分别变成绝对的“向上移动”和“向下移动”命令。我们实际操作一回试试：先移动到某一行的末尾，再按下“C-p”组合键。在其他的编辑模式里，光标将上移一行，但仍然停留在行尾。如果上一个文本行比当前文本行短，光标在上移的同时也会相应地向左移动。但在图形模式里，“C-p”和“C-n”组合键引起的光标移动将是绝对的。它们将永远沿直线上移或者下移。

“C-f”、“C-b”、“C-p”和“C-n”这4个组合键能移动到想去的任何位置。键盘上的方向键通常也能工作；但还需要掌握沿默认绘制方向进行移动的光标移动命令——它们将使侧向移动更迅速。“C-c C-f”组合键的作用是沿默认绘制方向前进（这里所说的“前进”其方向可能是上、下、左、右及各种45°方向），而“C-c C-b”组合键的作用是沿默认绘制方向后退。（相对于默认绘制方向的向“上”及向“下”移动没有定义。）

举个例子。假设画了一个如图8-2所示的小房子，现在想沿左房顶向下移动光标。于是，先用“C-c /”组合键把默认绘制方向设置为“左下”。如果光标在左房顶最高处的“瓦片”上，按下“C-c C-f”组合键将沿左房顶向下移动一行，而按下“C-f”组合键将移动到右房顶最高处的“瓦片”上，如图8-4所示。

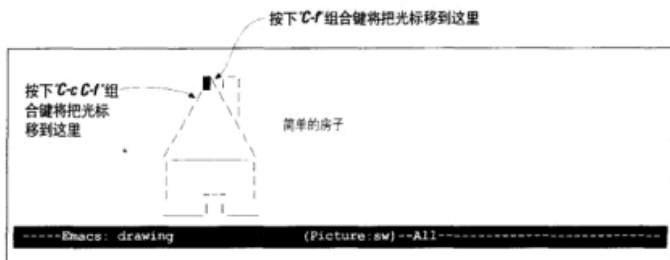


图 8-4：图形模式中的默认方向

插入文本行

如果在图形模式里多呆一会儿，还将发现更让人吃惊的事情：在图形模式里按回车键会移动到下一行的开始，但并不会插入一个空白行，这是因为 Emacs 会假定不想改变各行之间的相对关系。如果真的想插入一个新行，就需要使用“C-o”组合键；当前行的下面将出现一个新的空白行，但光标却不动。请看下图，光标的初始位置是在第一行中的“0”字符处。如果想在这两行之间插入一个新行，就需要按下“C-o”组合键。如下所示：

初始状态：

```
Buffers Files Tools Edit Search Help
abcdefghijklmnopqrstuvwxyz01234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz01234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ
--***Emacs: *scratch* -- (Picture:right)--L1--All--
```

初始状态文本；光标位于第一行的字符“0”上。

按下：C-o

```
Buffers Files Tools Edit Search Help
abcdefghijklmnopqrstuvwxyz01234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz01234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ
--***Emacs: *scratch* -- (Picture:right)--L1--All--
```

“C-o”命令插入一个新行，但没有移动光标。



在图形模式里，想用一个真正的“回车符”把某行从半中间断开还真有点难度。一种可行的方法是：先移动到文本行半中间的某个点上，然后用“C-k”组合键删除其后半截；再用“C-o”组合键插入一个空白行，按回车键移动到空白行的开头，最后再用“C-y”组合键把上一行刚被删除的后半截恢复在此空白行上。另外一种办法是先使用断行命令（“ESC C-o”组合键），再把新行前半截的空格都删除。

删除操作也得像这样迂回进行。“C-c C-d”组合键是图形模式下的字符删除命令，它的作用是删除光标位置下的字符，并把该行上的其余文本向左移。单独使用“C-d”组合键将删除光标位置下的字符，并把它替换为一个空格。而DEL键将删除光标左侧的字符，并把它替换为一个空格。

