

Emacs 编辑环境教程

开始之前

了解本教程中包含的内容以及如何最好地利用本教程。

关于本系列

Emacs 编辑环境深受 **UNIX®** 开发人员的喜爱。它是世界公认的编辑器之王，但许多用户发现它学起来需要一定的过程。**Emacs** 环境初看起来并不是很直观，而且和其他编辑器和字处理器的工作方式不大一样。但 **Emacs** 的学习并不困难。一旦您熟悉了它，就会发现它有多么直观，而且越用越顺手。这个系列教程将为您提供指导，带您了解 **Emacs** 的基本知识，如它的功能、原理、按键命令布局和编辑文本的方法，然后深入它众多的强大编辑功能。

在完成本系列文章的学习后，您可以通过 **Emacs** 很方便地进行日常编辑工作，能越来越熟练地使用 **Emacs**，并对 **Emacs** 的许多高级功能有一个良好的认识。

关于本教程

本教程的第一部分将重点介绍 **Emacs** 的历史和起源，并在稍后的部分中阐明如何：

启动和停止 **Emacs**

操作并读文件，以便编辑它们

使用基本的编辑键

在文档间移动

使用强大的 **Emacs** 文件标识例程

用鼠标进行编辑和选择

目标

本教程的主要目的是带您初步了解 **Emacs** 编辑器，并简要地介绍该应用程序及其设计原则，告诉您如何在这个编辑环境中高效地工作。

在完成本教程后，您将了解在用 **Emacs** 执行基本文本编辑任务时应该知道的所有知识。

先决条件

如果要从本教程中受益，您不一定要了解先前所介绍的那些关于 **Emacs** 的知识；不过，您应当对文本编辑器和字处理器能做的工作有一个初步的认识。虽然本教程是针对所有程度的 **UNIX** 专业知识而编写的，但是如果您了解 **UNIX** 文件系统，会很有帮助，这包括： 文件

目录

权限

文件系统层次结构

系统要求

本教程需要您在基于 **UNIX** 的系统中有一个用户帐户，且此系统中安装有最新版本的 **Emacs**。

Emacs 有几个版本；最原始，也是最流行的是 **GNU Emacs**，它是由 **GNU** 工程在线发布的（请参阅[参考资料](#)）。

您必须拥有较新版本的 **GNU Emacs**，即版本 20 或更高版本。版本 20 和 21 是最常见的，您也可以获得版本 22 的开发快照。本教程可与这些版本的 **Emacs** 中的任何一种配合使用。如果您的系统运行的是较老的版本，您应该进行升级。

为了了解您运行的是什么版本的 **Emacs**，请使用 **GNU** 风格的 `--version` 标志，如下所示：

```
$ emacs --version
```

```
GNU Emacs 22.0.91.1
```

```
Copyright (C) 2006 Free Software Foundation, Inc.
```

GNU Emacs comes with ABSOLUTELY NO WARRANTY.

You may redistribute copies of Emacs

under the terms of the GNU General Public License.

For more information about these matters, see the file named COPYING.

\$

第 1 部分: 学习 Emacs 的基础知识

介绍 Emacs 的编辑环境

您首先应当知道的是, 在 UNIX 中最流行的两种编辑器是 Emacs 和 vi, 两者都有超过 30 年的历史。不过, 它们都是很难过时的工具。和 UNIX 一样, 这些编辑器的基本设计已有几十年之久, 但是在最先进的开发中, 它们仍然得到了广泛的应用。

Emacs 是历史上最早的开放源代码和免费软件工程之一。它的发明者 Richard Stallman 建立了 GNU 工程及其父组织—自由软件基金会 (请参阅[参考资料](#))。甚至在 Stallman 发布 GNU 公共许可 (GNU GPL, 一种许可条款, 如今的大多数开放源代码软件都是在该条款下发布的) 之前, 他就已经在类似的免费 copyleft 软件许可 (被称为 EMACS 公共许可) 下发布了 Emacs 的源代码。

Emacs 这个名字已经成为专有名词, 但它最初是一个首字母缩写词, 代表 **Editing MACros**; Stallman 最初实现的是用 TECO 语言写成的一组宏。Emacs 现在是用 Emacs Lisp 编写的, 后者是一种优雅的高级编程语言。

GNU 工程的简要介绍中称 Emacs 是一种自身配备了相关说明文档的可扩展文本编辑器 (the **extensible self-documenting text editor**)。Emacs 是可扩展的, 这意味着可以在已有功能的基础上添加或构建新功能。之所以有这种可能, 是因为它是用 Lisp 编写的, 您可以编写新的 Emacs Lisp 例程以添加新功能。您甚至能在 Emacs 会话还在运行时运行这些新功能。它自身配备了相关说明文档, 因为每个按键都有相应的即时帮助, 即使在键入一个命令时也能显示帮助信息。这样, 您可以单击 Help 按钮, 弹出各种可能情况的列表。

Emacs 被称为一个编辑环境, 因为它不仅仅是一个普通意义上的、用于纯文本编辑的编辑器。许多管理员和开发人员在各种平台上用它来编译和调试程序、管理电子邮件、在系统中操作文件、运行 shell 命令, 以及完成很多其他工作。人们甚至用它在 Usenet 新闻组中进行交流, 还用它来浏览网页。扩展包和内置插件可以处理从 Internet 中继聊天 (Internet Relay Chat, IRC) 和发送消息到网络通信的各种情况。一个流传很久的 UNIX 笑话是这样说的: “如果 Emacs 环境里有一个好的编辑器的话, 它就不至于这么糟糕了。”

因为很多任务都可以用 Emacs 来完成, 所以它也有自己的词汇表, 当您在这一部分中深入了解典型的 Emacs 窗口时, 您将学到这些词汇。这一部分还介绍了如何启动和停止 Emacs, 以及如何键入各种命令。

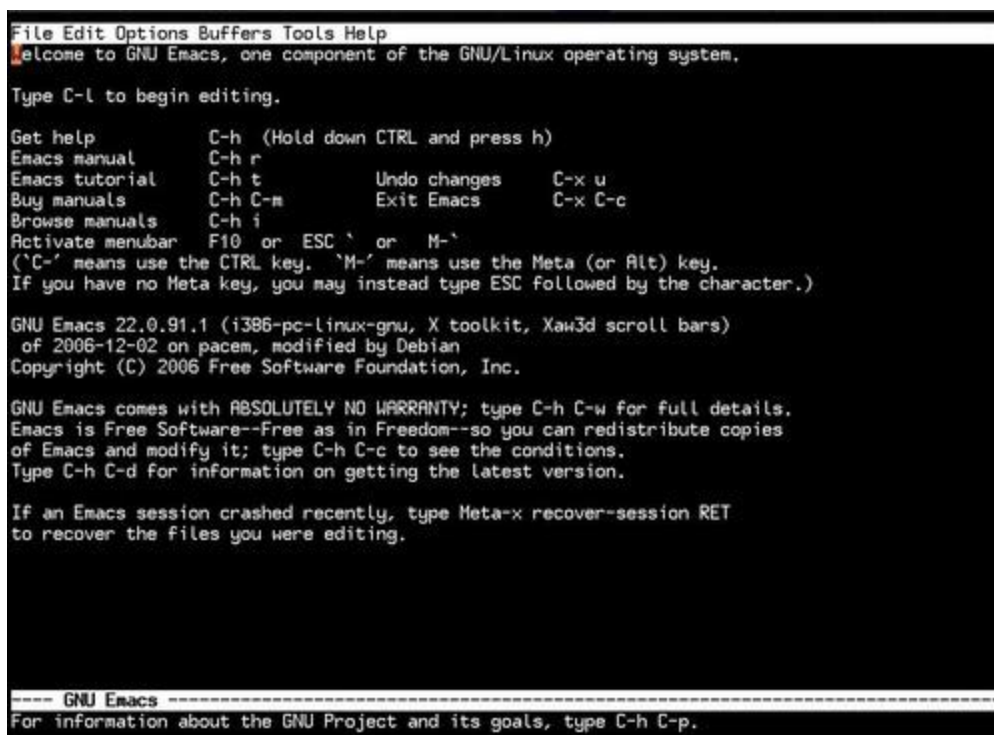
启动 Emacs

要从 shell 中启动 Emacs, 键入:

emacs

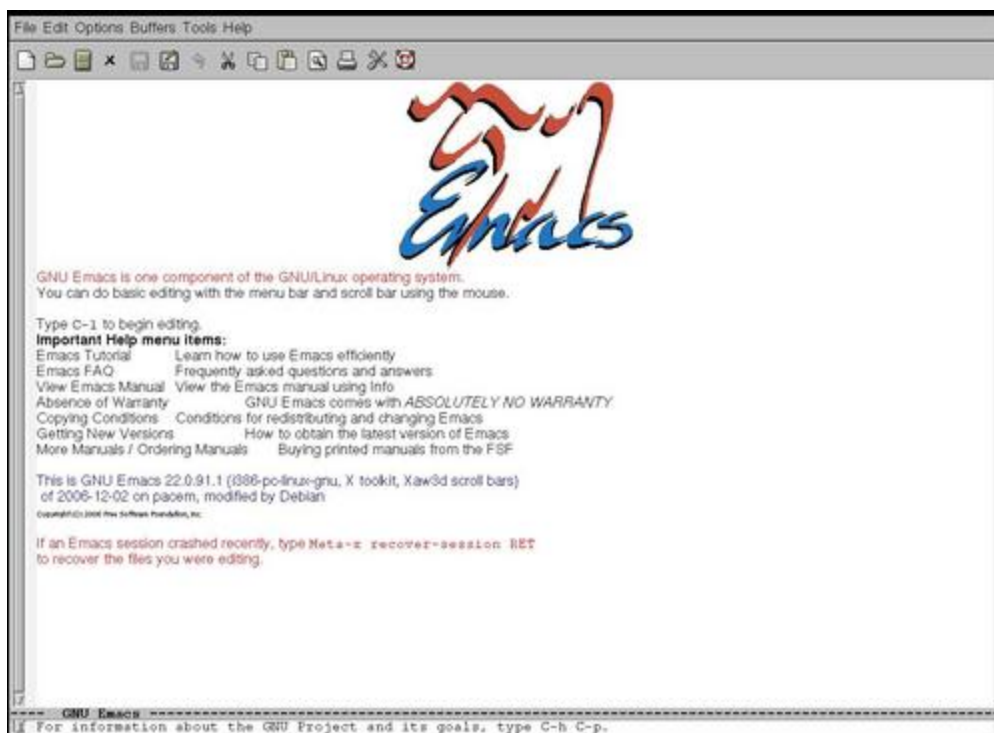
如果您在控制台或终端中启动 Emacs, 您会看到它打开后充满了整个终端窗口, 如[图 1](#) 所示。

