目录

1 学习 GNU Emacs **2**

1.1 前言 2

1.1.1 Debr Cameron, Bill Rosenblatt &Eric Raymond . . . 2

1.1.2 杨涛杨蓝云王建桥等译 2

1.1.3 学习本书的目的 3

1.1.4 GNU Emacs 和自由软件基金会 5

1.1.5 学习 Emacs 的方法 7

1.1.6 你可以参考下面列出的阅读顺序进行学习： 7

1.1.7 下面列出了一些大家可能打算在闲暇时学习的功能： . 7

1.1.8 最后，如果你坚持要从关到尾通读本书，那么请参阅下面对各章内容的简单介绍： 8

1.2 第一章 Emacs 的基本概念 11

1.2.1 E macs 简介 11

1.2.2 理解文件与编辑缓冲区 12

1.2.3 编辑模式 13

1.2.4 启动 Emacs 15

1.2.5 Emacs 的编辑画面 16

1.2.6 Emacs 命令 17

1.2.7 打开一个文件 20

1.2.8 如果读入了错误的文件 20

1.2.9 Emacs 的名称自动补足功能 21

1.2.10 插入和追加文件 22

1.2.11 Emacs 如何确定默认目录 22

1.2.12 保存文件 22

1.2.13 退出 Emacs 23

1.2.14 获取帮助 24

1.2.15 Help 菜单 24

1.2.16 小结 25

1.2.17 疑难解答 26

1.3 第二章文件编辑 28
1.3.1 光标的移动 30
1.3.2 移动光标的其它方法 30
1.3.3 把光标一次移动（或者多个）屏显画面 32
1.3.4 X 技巧：使用卷屏条 33
1.3.5 命令的重复执行 33
1.3.6 重新绘制屏显画面 34
1.3.7 Emacs 命令与你的键盘 35
1.3.8 文本的删除 36
1.3.9 恢复已删除的文本 37

1 学习 GNU Emacs

1.1 前言

1.1.1 Debr Cameron, Bill Rosenblatt &Eric Raymond

1.1.2 杨涛杨蓝云王建桥等译

G N U Emacs 是 Emacs 编辑器家庭中最受欢迎，传播范围最广，也是最强大和最灵活的 UNIX 文本编辑器，与其他文本编辑器的重要区别在于它是一个完备的工作环境。使用 Emacs 可以完成各种上学工作。本书循序渐进地讲述 Emacs 的入门知识，随着本书的深入，读者的 Emacs 使用水平将从初级（只会进行简单的文字编辑）提高到足以完成相当复杂的定制和程序设计任务的阶段。本书对 Emacs 19.30 中的新增功能做了全面的介绍，内容涉及字体和颜色，下拉菜单、卷屏条、增强的X窗口系统支持，以及对大多数标准按键进行正确的绑定。此外，书中还对 Emacs 自带的新闻阅读器程序 Gnus 和文件传输协议的透明接口 ange - ftp 模式进行了介绍。本书内容包括：

- 把 Emacs 当做一个因特网工具箱来使用
- Emacs 丰富而又全面的在线帮助功能

- 如何使用 Emacs 来编辑文件
- 把 Emacs 当做一种“Shell 环境”来使用
- 如何利用 Emacs 内建的排版功能
- 如何使用多个编辑缓冲区，多个 Emacs 窗口，多个X窗口
- 对 Emacs 进行定制
- Emacs 到X窗口系统的接口
- 用宏来完成重复性工作的理由和方法
- 作为程序设计环境的 Emacs
- Emacs LISP 程序设计入门
- 如何获得 Emacs

书中所附的速查卡列出了书中介绍的全部命令。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等窗口系统浒之前，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。

1.1.3 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 来写电子邮件演算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，你都会发现它其实是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 成为能够完成各种工作任务的工作环境，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。

阅读本书需要满足一些最基本的要求。要对UNIX操作系统都有一个基本的了解。具体来说，你必须知道文件和目录指的是什么，怎样给它们起名、对它们能进行哪些基本操作（如复制、删除和重命名等）