表 8-5 对图形模式命令和它们在文本模式下的“正常”行为进行了对比。

表 8-5：图形模式与文本模式的对比

键盘操作	文本模式	图形模式	图形模式下的等价命令
RETURN	插入一个空白行	把光标移动到下一行的开始	“C-o”组合键插入一个空白行
C-d	删除字符并把其余文本向左移	把字符替换为一个空格，不移动其余文本	“C-c C-d”组合键相当于文本模式下的“C-d”组合键
SPACE	把文本向右移，插入一个空格	把光标向右移，沿途经过的所有字符都将被替换为空格	无；请回到文本模式去插入空格 ^a
C-k	删除当前行上的文本；按两次“C-k”组合键将删除这一行	删除当前行上的文本，但不删除这一行	删除某行需回到文本模式进行操作或者使用 delete-rectangle 命令
TAB	插入制表符并把后续文本向右移	向右移动光标，但不影响沿途经过的文本	要想插入与一个制表位等价的空格，请回到文本模式进行操作
C-n	移动到下一行	向下移动，但光标的列坐标保持不变	(无)
C-p	移动到上一行	向上移动，但光标的列坐标保持不变	(无)

表 8-5：图形模式与文本模式的对比（续）

键盘操作	文本模式	图形模式	图形模式下的等价命令
C-f	向文件尾方向移动一个字符	向右移动一个字符	(无)
C-b	向文件头方向移动一个字符	向左移动一个字符；到达行首后就不再移动	(无)

- a. 如果想插入的是一个空白块，就得使用 **open-rectangle** 之类的命令。具体用法请参考本章前面对这一命令的介绍。另外，如果想在行尾插入空格，就得使用“C-f”之类的组合键。

有些工作还是临时切换回自己习惯使用的编辑模式里进行操作比较方便一些。“C-c C-c”组合键能回到进入图形模式前所在的编辑模式。在进行完必要的编辑之后，输入“**ESC x edit-picture**”命令可以重新进入图形模式。

如果想对绘制的东西进行平移，最好的方法是在图形模式里使用矩形编辑命令进行操作，这也正是下一小节将要讨论的内容。

在图形模式中使用矩形编辑命令

矩形编辑命令和图形模式是一对好搭档。用矩形编辑命令剪切和移动图形是非常方便的。本章前面介绍的矩形编辑命令都可以用在图形模式里，图形模式还有几个特有的矩形编辑命令。

在图形模式里能够保存的矩形可就不止一个了，可以同时把多个矩形区域分别保存到不同的“寄存区（register）”里。寄存区是 Emacs 开辟的存储空间，当前 Emacs 会话中创建的寄存区将一直呆在内存里，直到这次会话结束。这种能够保存多个矩形区域的能力对复杂的编辑工作（比如重新安排一个5栏表格中各栏的排列顺序时）特别有帮助。如果想清除一个矩形区域，并把它保存到一个寄存区里，请输入“**C-c C-w r**”命令。命令中的“r”是用来保存该矩形区域的寄存区的名字，它必须是一个单字符，比如“1”。“r”的值可以取任意字符、数字、字母、大写字母和小写字母都可。也就是说，键盘上有多少个独一无二的字符，就能保存多少个寄存区！（不过，最好不要用键盘上的功能键来充当寄存区的名字，因为功能键往往代

表着一个字符序列，而这会让 Emacs 不知所措。）这个命令会在清除的矩形区域处留下一块空白。如果想删除一个矩形区域而不留下一块空白，请在这个命令的前面加上一个“**C-u**”，即使用“**C-u C-c C-w r**”命令。

要想插入一个保存起来的矩形区域，需要使用“**C-c C-x r**”命令。命令中的“*r*”是用来保存该矩形区域的寄存区的单字符名字。

另外一个图形模式专用的矩形编辑命令是“**C-c C-k**”（命令名是 **picture-clear-rectangle**）。用“**C-c C-k**”清除一个矩形区域，等于把矩形区域中的文本都转换为空格，在它的位置上留下一块空白。“**C-c C-y**”可以把用“**C-c C-k**”清除为白的那个矩形区域原来的内容恢复回来。“**C-c C-k**”每次只能保存一个矩形区域的内容，大家可能还记得，“**C-c C-y**”也只能把最近一次删除的矩形区域的内容粘贴回来。

表 8-6 对图形模式里的编辑命令进行了汇总。

表 8-6：图形模式编辑命令速查表

键盘操作	命令名称	动作
(无)	edit-picture	进入图形模式
C-c C-c	picture-mode-exit	退出图形模式，返回此前的编辑模式
C-c ^	picture-movement-up	把默认绘制方向设置为上
C-c .	picture-movement-down	把默认绘制方向设置为下
C-c >	picture-movement-right	把默认绘制方向设置为右
C-c <	picture-movement-left	把默认绘制方向设置为左
C-c `	picture-movement-nw	把默认绘制方向设置为左上
C-c '	picture-movement-ne	把默认绘制方向设置为右上
C-c /	picture-movement-sw	把默认绘制方向设置为左下
C-c \	picture-movement-se	把默认绘制方向设置为右下
C-c C-f	picture-motion	沿默认绘制方向向前移动光标
C-c C-b	picture-motion-reverse	沿默认绘制方向向后移动光标