图 1. 终端窗口中的 Emacs



如果您是在 X 客户端，Emacs 一般会在属于它自己的窗口中打开，如图 2 所示。您还可以指定让它在某个终端窗口打开（如图 1 所示），方法是使用 `-nw` 选项。乍一看，Emacs 的这两个视图似乎是不同的应用程序，但两者只有表面的差别。它们仅有的真正不同之处在于缺省的颜色、X 客户端显示的图形徽标，以及 X 客户端顶部附近的图标组（包含某些最常用命令的快捷方式）。两者的文本功能和 X 版本是一模一样的。

图 2. Emacs, X 客户端



Emacs 窗口剖析

这个屏幕的有些部分需要现在就解释一下。我们从顶部开始，向下介绍。
菜单栏

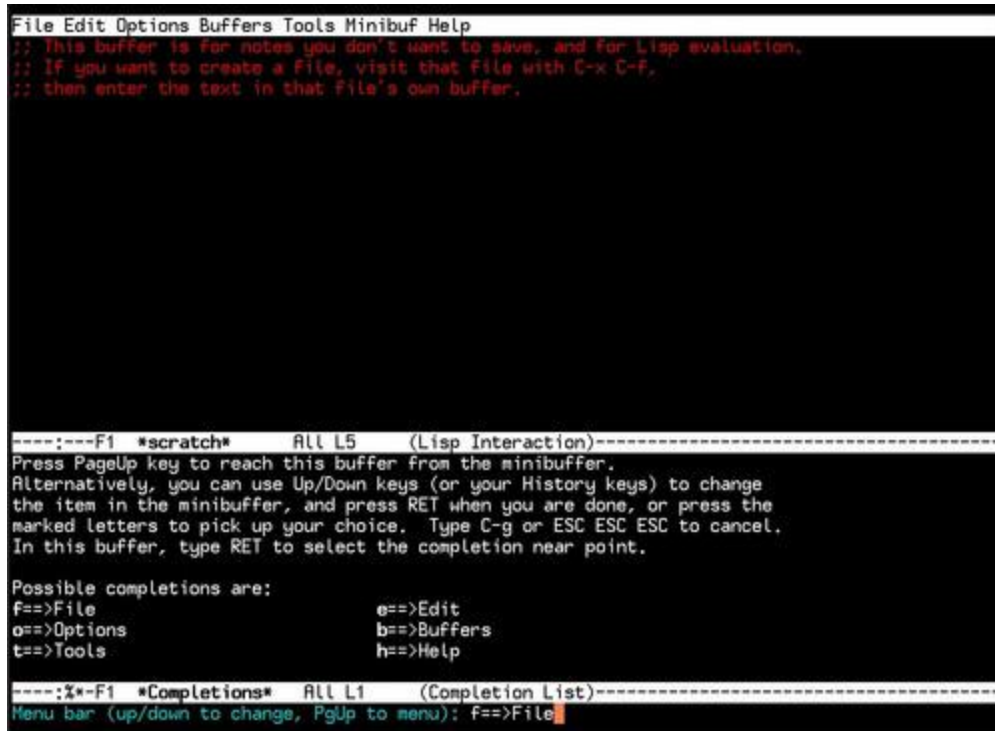
在 Emacs 屏幕顶部是一个突出显示的条，上面有一些单词。这被称为菜单栏，您可以在菜单中选择常用的 Emacs 命令。您能用键盘访问这些菜单；在 X 客户端中，您还可以用鼠标展开菜单。

专家们通常会配置他们的 Emacs，使它关闭菜单栏，好在屏幕上留出更大的编辑空间。但是在您学习 Emacs 时，菜单栏是帮您熟悉其丰富功能的好方法。

此外，如果您在 X 中打开了 Emacs，您会看到顶部有一组特殊的图标（请参见图 2）；它们是某些最常用的菜单选项的快捷方式。

要用键盘访问菜单栏，请按 F10。现在就试试，看看终端中是如何打开一个新的窗口并显示菜单列表的，如图 3 所示。

图 3. 从 Emacs 菜单栏中进行选择



您可以使用向上或向下的方向键，在菜单选项中移动，然后按 Enter。如果您选择的项目有一个子菜单，它会显示在屏幕上。您可以用同样的方法在新的子菜单中选择一个项目，直到您选中了要运行的命令为止。

如果您想终止这一过程，可以随时按下 Ctrl-G。这个特殊的键盘输入会使计算机发出一声蜂鸣声，并退出当前正在执行的任何命令。如果没有正在执行的命令，按下这个键盘输入只会使计算机发出蜂鸣声。请现在尝试一下。

您会看到 Emacs 窗口恢复成在您选择菜单之前的样子。

窗口

屏幕中央大片的主要区域被称为 Emacs 窗口，它是您进行编辑工作的地方。当您打开一个要编辑的文件时，这里将显示该文件的内容。当文件或文档中的内容显示在 Emacs 窗口中时，它被称为缓冲区。您可以随时在 Emacs 中同时打开多个缓冲区（即使它们不会显示在主窗口中也一样）。在编辑会话中，虽然只有一个缓冲区会显示在窗口中，但您一般仍会打开多个缓冲区。

在 X 客户端中，在窗口的左边会显示一个滚动条。（终端的版本也会出现，但它仅在打开某个缓冲区时显示。）滚动条显示 Emacs 窗口中的文本与缓冲区剩余部分的位置关系与大小关系。

模式行

在每个 Emacs 窗口中都有一条横贯底部的高亮条，它被称为模式行，您可以把它看成是状态栏。它将为提供有关 Emacs 会话和当前缓冲区（在上面的窗口中显示）的信息，包括您做出的

最新修改是否已被保存到磁盘、光标所在行的行号、屏幕底部显示的内容在缓冲区中的位置（用占总体的百分比表示），以及当前有哪些 Emacs 功能和设置处于活动状态。

迷你缓冲区 (minibuffer)

在模式行下面和屏幕（或 X 客户端窗口）底部的一片小空间被称为迷你缓冲区 (minibuffer)。这是 Emacs 用来显示与操作相关的消息的地方。当 Emacs 要求您输入某种内容（如某文件的名称）时，将在此处显示。

与 UNIX shell 类似，迷你缓冲区使用制表符作为提示符。按下 Tab 键，可获得一个可能情况的列表。

学习如何键入 Emacs 的键绑定

一个用来调用特定命令的 Emacs 组合键被称为一个键绑定。这些都可以自定义，但 Emacs 也附有缺省的绑定。

一眼看去，为数众多的 Emacs 键绑定似乎很复杂，让人不知所措，不过请务必记住，它们是为了提高您的速度和便于记忆而专门设计的。通常每个绑定都有一个记忆的方法，如 S 键就是用来保存 (save) 的。在您尝试使用它们时，请想想这一点。

总的说来，Emacs 命令可分为两大类：一类要使用 Ctrl 键，而另一类使用 Meta 键。

学习如何键入 Ctrl 组合键

许多 Emacs 命令是由某个 Ctrl 组合键指定的。在 Emacs 的符号中，Ctrl 键写成“C-”，后面则是与第二个按键对应的字符。例如，在 Emacs 符号中，Ctrl-X 组合键写成“C-x”。

要输入一个 Ctrl 组合键，请按住 Ctrl 键，然后按第二个键，然后将两个键同时松开。大多数命令都有一个 Ctrl 组合键，在按下此组合键后要再输入一个单词或第二个 Ctrl 组合键。

例如，可以尝试运行“C-x C-s”命令，将当前的缓冲区保存到磁盘。因为您没有作出更改，所以也无需进行保存，但这个组合键值得一试。现在，请尝试进行以下操作：

按住 Ctrl 键。

按 X 键，然后把两个键同时松开。

再按住 Ctrl 键。

按 S 键，然后把两个键同时松开。

这个键盘输入将运行“save-buffer”命令。在迷你缓冲区中，Emacs 会报告“(No changes need to be saved)”。

Emacs 用户在按这些组合键时，在第二个步骤中往往不会松开 Ctrl 键，这样可以省去第三个步骤，加快键入速度。请尝试一下。

学习如何键入 Meta 组合键

Emacs 键盘输入中的第二种主要类型是 Meta 键；在 Emacs 符号中，Meta 按键被表示为“M-”。如果您从未听说过 Meta 键，那是因为目前的大多系统并没有这个键。输入 Meta 组合键，有三种方法：