1.1.4 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，但它的起源确实是整个计算机史中不能不提的故事。负责维护和发行 GNU Emacs 的自由软件基本会（Free software Foundation, 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)、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 Richar 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 vcpbm 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，基本中最重要的是U nipress Emacs 和C C A 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。Stallman 的 Emacs 在自由软件基金会（F S F）和G N U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，它的确切含义是指把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，而且你几乎永远也得不到程序的源代码（除非你非常有钱）。如果某个商业化程序出了问题或者它不具备你需要的功能，就只能等卖给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，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都可以找到。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。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。GUN 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。F S F 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F S F 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R C S 等，也纳入 (FSF 的 copyleft 版权规定中。F S F 基金会还发行一 Linux 版本 (D ebian Linux)。有了 Linux 和G N U 软件工具，拥有一个能够完全体现F S F 价值观的完备的操作环境就成为可能。

1.1.5 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。本书的目的使大家尽快进入 Emacs 的大门。开篇的两章内容向大家介绍应该了解的基本概念，以后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主题上去。此外，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。

1.1.6 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⊠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公供大家参考。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的掌握，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情又不知该如何在 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。

1.1.7 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）

1.1.8 最后，如果你坚持要从关到尾通读本书，那么请参阅下面对各章内容的简单介绍：

1. 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。
2. 第二章，文件编辑：介绍 Emacs 的编辑操作，包括光标移动命令、广西的复制和 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 按照你的设定的方式去完成工作。
3. 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。
4. 第四章，使用编辑缓冲区和窗口：介绍多个编辑缓区、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。
5. 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
6. 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。
7. 第七章，Emacs 的因特网工具箱：介绍利用 Emnacs 编辑器远程访问其他计算机、用其 FTP 功能检索文件及浏览成维网的方法。

8. 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）。

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

10. 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。

11. 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。

12. 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。

13. 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。

14. 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。

15. 第十五章，Emacs 下的版本控制：介绍对文件版本进行控制的V C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。

16. 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。

17. 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。

18. 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。

19. 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。

20. 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。

21. 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

22. 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：为了更多的推出高质量的软件，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。

23. 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。

24. 词汇表：对 Emacs 术语的解释。

25. GNU Emacs 速查卡：最基本的 Emacs 编辑命令。

    S-right 按住 SHIFT 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。

    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 dduusr 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，让它更能折射出你的工作和生活方式。

1.2 第一章 Emacs 的基本概念

1.2.1 E macs 简介

GUN Emacs 是目前 UNIX 世界里最为常用的广西编辑器。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 并是不是市面上新新的编辑工具，它也肯定不是最好的。但它却将是你可以找到的最在用的编辑工具。我们期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。在编写它的时候，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。

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“它能把想做的许多事情都集成到一起来”这一点。

那么，所谓“集成”的意思又是什么呢？下面这个简单的例子就能说明问题。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，然后试试这格新命令：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，人可以编辑自己的.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 duhwq 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 两天之后，我们就再也用不着想什么"C - P 代表把光标上移一行"这类的东西了。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 来进行荼。接下来就该考虑怎样才能计 Emacs 的高级功能为自己服务的事情了。由于 Emacs 城有很多扩展菜单，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、一名话、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，这经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。

1.2.2 理解文件与编辑缓冲区

编辑器并不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：虽然缓冲区的内容看起来与文件非常想像，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文件了。

1.2.3 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，好偈还挺复杂，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、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（fundamentai 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 E

macs 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) 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 etex 模式 (latex mode) 按 latex 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C + + 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, E macs 会根据正在进行的编辑工作尝试进行正确的主模式。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 E X格式排版的文件, 它会转入T E X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。

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 (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) 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）临界时标记模式 (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）V C模式（vc mode) 在 Emacs 下使用各种版本控制系统

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城。

Emacs 还有其他一些没有列出的编辑模式，比如一些不常见的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。