表 8-6：图形模式编辑命令速查表（续）

键盘操作	命令名称	动作
C-f	picture-forward-column	把光标向右移动一个字符
C-b	picture-backward-column	把光标向左移动一个字符
C-n	picture-move-down	把光标向下移动一个字符
C-p	picture-move-up	把光标向上移动一个字符
C-d	picture-clear-column	把光标下的字符转换为一个空格、不把其右侧的文本向左移
C-c C-d	delete-char	删除光标下的字符、把其右侧的文本向左移
C-k	picture-clear-line	删除当前行上的文本、但即使连接两次也不会删除这一行
C-o	picture-open-line	插入一个空白行
C-c C-w r	picture-clear-rectangle-to-register	清除矩形区域的内容并把它保存到寄存区 r 里
C-u C-c C-w r	picture-clear-rectangle-to-register	删除矩形区域并把它保存到寄存区 r 里
C-c C-x r	picture-yank-rectangle-from-register	把保存在寄存区 r 里的矩形区域内容粘贴到当前位置

Emacs 的大纲模式

在编写书籍、大段的备忘录或者技术报告等长篇大论的时候，要想随时掌握其通篇的布局结构往往是很困难的。如果不需要调整段落次序就能把大纲充实为文章，那当然很理想；但实际情况往往并不会这么顺利，甚至可能会相当麻烦。内容和标题经常会交织在一起，很难既看得见树木，又看得见森林。

Emacs 的大纲模式（outline mode）为这一问题提供了一种内置的解决方案。这个编辑模式允许有选择地展开和隐藏文章的段落内容——以它与文档结构的关系为依据。比如，可以把文档的内容都隐藏起来而只留下文章的各级标题，以便对文档的整体布局有一个了解。在查看文章各级标题的时候，可以把注意力集中在文章的

布局结构上，而不必关心遣词造句的问题；在解决布局结构方面的问题后，再让文章的内容重新显示出来。

如果只是写一份文字量很少的大纲，大纲模式的优势并不明显；它更适用于编写包含多级标题（或者大段程序）的文档。文档越长，要想迅速掌握它的整体布局也就越困难；而越是在这种情况下，大纲模式能够有选择地隐藏和显示文章内容的功能对我们就有更有帮助。

使用大纲模式的好处确实不少，但有一点需要大家特别注意：如果你的经验不够老到，大纲模式会从很多方面给你带来意想不到的问题，把文档弄得一团糟。最好的方法是用大纲模式对文档的一份副本进行编排（大家马上就会看到这很容易做到），而不是对原始文件进行编辑。这种使用副本进行工作的方法能够让你享受到大纲模式带来的好处，而原始文件却不会有任何风险。

大纲模式要求在大纲或者文档里遵守一些特殊的约定。在下面这个对照表里，我们把传统意义上的大纲列在左边，把符合大纲模式要求的同一份大纲列在了右边：

传统意义上的大纲	大纲模式
All About the Universe	All About the Universe
I. Preface	*Preface
A. Scope of book	**Scope of book
This book is all-inclusive	This book is all-inclusive
B. Intended audience	**Intended audience
Universe dwellers	Universe dwellers
II. Chapter 1	*Chapter 1
A. Universe basics	**Universe basics

传统大纲使用了一种层次化的布局结构，各级标题用罗马数字、大写字母、数字和小写字母等进行区分。而大纲模式里的各级标题在默认设置情况下是靠星号(*)来区分的，一颗星表示这是一个一级标题，两颗星表示这是一个二级标题，依次类推。不以星号打头的文本行（比如上表中的“*This book is all-inclusive*”）被称为“正文(body)”。Emacs要求星号必须出现在第一列里（可以使用传统大纲的缩进安排），但前提是星号必须从第一列开始出现。

上面给出的大纲示例里只有两个正文行。但既然是写书，正文行肯定会越来越多。就上表里的大纲示例而言，“This book is all-inclusive”这句话将被替换为一大段前言，而另一个正文行将被充实为这本书的第一章内容。如果使用得当，拟订大纲和书写正文这两种不同的工作，在大纲模式里并不会有明显的区别。随着大纲的逐步充实和细化，它也就逐渐演变为一篇文章（注 2）。