Meta 键常被绑定到 Alt 键，使用方法与 Ctrl 键类似。如果您的设置就是这样，请使用该设置，这是最简单、最常见的方法。

通常您可以使用 Esc 键完成一个 Meta 按键序列，但您的操作与输入一个 Ctrl 按键序列时不同。按下 Esc，然后松开，再按第二个键。

您可以用 Ctrl-[代替 Esc 键。如果您通过网络线路运行 Emacs，无法使用 Esc 和 Alt 键，这种方法会很方便。

请尝试输入 `M-b` 命令，这会把光标向回移动一个单词，有三种操作方法：

按住 `Alt` 键，然后按一下 `B`。

按下 `Esc`，然后松开，再按一下 `B`。

按住 `Ctrl` 键，再按 `[`，同时松开两个键，然后再按一下 `B`。

您运行的每个 Emacs 命令都是一个函数，由 Emacs Lisp 定义，并有一个函数名。甚至连用向上方向键将光标向上移动一行也是一个函数（`previous-line`）。您可以使用 `M-x` 命令，然后再输入函数名，运行任意一个函数。

请尝试以下操作：

按住 `Alt` 键。

按 `X` 键。

同时松开两个键，您会注意到缓冲区中会出现“M-x”字样。

键入 `previous-line` 并按 `Enter`。

当您这样操作时，光标会向上移动一行。当然，您平常是不会这样运行这个函数的，因为使用向上方向键比这容易多了，不过这是个好例子。函数为数众多，而且由于 Emacs 是可扩展的，您可以编写自己的函数以扩展其功能。

与通过向上方向键运行 `previous-line` 类似的是，许多函数都分配了快捷键。当您输入这样的键或组合键时，将运行相应的函数。您还可以使用 `C-p` 运行 `previous-line` 函数。现在，请尝试下面的操作：

按住 `Ctrl`。

按 `P` 键。

再按方向键，将光标移到窗口的顶部。

作为总结，表 1 列出了您可以在 Emacs 键绑定中使用的主要按键前缀类型。和 Emacs 中的其他东西一样，这些都是可以自定义和重新定义的。

表 1. 常用的缺省 Emacs 按键前缀

按键前缀	描述
<code>C-c</code>	当前编辑模式特有的命令
<code>C-x</code>	文件和缓冲区命令
<code>C-h</code>	帮助命令
<code>M-x</code>	函数名称

停止 Emacs

要退出 Emacs，请键入：`C-x C-c`

如果有未保存的缓冲区，这个命令会使您有机会保存它们。

这是退出 Emacs 的一般方法。请现在就试着操作一下，然后在 shell 的提示符下键入

`emacs`，重新启动 Emacs。

缓冲区和文件

在这一部分中，将介绍最重要的 Emacs 缓冲区和文件命令：如何将文件载入缓冲区，如何将缓冲区保存到文件，如何在缓冲区间切换，以及如何“杀死”缓冲区。

从头创建一个新文件

当您用普通的方式启动 Emacs 时，它将向一个名为 `scratch` 的缓冲区开放，该缓冲区的用途显示在缓冲区顶部的一条消息中：

```
;; This buffer is for notes you don't want to save, and for Lisp evaluation.
```

```
;; If you want to create a file, visit that file with C-x C-f,
```

```
;; then enter the text in that file's own buffer.
```

模式行左边的区域始终用来显示当前缓冲区的名称。在此处，您可以看到这个缓冲区的名称，即 `*scratch*`。如果是 Emacs 自动创建的特殊缓冲区，其名称两边会有星号。

试着输入一行文本：`This is a practice file.` 然后再按 `Enter`，将光标向下移到一个新行。当您开始输入时，您会看到模式行的左侧出现两个星号，这表明当前的缓冲区中有未保存到磁盘的文本。

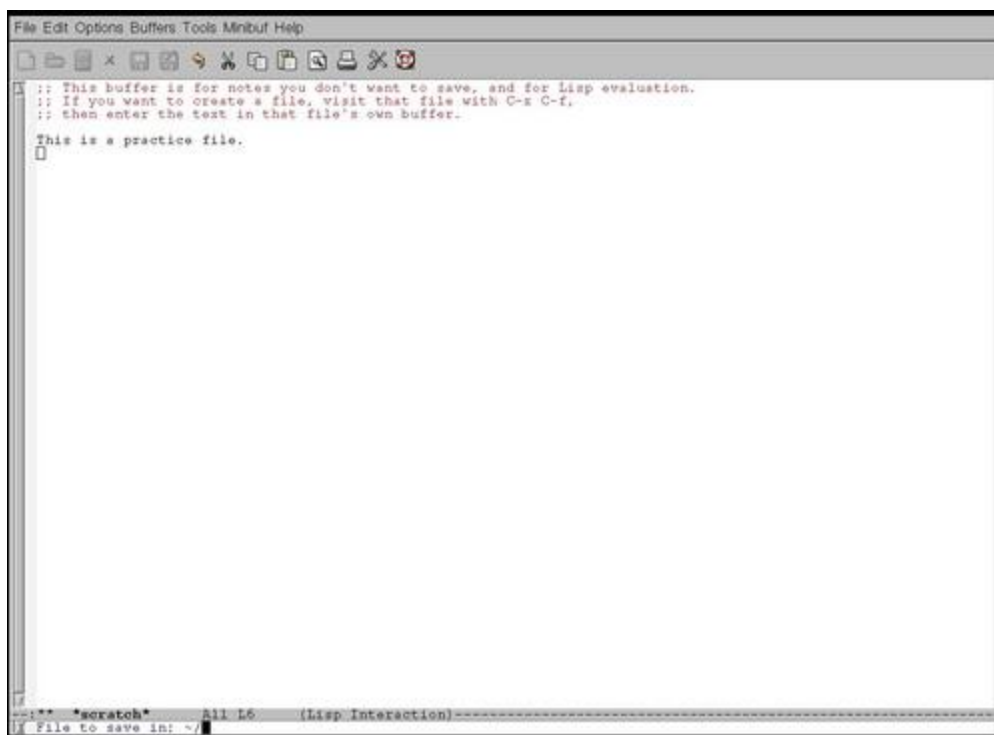
创建新文件的方法之一是将 `scratch` 缓冲区写到某个文件中。组合键 `C-x C-s` 会运行

`save-buffer`

命令，将当前的缓冲区保存到某个文件。正如您先前看到的，当您在一个无需保存的缓冲区中运行此命令时，Emacs 会显示相应的信息。但如果在您运行该命令的缓冲区中有未保存的更改，且该缓冲区与磁盘上的某个特定文件相关联时，缓冲区的内容会被写入该文件。如果您是在新缓冲区或某个没有与文件关联的缓冲区（例如您现在所在的 `scratch` 缓冲区）中运行该命令，Emacs 将提示您指定用来保存该缓冲区的文件名称。

现在，请尝试下面的操作：键入 `Ctrl-x Ctrl-s`。看看在迷你缓冲区中是如何要求您给出要保存的文件名称的，如[图 4](#) 所示。现在键入 `practice`。

图 4. 保存 `scratch` 缓冲区



在您按下 **Enter** 后, Emacs 会在迷你缓冲区中报告, 有一个名为 **practice** 的新文件已被写入磁盘。键入 **C-x C-c**, 退出 Emacs, 然后查看目录, 找到您的文件:

```
$ ls
```

```
practice
```

```
$
```

关于 **scratch** 缓冲区, 还有一件事是需要记住的。如果 Emacs 中没有打开其他缓冲区, 那么 **scratch** 缓冲区会一直存在, 当它是打开的唯一缓冲区时, 是无法被关闭的。

利用文件名启动 Emacs

要使 Emacs 启动时将某个特定的文件的内容放进一个新的缓冲区中以便编辑, 请提供此文件的名称作为参数。如果您提供了多个文件, 则每个文件会在属于它自己的缓冲区中打开。如果您提供的文件名在磁盘上不存在, Emacs 会为该文件创建一个新的缓冲区, 并说明(在模式行中)这是一个新文件。这是您在启动 Emacs 并编辑一个新文件时所做的。当您把缓冲区保存到磁盘时, Emacs 将其写入文件, 并使用您提供的参数做为它的文件名。

请试着用您的文件“**practice**”启动 Emacs:

```
$ emacs practice
```

您仍会看到 Emacs 的欢迎屏幕, 但如果您按下一个键, 如 **C-g** (或等足够长的时间), 您会在 Emacs 窗口中看到您文件。没错, 欢迎屏幕是另一个可供配置的选项。

打开一个文件

Emacs 不会对文件的内容进行直接操作。它会把文件内容的副本读取到您编辑的缓冲区中。使用 **C-x C-f**, 即 **find-file** 命令, 以便用文件的内容打开一个缓冲区。

现在, 请试着用这个命令打开您的文件:

键入 `C-x C-c`，退出 Emacs。

再次启动 Emacs，但这一回不指定您的文件：

`$ emacs`

键入 `C-x C-f`，当您缓冲区中有相应要求时，请您提供文件的名称 (practice)。

访问缓冲区

`C-x b`

命令可以从当前缓冲区切换到您指定的另一个缓冲区。迷你缓冲区中始终会提供一个缺省的缓冲区选项。如果您按下 **Enter** 且没有指定缓冲区的名称，您会切换到上述的缓冲区。缺省值一般是您上次访问的缓冲区。如果您之前没有访问任何缓冲区，则缺省值通常为 **scratch** 缓冲区。

现在键入 `C-x b`，注意 **scratch** 缓冲区会作为建议项显示在迷你缓冲区中。键入 **Enter**，您的 **practice** 文件将从窗口中消失，代之以您熟悉的 **scratch** 缓冲区中的内容。（在缺省情况下，**scratch** 缓冲区会包含三行消息，但有时该缓冲区为空白。）再次键入 `C-x b`，然后按 **Enter**，切换回 **practice** 缓冲区。

在 Emacs 中，如果您希望创建一个新的缓冲区，请切换到您要指定名称的那个缓冲区。

再次键入 `C-x b`，不过要将该缓冲区命名为 **mybuffer**。注意，窗口是空的，这是一个全新的缓冲区。键入一行文本，然后按 **Enter**：

On what wings dare we aspire?