1.2.4 启动 Emacs

启动 emacs 的办法很简单，输入“Emacs“再加上要编辑（注 1[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：输入一个文件名开始一次 Emacs 会话。如果给出的文件已经存在， E macs 将读入文件并把它显示在屏幕上。也可以省略文件名。如果只输入了“E macs”，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 * 空缓冲区里去，这是个试验各种练习的好地方。

[1]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（命令示找到）”错误信息，请向你的系统管理员求助。

1.2.5 Emacs 的编辑画面

进入 Emacs 的时候，你将在屏幕上 rhgcg 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 所求）。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。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（特殊是 vi）的一个不同之处。现在就来输入些东西试试，你将发现 Emacs 是很容易用不用的。（万一遇到了麻烦，请按 C-g 组合键）。在屏幕的底部（倒数第 2 行），Emacs 会给出一大堆当前工作情况的 wyth.p 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拓上次存盘之后又被修改过。如果没有做过任何个性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称（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：那一行（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“Bow”; 如果是文件中间，它会给出个百分数（比如“50%”表示现在看到的文件中部的内容）; 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。如果我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，它能真难地反映出在文件中的位置，基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 一个规范性的 X 窗口。

1.2.6 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求：

- 最常用的命令（比如那些光标移动☒主）都被绑定为“C-n”（n 可以是做生意字符）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 可以是做生意字符。“ESC n”的输入方法是：按下 ESC 键，释放它，再按下“n”键（注 2^2）

- 其他常用命令被绑定为“C-x something”（即 C-x 后面再加上一些东

[2]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 ☒而如果你有个 META 键，那么你就可以按住 META 键来输入寻一连串的命令。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。

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。在大家将要并不能的各种命令中，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。
- 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。

1. 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）菜单里都有哪些东西，请先把“I”开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，而鼠标光标的开头则变为一个箭头。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动 Emacs 画面的主窗口里，离开菜单，再松开鼠标按键。菜单时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（比如在写入文稿、不想把手移到鼠标上去的时候）。学会键盘操作命令还有一个好处，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。

2. 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。至少有两大理由把持增加这些菜单：

   - 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[3]）。屏幕画面显示如下：Emacs 打开一个列出菜单选项的编辑缓冲区。

3. 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。如果想要的是另一个选项上，按支上、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想要的选项为止。

[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。

1.2.7 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 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 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，但我们建议你不是先读完本章的“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。

1.2.8 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

读入的那一个”。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。

1.2.9 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，以构成一个惟一识别的文件名既可；名捕按下 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 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 按下 TAB 键时，Emacs 将自动补足文件名的剩余部分。按下：RETURN 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（足够惟一地确定文件即可）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。

1.2.10 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[4])，然后按下“C-x i”。与“C-x C-f”的情况相类似，Emacs 会提示用户在辅助区里输入文件名。

1.2.11 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。如果想查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* 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，但我其实不必为此费心--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。

1.2.12 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

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

filename”(文件已经存盘) 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 Emacs19 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-\”代替“C-s”, 用“C-^”代替“C-q”。不过这条命令只能解决本次会少中的这个问题。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 ”给键) 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 wrtie-file 命令把另外一个名字 (也许不要用另外一个路径) 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的一个文件 ; 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。

1.2.13 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“n”，Emacs 会再次提问是否年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。

1.2.14 获取帮助

GNU Emacs 的在线帮助功能丰富，具体细节我们将在第十章里做详细介绍。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 [5]）菜单里以一个选项。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。大家可能不会注意到：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。

1.2.15 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

[5]注 5：在某些键盘上，用来进入帮助功能的按键是 F1。

(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。

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

1.2.17 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”组合银奖地能出帮助系统。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三章中“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。)

- 无法进行在线帮助系统。“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，表参考第三章“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。有的工作站在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。要想恢复工作，请按下“C-g”组合键。如果问题还没有解决，请通过键盘输入“ESC x enable-flow-contro 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 Emacs19 (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 Emacs19，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关——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，这里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 poerful enough to run Emacs (本终端功能不足，无法运行 Emacs)”。终端弃置有问题，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。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 no(请输入 y 或 n)”，而你已经这样做了。问题的原因可能输入的是大写字母，而 Emacs 要求的是小写字母。请检查是否按过键盘上的“CAPS LOCK”键。