进入大纲模式

要想进入大纲模式，请输入“**ESC x outline-mode RETURN**”命令。状态行上将出现“Outline”字样。（大纲模式也可以作为一种副编辑模式使用，我们将在后面的内容里讨论这个问题。）

进入大纲模式后，可以通过几个特殊的命令，从大纲的某个部分快速移动到另外某个部分。比如，按“**C-c C-n**”组合键将移动到下一个标题或子标题；而“**C-c C-p**”组合键将移动到上一个标题。“**C-c C-f**”组合键将移动到同级的下一个标题——可以用这个命令从某个一级标题移动到大纲里的另外一个一级标题，或者从某个二级标题移动到另外一个二级标题（这两个二级标题必须在同一个一级标题下）。“**C-c C-b**”组合键则向回移动到同级的上一个标题。如果想从某个二级标题移动到它的一级标题，即在大纲的层次结构里上移一级，就需要按下“**C-c C-u**”组合键。（如果已经在某个一级标题上，那么“**C-c C-u**”将蜂鸣，因为此时已经没有上级标题可去了。）图 8-5 给出了这些光标移动命令在大纲示例上的移动效果。

这些命令使文章布局方面的许多问题都迎刃而解。如果经常遇到诸如“我知道自己正在写的是论据，可我记不起上一级的论点是什么”之类的事情，那么按下“**C-c C-u**”组合键就能返回到大纲的上一个层次。如果想了解该小节里的论据都是如何组织和安排的，可以用“**C-c C-b**”和“**C-c C-f**”组合键查看其前后的其他小标题。

注 2：当然，应该在文章完成之后把那些星号去掉。如果使用的是 troff 或 TeX 之类的排版软件，就可以用一个查询—替换操作把星号风格的标题修改为与宏定义包定义风格相对应的标题。如果只是想在文章写好之后去掉文件里的星号，用一个正则表达式（将在第十三章介绍）替换就能达到目的。另外，如果你胸有成竹，可以先对大纲模式进行定制，让它从一开始就按文档排版软件所能识别的标题风格进行工作，如果选择了这条路，就可以把拟订大纲和书写正文两种功能完美地集成在一起。

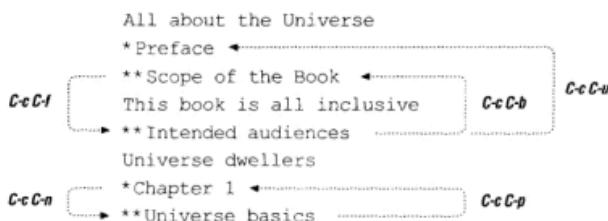


图 8-5：在大纲模式里移动

文本的隐藏和显示

大纲模式最重要的功能之一是它能够有选择地隐藏和显示文本的不同部分。用大纲模式查看一份长文档的布局安排是它最出彩的功能——把标题以外的所有东西都隐藏起来之后，就可以清楚地看到整个文档的布局结构，各小标题是否承上启下、它们是否需要重新安排等都一目了然。

hide-body 命令（“**C-c C-t**”组合键，隐藏正文）听起来有点像侦探小说里“隐藏证据”的味道，它的作用是把除各级标题（即以星号打头的文本行）以外的正文（或文本）行都隐藏起来，不让它们显示在屏幕上。在隐藏正文的时候，Emacs 会给相应的标题行加上一个省略号(...). 这个省略号的作用是表明这里隐藏着一些文本。编辑缓冲区本身并没有被修改——如果注意看状态行的左边，就会发现表明该编辑缓冲区已被修改过的星号并没有出现。如果在有文本被隐藏的时候对文件进行了存盘退出操作，Emacs 会把显示在屏幕画面上的东西和被隐藏了的文本都保存起来；而下次读入这个文件的时候，Emacs 又会把被隐藏的文本都显示在屏幕画面上。

使用 **hide-body** 命令可以迅速查看到一份长文档的布局结构。可以先给文档复制出一份副本，然后在副本里用一个查找 - 替换操作给所有的 **一级标题** 加上一颗星号 (*) 标记，给所有的 **二级标题** 加上两颗星号标记，依次类推（注 3）。最后，按下

注 3：请注意，首先进行最低一级标题的替换。如果有三级标题，要先给三级标题加上 *** 标记，然后是二级标题，最后是一级标题。要注意一级标题，文件中也许有不是标题的星号，如果使用正则表达式查找替换仅仅出现在行首的星号，就不会有什么问题了。