键入 `C-x C-s`，将这个缓冲区保存到磁盘。注意，现在 Emacs 会要求您指定一个文件名。

如果您创建的新缓冲区没有与磁盘上的某个文件关联，您可以在决定保存它时指定一个文件名。您指定的文件名不必与缓冲区名称相同，可以把它命名为 **practice.b**，请注意，Emacs 会改变缓冲区的名称，使之与新文件相符。

杀死缓冲区

请使用 `C-x k` 命令杀死某个缓冲区，或将其排除在您的 Emacs 会话之外。您将被提示键入要杀死的缓冲区的名称，当前的缓冲区是缺省缓冲区，如果您直接按下 **Enter**，会将它杀死。

现在试着杀死旧的 **practice** 缓冲区：键入 `C-x k`，然后当 Emacs 在迷你缓冲区中要求输入一个缓冲区名称时，键入 **practice**，然后按 **Enter**。**practice** 缓冲区会被杀死，而 **practice.b** 缓冲区则像刚才一样留在窗口中。

将缓冲区保存到磁盘

您已经知道了如何通过运行 `C-x C-s`

`save-buffer`

命令，将缓冲区的内容保存到磁盘。您需要了解这一过程的更多信息。

模式行的前几个字符描述了缓冲区的状态。现在，在您的 **practice.b** 缓冲区中键入另一行文本，然后按 **Enter**：

What the hand dare seize the fire?

注意，模式行中的两个破折号会变成两个星号。破折号说明缓冲区的内容与磁盘中的相同，而星号则表示缓冲区中有未保存的编辑内容。

保存更改并退出 Emacs：键入 `C-x C-s C-x C-c`。您的目录中会多出几个新文件：

\$ ls

practice

practice.b

practice.b~

`practice` 和 `practice.b` 文件是您创建的，但 `practice.b~` 则是一个由 Emacs 自动创建的文件。这是 Emacs 的备份文件，当您每次在 Emacs 中编辑一个已经存在的文件时都会创建这个备份文件。当您编辑内容并保存文件时，Emacs 会写到一个新文件中以制作备份，这个新文件的名称与原先的文件相同，只是多了一个波浪符号。旧的 `practice` 文件没有备份，因为您在创建它之后从未对其进行编辑。如果您此后在 Emacs 中编辑了它，Emacs 会将内容写入相应的 `practice~` 备份文件。

Emacs 还会写入另一种文件，被称为 `autosave` 文件，它与原先的文件名称相同，但是在名称前后会各添加一个英镑符号。在您操作缓冲区期间，Emacs 会按设置的间隔向 `autosave` 文件写入内容。在缺省情况下是每输入 300 个新字符就向 `autosave` 文件写入一次。通常，在您杀死缓冲区时是不会见到 `autosave` 文件的。这是因为与缓冲区关联的 `autosave` 文件被删除了。如果您的系统崩溃，或您在编辑会话中失去连接，`autosave` 文件很有用，它为您提供了一个选择，使您可以在目录下找到这个文件以恢复丢失的编辑内容。

缓冲区和文件命令总结

表 2 包含的是某些最常用的缓冲区和文件命令列表，在您学习 Emacs 的过程中将用到这些命令。该表提供了命令绑定的键盘输入和命令的函数名。请记住，您总能使用相应的键绑定，或将相应的函数名作为 `M-x` 的参数，以运行某个命令（请参考[学习如何键入 Meta 组合键](#)）。

表 2. 常用的 Emacs 缓冲区和文件函数

绑定	函数名	描述
<code>C-x C-s</code>	<code>save-buffer</code>	将当前的缓冲区保存到磁盘。
<code>C-x s</code>	<code>save-some-buffers</code>	要求将所有未保存的缓冲区保存到磁盘。
<code>C-x C-c</code>	<code>save-buffers-kill-emacs</code>	要求将所有未保存的缓冲区保存到磁盘，并退出 Emacs。
<code>C-x C-z</code>	<code>suspend-emacs</code>	挂起 Emacs 并使之成为一个后台进程。

<code>C-x C-b</code>	<code>list-buffers</code>	列出所有缓冲区。
<code>C-x k</code>	<code>kill-buffer</code>	杀死一个缓冲区（缺省情况下为当前的缓冲区）。
<code>C-x C-q</code>	<code>vc-toggle-read-only</code>	切换当前缓冲区的可读状态（如果适用还可以执行版本控制）。
<code>C-x i</code>	<code>insert-file</code>	在 插入点 插入某个文件的内容。

在 Emacs 中编辑文本

在缓冲区中输入和更改文本，以及在文本中进行导航，是您在 Emacs 中最重要的操作，无论您是在编辑文件，创建新文件，还是仅仅想研读某个文件，都要用到它们。

在此部分，您将学习用来完成下述工作的基本按键序列和命令：如何在缓冲区中输入文本，如何在文本中导航，以及如何对文本进行基本的编辑，如删除字符和单词。

键入文本和在缓冲区中移动

正如您在[从头创建一个新文件](#)这一部分看到的，在 Emacs 缓冲区中输入文本是很容易的，您只要动手打字就行。您可以键入字母字符，将其输入某个缓冲区中。

Emacs 有时会被称为无模式编辑器，常被拿来与模式编辑器如 vi 等作比较。这意味着编辑器的行为以及您可以键入的键盘输入和命令在您的会话中是保持不变的，与 vi 等编辑器不同，后者的击键会根据您是在命令模式还是在输入模式而有不同的含义。Emacs 没有这类模式；不过，它确实有一种不同的模式，可以用来更改其行为或扩展其功能，这将是本系列的下一篇文章的主题。

在进行普通的输入时，有些事是您必须记住的。

在插入点插入文本

在 Emacs，有一个重要的概念，被称为插入点，它表示字符的插入位置。这是缓冲区中一个想像的位置，处于光标所在字符和前一个字符之间。

每当您在缓冲区中输入文本时，该文本都会在这个点插入。在缺省情况下，所有文本都在同一行，在这个点之后的文本会向右推移，为您插入的内容让出空间。按 Enter，移到下一行，然后再按一下 Enter，插入一个空白行。

请试着在 practice.b 的开始处插入一段文本。

启动 Emacs 时打开该缓冲区：

```
$ emacs practice.b
```

在每一行的末尾按 Enter，以输入一个段落，然后在结尾再按一下 Enter，创建一个空白行。

```
Tyger! Tyger! burning bright
```

```
In the forests of the night,
```

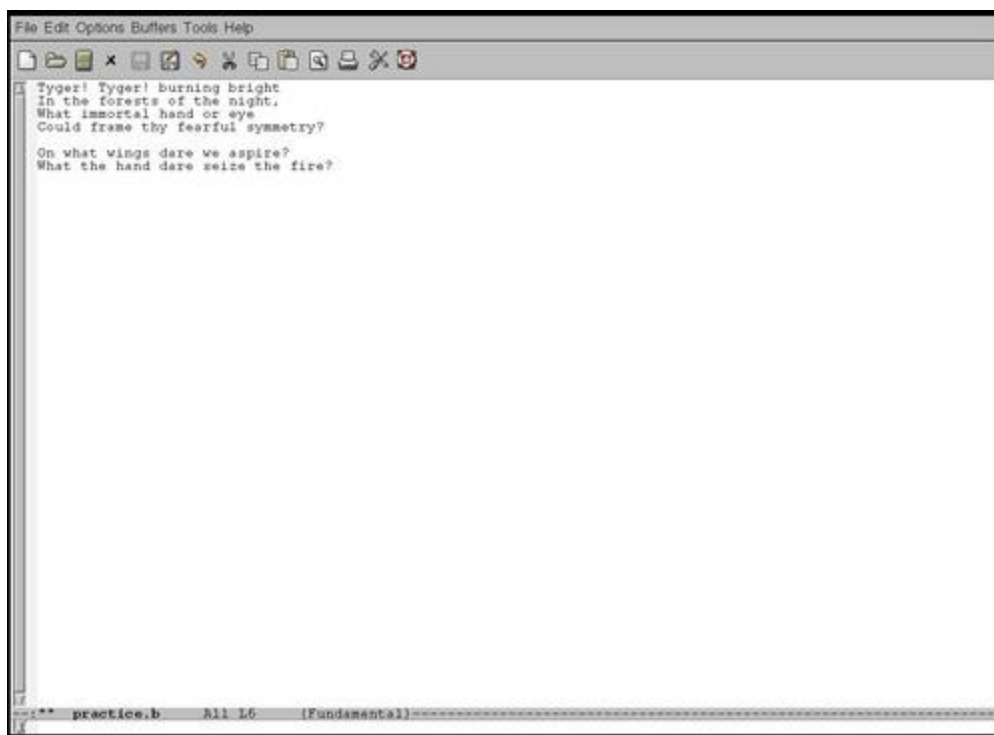
```
What immortal hand or eye
```

```
Could frame thy fearful symmetry?
```