1.3 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
- 段落重排
- 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 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支欣赏 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主两页内容，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（注第二章 1:[6]）。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，它在需要违法行为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）

[6]在某些情况下（我们稍后会讲到），Emac 会在显示画面的右边界放上 ），这一行的其余部分就不显示出来了。

替用户把设置为 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 2^7)，把这个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。

1.3.1 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）或者按支方向键。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。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”将把光标移动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”不仅把光标移动下一行，还创建了一个新行。别问我们为什么会这样。

1.3.2 移动光标的其它方法

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身后移动过一个单词：“ESC f”把无票右移一个单词；“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。“ESC 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”把光标移动到

[7]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。

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 }”; 如果想把光标下移一个段落，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。

下面是记忆这些编辑基本命令的重要提示。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”把光标移动到文体行的的开始，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，经最后一个标点符号的后面必须有两空格。如果只有一个空格，Emacs 就无法分辨。类似地，把光标向前或者身后移过一个段落也涉及到 Eamcs 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。

如果文件里有分布符，可以通过敲入“C-x]”（命令名为 forward-page）或“C-x[”（命令名为 backword-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 imdeEmacs 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 命令会把光标移动缓冲区的末尾，而 backword-page 命令则会把光标移到缓冲区的开头。

文本模式里的分页残奥会是一个进纸换页符，它的作用是告诉打印机先前进到下一个打印页为（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”进给纸这个术语的来历）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”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字符 (^L)，但实际上是一个。(大家用 DEL 键席卷删除一个这样的字符就明白了。)

1.3.3 把光标一次移动 (或者多个) 屏显画面

如果想一次把屏幕的的文件上卷一页，可以使用键盘上的 PgDn 键或者按下“C-v(命令名是 csroll-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”提供了快速翻卷文件的简便方法。

如果输入的光标移动命令的目标位置在当前屏显文体内容以外，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 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，需要拉卡拉到达某个文件的开头或者结尾。在 Eamcs 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“Syntan 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，或其它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。

1.3.4 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，它的开头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；而把滑块一起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。

1.3.5 命令的重复执行

现在来学习几个提高操作效率的技巧。Eamcs 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”哪样也在“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 emacs 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[8])。

[8]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”就几乎问题能直到命令重复

1.3.6 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，可能伟适用电子邮件到达、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 是不会被来来往往的字符弄糊涂的，不过它们却可能氢用户给弄糊涂，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的位置上等等。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

因子的作用。

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-word	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 [	forward-page	光标后移一页
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)	goto-line	光标前进到文件的第 n 行
(无)	goto-char	光标前进到文件的第 n 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 n 次后续命令
C-u n	universal-argument	重新执行 n 次后续命令 (省略 n 时重复 4 次)

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。

1.3.7 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我稍有差别。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。

1.3.8 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；再次按下这个组合争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 yyif.mmbwyysgjb 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 争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。

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）。两次提醒大家注意 ESC 是如何扩大命令的作用范围的 :“C-d”对字符操作，而“ESC d”对单词进行操作。

Emacs 提供了向前、身后删除单词、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，具体是向前删还是身后删要看命令本身是向前操作不是身后操作。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 worst of times	It was the wolf time
It was the worst of times	It was the wof times
It was the worst of times	It was the worst 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起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不对劲光标位置之间的东西，可以试试复杂点的组合命令“ESC -C-k”（即先按 ESC 键，后面跟一个短划线，然后是“C-k”）。

1.3.9 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 killp-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。删除弄不清民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[9]），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 会把它们收集起来作为一个整体放到删除环里去——这里用一个“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。

[9]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。

光标位于左上角，按下 C-k C-k C-k C-k 用“C-k”删除了前两行。按下：C-y 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”命令删除的所有的东西和用“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”删除文本也将进入删除的单个字符了。（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的文本的工作。比如，假设先按了一次“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 edit–>Paster Most Recent	yank	恢复被删除的文本
C-w 或 SHIFT -DELETE edit->Cut	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