注意：这是 The Tyger! 这首诗的第一节，作者是 William Blake。本文作者在教程中多次引用本诗的某几个小节。

现在您的 practice.b 缓冲区应如[图 5](#) 所示。

图 5. 插入一个段落



移动插入点

您会看到，在您键入时光标会一直跟随，插入点保持不变：从您键入第一个句子开始，您键入的一切内容都被插入字母 **O** 之前。

要移动插入点，您可以使用方向键，而且正如您预期的那样，所有其他与光标动作有关的键都可供使用，如 **PgUp**、**PgDn**、**Home** 和 **End**。但 Emacs 自带用来移动光标的键绑定，而且因为您不必将手移到键盘的基准键之外就能使用它们，所以您在打字时会觉得它们非常有用。

正如您在[学习如何键入 Meta 组合键](#)部分中看到的，您可以使用 **C-p** 将光标向上移动到上一行；类似地，**C-n** 将光标向下移动到下一行。**C-f** 会向前移动到下一个字符，而 **C-b** 会向后移动到上一个字符。

如果不想用 **PgDn** 和 **PgUp** 键向上和向下移动一屏，请使用 **C-v** 和 **M-v** 这两种键盘输入，它们可起到与前两个键相同的作用。要转至当前行的开始处，请使用 **C-a**；使用 **C-e**，转到当前行的结尾。

通常 **Meta** 键会被绑定到某个命令，这与相应的 **Ctrl** 键类似，对于移动命令，这条规则也同样适用。当使用 **Meta** 来代替 **Ctrl** 时，**F** 和 **B** 键可以向前和向后移动一个单词 而不是一个字符，而 **A** 和 **E** 键可以移动到当前句子的开头和结尾。（在缺省配置中，没有定义 **M-n** 和 **M-p** 键盘输入。）

[表 3](#) 列出了各种移动和导航的键，以及它们的函数名和描述。试着用它们移动到缓冲区的开头、结尾，和中间的某些位置。

表 3. 有用的 Emacs 键盘输入（用于移动和导航）

键盘输入	函数	描述
C-p , UpArrow	previous-line	将插入点向上移动到上一行。

C-n , DownArrow	next-line	将插入点向下移动到下一行。
C-f , RightArrow	forward-char	将插入点移动到下一个字符。
C-b , LeftArrow	back-char	将插入点移动到上一个字符。
M-f	forward-word	将插入点移动到下一个单词。
M-b	backward-word	将插入点移动到上一个单词。
C-v , PgDn	scroll-up	将文本向上滚动一屏。
M-v , PgUp	scroll-down	将文本向下滚动一屏。
Home	beginning-of-buffer	将插入点移到缓冲区的开始处。（在某些版本中，这个键被缺省定义为移动到当前行的开始处。）
End	end-of-buffer	将插入点移到缓冲区的末尾。（在某些版本中，这个键被缺省定义为移动到当前行的末尾。）
C-a	beginning-of-line	将插入点移到本行的开始处。
C-e	end-of-line	将插入点移到本行的结尾。
M-a	beginning-of-sentence	将插入点移到句子的开始处。
M-e	end-of-sentence	将插入点移到句子的结尾处。
C-{	beginning-of-paragraph	将插入点移到段落的开始处。
C-}	end-of-paragraph	将插入点移到段落的结尾处。

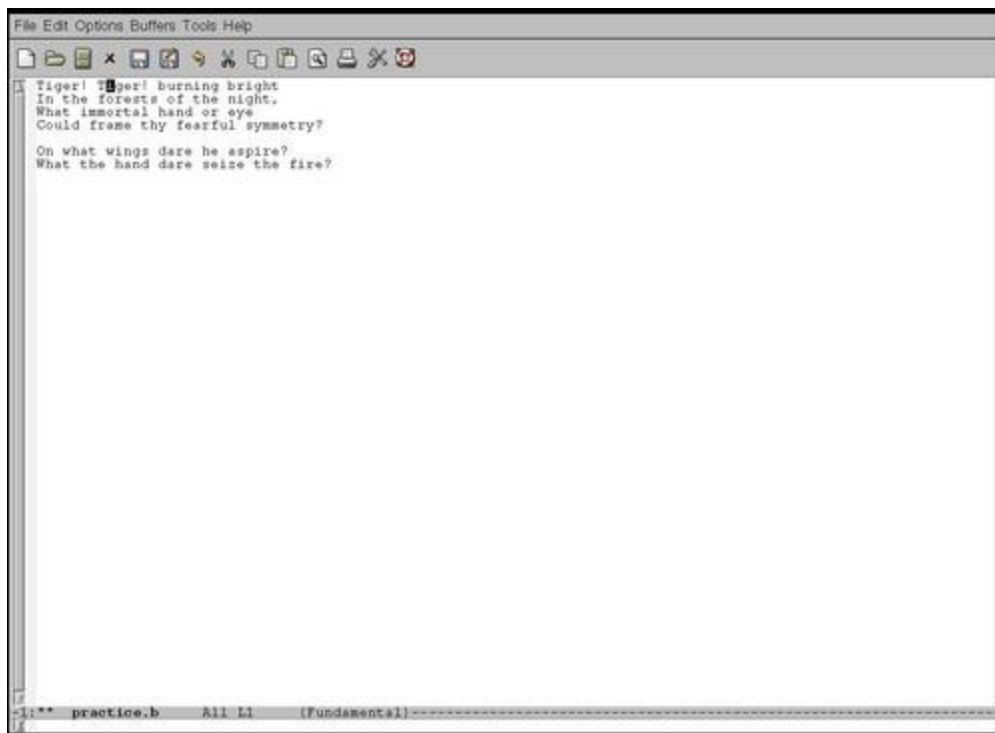
使用改写模式

正如您看到的，您键入的文本被插入到插入点处。不过您也可以改写现有的文本。按一下 **Ins** 键；这将切换到改写模式，该模式在缺省情况下是关闭的。查看模式行，您会看到 **Ovwr**，说明该模式已处于活动状态。

使用[表 3](#) 中描述的移动命令，移动光标，使之处于 **we** 中的 **w** 处，然后键入一个 **h** 字符。

请试着转至缓冲区的顶部：键入 `M-a M-a C-f`，将光标移到 `y`，然后输入一个 `i` 字符。对于下一个 `y`，采取同样的操作，键入 `M-f C-f C-f C-f i`，以使缓冲区看起来如[图 6](#) 所示。

图 6. 改写缓冲区中的字符



再按一下 `Ins`，关闭改写模式。

引用插入

文本并不仅限于用字母键输入。您可以输入控制字符，也可以根据字符代码输入字符。为此您可以执行引用插入，该命令被绑定到 `C-q`；此后再按一个键（或组合键）如 `Ctrl` 组合键，以便在插入点输入该键。您还可以输入一个字符代码值（用八进制表示），然后再按 `Enter` 键。

移到缓冲区的末尾，键入 `Page break here`，然后按 `Enter`。现在键入一个分页符，即 `Ctrl-I` 代表的字符，在它的后面接一个转义字符，该转义字符的八进制字符值为 `033`：

`Enter` .

删除，撤消和重复

现在您可以试试 **Emacs** 中的一些工具，这些工具用来处理现有文本以及撤消（和重复）您已经做出的操作。

删除文本

使用 `Backspace` 或 `Del` 键，删除插入点左边的字符。试着用这两个键删除您刚才输入的两个控制字符。

要删除插入点的字符，请用 `C-d`；类似地，`M-d` 会从插入点开始，删除到单词的末尾。

您还可以向回删除，`M-Del` 和 `M-Backspace` 都可以从插入点一直删除到单词的开头。

试着用这些命令删除您刚刚输入的 **Page break here** 这个句子，然后删除这些单词所在的空行。

将光标移动到文件的最后一行（应该是“**What the hand dare seize the fire?**”这个句子），然后按几次 **M-d**，把这一行删除。

撤消和重复

啊呀，如果您并不想删除最后一句怎么办？您可以通过运行 **undo** 函数把它找回来。该函数被绑定到 **C-_**，您要键入它，请按住 **Ctrl**，并用 **Shift** 键输入下划线。请回到最后一行试试，每个单词操作一次，找回这些单词。

多操作几遍，直到 **Page break here** 重新显示为止。

现在您又对上次的撤消操作感到后悔了，您确实不想要 **Page break here** 了。您可以重复您撤消的操作，方法是键入 **C-g**，这将取消所有的撤消操作，然后多次键入 **C-_** 使单词再次消失。

多次运行同一个命令

C-u 是一个通用参数命令，后面接一个数字和一个命令。它将按某个次数多次运行特定的命令。

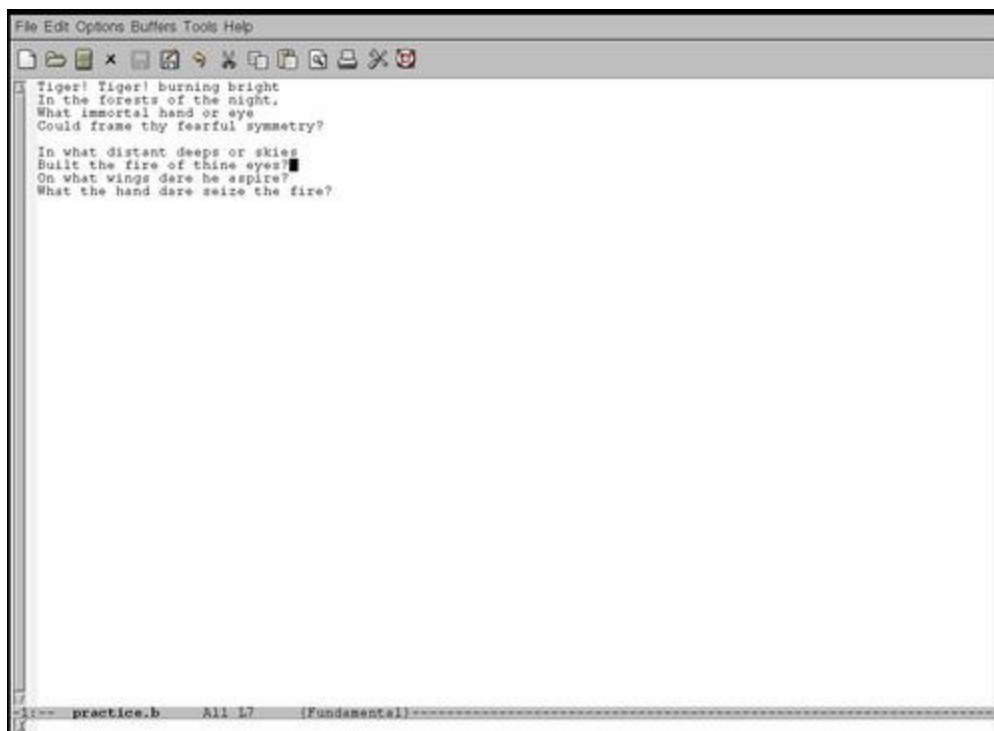
尝试以下操作：移动到第二句诗的开始处，键入 **In what distant d**，然后再键入 **C-u 2 e**，这将写入两个 **e** 字符。输入以下内容，完成诗句：

ps or skies

Built the fire of thine eyes?

键入 **C-x C-s**，保存您的缓冲区，现在的显示结果如 [图 7](#) 所示。

图 7. 使用通用参数缓冲区插入字符



在没有数字参数的情况下，`C-u` 会假定这个数字为 4。如果您将 `C-u` 本身作为参数，则 4 会自乘，得到 16；您可以在反复自乘任意多次之后再指定一个命令，如 `C-u C-u C-u A` 会在插入点处键入 64 个 A。

请用一个 `C-b`，向回移动，您会看到光标会回到前面的一行。现在试试通用参数，键入 `C-u C-b`；注意光标向回移动了四个字符（而不是一个）。再试试 `C-u C-u C-b`，注意光标会向左移动 16 个字符。键入 `C-u C-u C-u C-b`，然后再指定 1000 作为参数：`C-u 1000 C-b`。光标会移动到缓冲区的开始处，但请注意，Emacs 会发出蜂鸣声，这表明它在完成要求的操作次数之前就已经到了缓冲区的顶部（无法再向回移动了）。

用于编辑的按键表

我们来总结一下刚才学到了什么。在您继续学习下一部分之前，请看看[表 4](#)，它列出了重要的编辑命令和执行这些命令时用的缺省键盘输入。

表 4. 常用 Emacs 编辑命令

键盘输入	函数	描述
Ins	<code>overwrite-mode</code>	切换改写模式（缺省为关闭）。
Backspace, <code>Del</code>	<code>delete-backward-char</code>	删除插入点前的字符。
<code>C-d</code>	<code>delete-char</code>	删除插入点处的字符。
<code>M-d</code>	<code>kill-word</code>	从插入点开始向前删除字符，直到单词末尾。
<code>M-Backspace</code> , <code>M-Del</code>	<code>backward-kill-word</code>	从插入点开始向回删除字符，直至单词的开始处。
<code>C-_</code>	<code>undo</code>	撤消您的上一次键入或操作
<code>C-q</code> 字符 或 XXX	<code>quoted-insert</code>	在插入点插入按键本身代表的字符或由八进制数字（XXX）表示的字符。
<code>C-u</code> 次数 命令	<code>universal-argument</code>	按总的次数（缺省为 4 次）连续执行命令。

标记和鼠标

您正在顺利地学习关于 Emacs 编辑的所有基本知识，但还有一些重要的概念是您需要知道的：如何标记文本区域并在这些区域上执行操作，以及如何使用鼠标。

标记、移除、删除

Emacs 有一项标记文本区域的功能，您可以将这个区域作为一个整体进行编辑：[表 5](#) 中对这些命令进行了描述和简要介绍。

标记一个区域

移动到缓冲区的顶部，即诗的第一节的开始处，然后键入 **C-Space**，方法是按住 **Ctrl**，再按空格键。这被称为设置标记；迷你缓冲区中会出现一条消息，告诉您已经设置了标记。

插入点和您设置标记的位置之前的部分被称为区域。

将插入点由开始处移动到小节后的空白处，将整个第一节设置为区域

删除和恢复文本

有些特殊的命令可用来操作区域，包括 **C-w**，它能删除区域。

键入 **C-w**，删除您刚才定义的区域。

每次您删除文本时，这些文本都会保存在 **Emacs** 的 **kill ring** 中。您可以用 **C-y** 把它们恢复到插入点。移动到缓冲区的末尾，按 **Enter**，插入另一个空白行，然后将这个小节移回去。

您不仅能删除区域；使用 **C-k** 还可以删除从插入点到行末的所有文本。向上移到以 **Tiger** 开头的一行，然后用 **C-k** 删除它。注意，这一操作不会删除空白行；还要第二次使用 **C-k**。按此操作，然后使用 **C-y** 将整行移回去。如果您用多个删除命令连续进行删除，它们会叠加在一起，只返回一个移除内容。

您可以将某一行恢复任意多次。移动到缓冲区的顶部，然后再次使用 **C-y** 把它移回去。

您还可连续多次使用 **C-k** 键盘输入进行删除，删除的内容会被移到一起。请试试一次删除多行，然后使用 **C-y** 将它们移回原先的位置。

复制文本

如果您是想复制区域，不必删除它。如果要将区域保存在 **kill ring** 中而不是删除它，请使用

M-w（而不是 **C-w**）。

尝试下面的操作：

将插入点移到最后一节第二行的开头。

键入 **C-Space**，设置标记。

将插入点移到这一节的下面。

键入 **M-w**，将这三行复制到 **kill ring**，而不是删除它们。

将插入点向上移到缓冲区中第一行下的空行。

用 **C-y** 移除这三行。

此后您的缓冲区将如 [图 8](#) 所示。

图 8. 移除多行

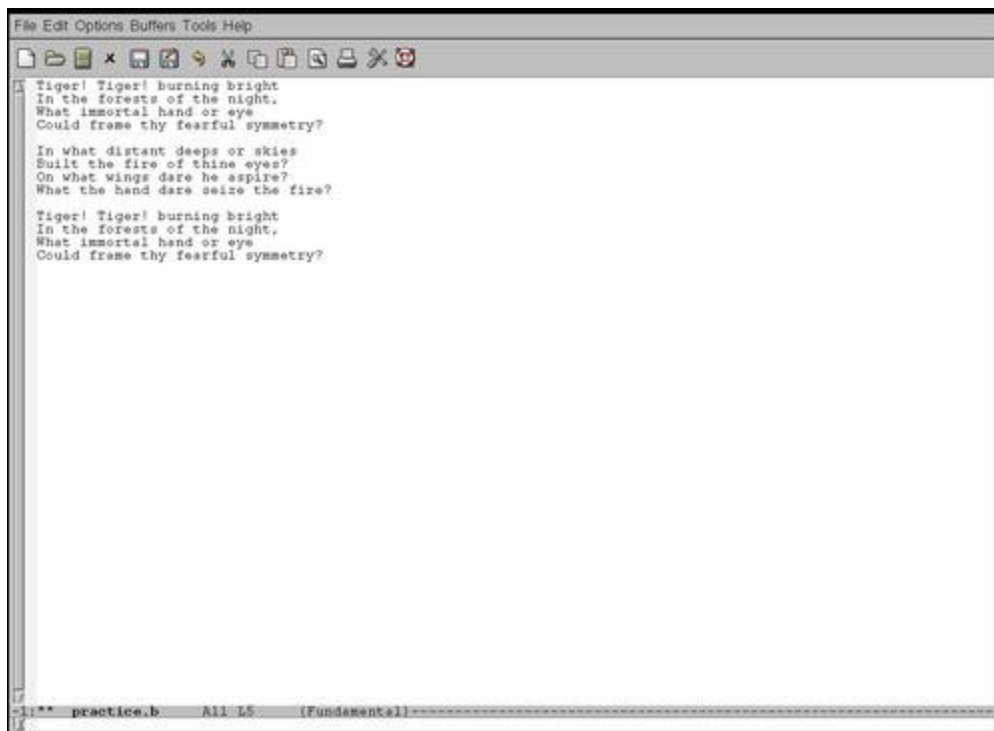


表 5. 用来标记和删除文本的 Emacs 函数

键盘输入	函数	描述
C-Space,C-@	set-mark-command	在插入点设置标记。
C-k	kill-line	删除从插入点到行末的所有文本。
C-w	kill-region	删除区域。
M-w	kill-ring-save	将区域保存在 kill ring 中，但不删除它。
C-y	yank	恢复来自 kill ring 的文本。

用鼠标进行标记和移动

虽然 Emacs 是为了用键盘快速操作而设计的，但是您也可以使用鼠标，这对文本操作来说有时是很方便的。

要将插入点移到缓冲区的任意位置，请把鼠标指针移到该位置，然后单击一下鼠标左键。试着用鼠标移动到诗的第二节和第三节之间的空白处，然后键入新的一节。

When the stars threw down their spears,

And watered heaven with their tears,

Did he smile his work to see?

Did he who made the Tiger make thee?

用左键双击一个单词以选中它。双击您刚刚键入的 **Tiger**，它将突出显示，然后按 **Del** 将它删除。现在键入 **Lamb**，将这个单词插入插入点处。

要用鼠标选择一整行，请用鼠标左键三击该行。请在顶行尝试此操作，然后按 **Del** 将该行删除。键入 **C-** 恢复删除的内容。

在缓冲区的中间单击鼠标左键，然后再输入两段，完成这首诗：

What the hammer? what the chain?

In what furnace was thy brain?

What the anvil? what dread grasp

Dare its deadly terrors clasp?

And what shoulder, and what art,

Could twist the sinews of thy heart?

And when thy heart began to beat,

What dread hand? and what dread feet?

您可以单击鼠标左键，然后拖动指针，以选择一个区域。您还可以按一下鼠标左键（将指针设置在某个位置），选择一个区域，然后在别处单击右键（将标记设置在此处）。当您这样操作时，您选择的文本无需删除就会被放置在 **kill ring** 中，您可以使用 **C-y** 或鼠标中键将文本复制到插入点处。要将突出显示的区域放在 **kill** 缓冲区中并删除它，请双击鼠标右键以设置区域。

试着调换您刚才键入的两节诗的次序：

在两节诗中间的空行的开头单击鼠标左键。

在您刚刚输入的第二节下面，双击鼠标右键。

在以 **What the hammer?** 开始的小节前的空行行首，单击鼠标中键。

键入 **C-x C-s**，将您的缓冲区保存到磁盘。您的 **Emacs** 会话应如[图 9](#) 所示。

图 9. 用鼠标插入

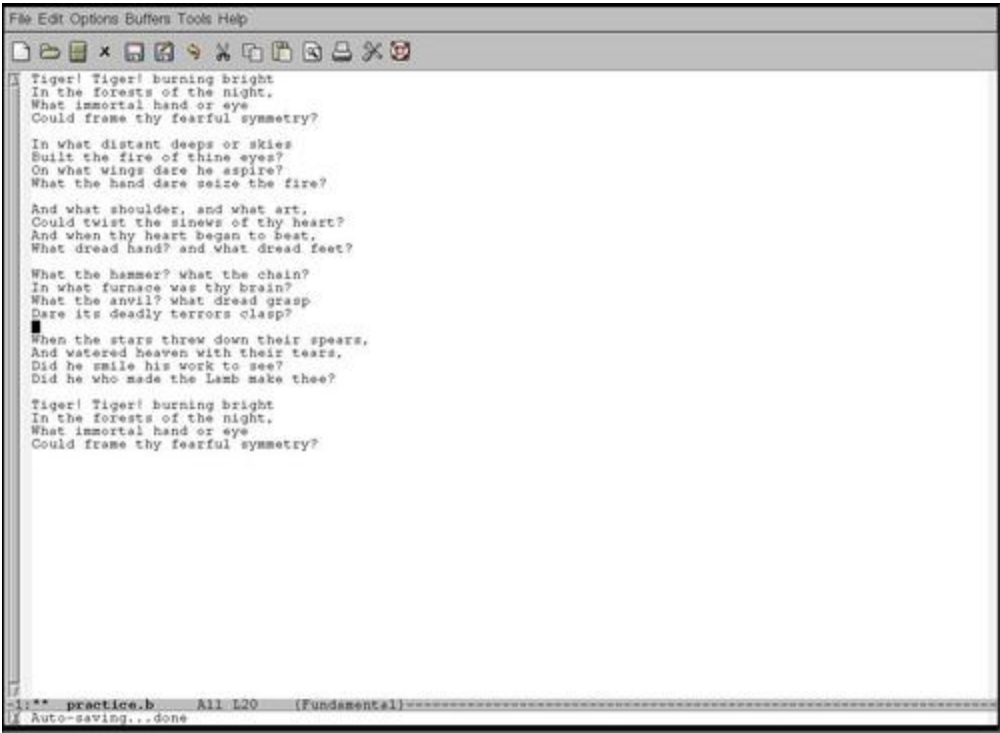


表 6 描述了 Emacs 中的各种鼠标操作。

表 6. Emacs 中的鼠标操作

鼠标命令	描述
<div>B1</div>	这一命令将设置插入点位置；拖动鼠标左键以设置区域。
<div>B1-B1</div>	这一命令标记一个单词。
<div>B1-B1-B1</div>	这一命令标记一行。
<div>B2</div>	这一命令插入最近被杀死的内容，并把光标移动到插入内容的末尾。
<div>B3</div>	这一命令会设置并突出显示区域，然后无需删除就将其放在 <code>kill</code> 缓冲区中。如果某个区域已经被突出显示并设置，该区域的末尾将移动到您单击的位置。
<div>B3-B3</div>	这个命令将突出显示区域，然后删除它。如果某个区域已经被突出显示并设置，该区域的末尾将移动到您单击的位置，此后该区域将被删除。

总结

总结

祝贺您！您已经完成了学习使用 **Emacs** 系列中的第一篇教程。您已经了解了有关 **Emacs** 编辑环境的所有基本知识：如何打开新缓冲区，并将它们保存到文件；如何在缓冲区中移动，输入和编辑文本，标记和操作文本区域；甚至包括如何使用鼠标进行文本操作。

虽然此教程涉及的内容很广，但您还有许多东西要学。本系列的后续教程将带您了解 **Emacs** 中复杂的编辑功能以及如何在工作中使用它们。

第 2 部分：学习 Emacs 的基本模式和编辑特性

编辑模式

Emacs 被划分为无模式的编辑器，这意味着它与其他编辑器（如 **vi**）有所不同，对于运行编辑器命令或者向缓冲区插入文本的插入模式，没有特殊的命令模式——正如您在本系列前面的教程中所看到的，可以在任何时候完成命令和文本插入。

然而，**Emacs** 也有它自己的编辑模式，这就是扩展它的能力或者改变一些特性工作方式的函数。这些模式通常用于编辑某种类型或类别的数据，如常规文档（使用任何印欧语言编写）、用特定的计算机编程语言（**C**、**Fortran**、**Lisp** 等等）编写的源代码、采用某种方式进行格式化（大纲、电子邮件消息、**Usenet** 文章、基于字符的示例等等）或使用标记语言（超文本标记语言（**HTML**）、**Nroff**、**TeX** 和可扩展标记语言（**XML**））的文本。甚至专门有一种模式用于编辑非文本（二进制）的数据。另外，**Emacs** 还提供了许多特殊的模式，可用于其他数据类型和系统处理，包括网络连接和网际中继交谈（**IRC**）、**Shell** 会话和 **UNIX** 文件系统自身。

这些模式可以划分为主要模式和次要模式。主要模式规定了主要的编辑行为，并且仅应用于当前编辑会话中的缓冲区。在任何时刻，每个缓冲区都有且仅有一个活动的主要模式。

尽管在任何时刻，一个缓冲区仅有一个活动的主要模式，但您可以根据需要在主要模式之间进行切换。一些专门的主要模式还提供了额外的功能和辅助（如上下文突出显示和颜色设置），当您在编辑某些类型的文档时，这是很有帮助的，但是您无需为编辑某种类型的文件或文档选择特定的模式——可在任何模式中编辑一个 **C** 程序源代码文件，正如它可在编辑 **C** 程序语言的特殊模式中完成。

次要模式通常提供了一些与任何特定的主要模式无关的特性或功能。可以把它们看作用于控制这些特性的切换：使用其函数名来调用一个次要模式可以开启或关闭该模式，任何时候您都可以开启多个次要模式。

次要模式包括 **Overwrite** 模式（本系列文章的第一部分教程对其进行了描述）、用于管理签入版本控制系统（**RCS**）的文件的 **RCS** 模式、以及处理自动文字回绕的 **Auto Fill** 模式。可以在任何时候同时激活所有这些次要模式以及许多其他类似的模式。

本教程的这一部分向您介绍成功地使用 **Emacs** 模式所需要了解的内容：

如何知道哪些模式是活动的

如何获取当前模式及其功能的描述

如何调用一个模式

您应该了解哪些模式

查看哪些模式是活动的

如本系列文章的第一个教程中所述，靠近 **Emacs** 窗口底部的突出显示栏，被称为模式行，可以告诉您当前缓冲区的所有情况——包括哪种模式当前是活动的。在模式行右边的括号里注明了当前的模式。最先列出主要模式的缩写名称，后面是任何次要模式的缩写名称。

当您启动 **Emacs** 编辑器而不打开任何文件时，则处于暂存缓冲区。在缺省情况下，这个缓冲区以 **Lisp Interaction** 模式打开，这是一个用于计算 **Lisp** 代码的特殊模式。

您可以用通常的方式启动 **Emacs**，观察模式行里写了些什么。

当您改变这些模式的时候，将在模式行里看到其反映出来了模式的改变。现在，请尝试下面的操作：按 **Ins** 键以打开 **Overwrite** 模式，并注意模式行的变化。（**Ins** 键与 **overwrite-mode** 函数进行了绑定）。

再次按 **Ins** 以关闭 **Overwrite** 模式。

当启用一个次要模式时，通常在括号里会有显示，紧接在主要模式的后面。然而，并非所有的次要模式都有这种指示器——有些次要模式是非常明了的（如 **ToolBar** 模式），它在 **Emacs** 框架的顶部显示了图形工具栏。较新版本的 **Emacs** 中其他的次要模式都非常简单（并且始终处于开

启状态),以至于如果要把它们全部显示出来,只会使显示变得非常混乱;例如,次要模式 **Unify 8859 On Encoding** 的目的是为各种 **ISO 8859** 字符集提供统一的编码方式,这样有助于进行国际化。

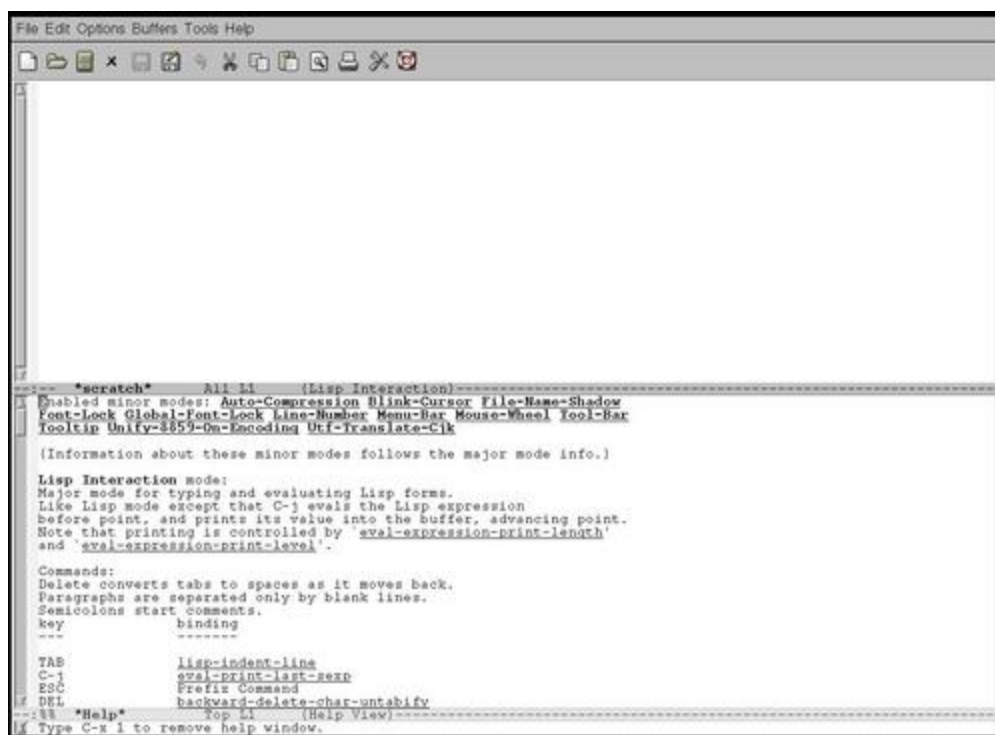
另外,一些模式提供了额外的指示器,这些指示器也会在模式行里显示出来。例如 **Line Number** 模式,通过在 **L** 后面跟上行号来表示,该行号是光标在缓冲区中的当前位置。

获得当前模式的描述

可以使用 **describe-mode** 函数,该函数与 **C-h m** 进行了绑定,用于获得当前模式的描述。当您运行这个函数时,将打开一个新的帮助缓冲区,其中列出了所有的键绑定,这些键绑定是您进行输入的缓冲区的当前主要模式所特有的,后面跟的是应用于任何处于打开状态的次要模式的绑定。

现在输入 **C-h m**,如图 1 所示。

图 1. 列出适用的当前 Emacs 模式的键绑定



正如您所看见的,用于 **Lisp Interaction** 模式的特殊绑定之一是 **Tab** 键。在这个模式中, **Tab** 并不会像您希望的那样在打字机或文字处理应用程序中将光标移动到下一个制表位处,而是按照模式描述所说的,它会对当前 **Lisp** 代码行进行缩进。因为您没有在这个缓冲区中编写任何 **Lisp** 代码,所以 **Tab** 什么都不做——您可以试试看。

缺省模式

尽管暂存缓冲区通常设置为 **Lisp Interaction** 模式,但它并不是缺省 **Emacs** 模式。要找出缺省模式是什么,可以切换到一个新的缓冲区:输入 **C-x b** 并指定 **lamb.txt** 作为缓冲区的名称。输入 **C-x 1** 关闭帮助缓冲区。

您将看到这个新的 **lamb.txt** 缓冲区有了新的主要模式,即 **Fundamental** 模式。这就是 **Emacs** 缓冲区的缺省模式。在所有 **Emacs** 模式中,这种模式是最简单明了的,拥有最少的特殊键绑定和设置。在这种模式中, **Tab** 键将按照您所希望的方式进行工作。

再次键入 **C-h m** 继续工作并获取这些模式的描述。

您将看到下面对 **Fundamental** 模式的描述:

Fundamental mode:
Major mode not specialized for anything in particular.
Other major modes are defined by comparison with this one.

再次键入 **C-x 1** 以关闭帮助窗口。

当您在新的缓冲区中打开现有文件中的内容时，**Emacs** 将基于该文件类型为您选择一种模式。如果您打开的文件包含 **C** 程序源代码，那么 **C** 模式将成为主要模式；同样，如果您打开的文件包含口语（如英语）文本，那么 **Emacs** 会将 **Text** 模式设置为主要模式。

您可以不断地改变模式，并且您还可以对所有这些设置进行配置，这样一来，就始终能够以特定的模式打开某种文件或者某种类型的缓冲区。表 1 描述了常用的 **Emacs** 模式，并给出了它们的函数名。

表 1. 常用的 **Emacs** 模式

模式	函数	类型	描述
Fundamental	fundamental-mode	主要模式	这一模式是缺省的 Emacs 模式，拥有最少设置和绑定。
Text	text-mode	主要模式	这一模式是编辑文本的基本模式。
Abbrev	abbrev-mode	次要模式	这一模式用于生成和使用缩写（请参见 Abbrev 模式）。
Auto Fill	auto-fill-mode	次要模式	这一模式用于自动文字回绕、填充较长的行和段落。
Overwrite	overwrite-mode	次要模式	这一模式用于覆盖缓冲区中任何现有的文本，而不是在当前位置插入文本。在缺省情况下，它与 Ins 键绑定。
C	c-mode	主要模式	这一模式用于编辑 C 程序源代码。
Line Number	line-number-mode	次要模式	这一模式用于显示当前行号。
Lisp Interaction	lisp-interaction	主要模式	这一模式用于编辑和编译 Lisp 代码。
Paragraph-Indent Text	paragraph-indent-text-mode	主要模式	这一模式是 Text 模式的一种特殊变体，其中的段落移动命令可用于首行缩进的段落，而不仅仅是由空行隔开的段落。
TeX	tex-mode	主要模式	这一模式用于编辑 TeX 文档。
WordStar	wordstar-mode	主要模式	这一特殊模式提供了 WordStar 编辑器的键绑定。

设置模式

Emacs 模式是一些函数。要调用其中某个函数，您需要输入 **M-x**，然后给出模式名。

现在可以尝试在新的缓冲区中调用 **Text** 模式：输入 **M-x text-mode** 并按 **Enter**。您马上可以看到模式行中的变化，其中用 **Text** 替换了 **Fundamental**。

在 **Text** 模式中进行输入

Text 模式是一种基本的模式，与 **Fundamental** 模式相比仅有很少的改变，所以它的优势并不明显；但它是对口语（如英语）文本进行编辑的一个很好的基础。从 **TeX** 模式到 **Outline** 模式，许多用于编辑某类文本或文档的特殊模式都是基于 **Text** 模式的。

尝试在新的缓冲区中输入一些文本，继续上一个教程中 **William Blake** 的主题：

Little lamb, who made thee?

Dost thou know who made thee?

Text 模式下，**Tab** 键被定义为相对于先前段落的缩进以缩进段落，如果先前的段落没有缩进，按 **Tab** 插入一个逐字制表符，移动光标到下一个制表符止。

要了解 **Text** 模式如何处理制表符，可以输入一些带制表符和空格的行：

在缓冲区的下一个新行中输入两个空格，然后输入文本 **Gave thee life, and baid thee feed**，并按 **Enter** 结束这个新的段落。

按 **Tab**，注意光标如何与前面的行（段落）进行对齐，然后输入 **By the stream and o'er the mead;**，并按 **Enter**。

输入一个不带缩进的新行：**Gave thee something of delight**，并按 **Enter**。

按 **Tab**。注意您是怎样输入多个空格的，插入了一个字母制表符。输入行 **Softest something, woolly, bright;** 并按 **Enter**。

输入 **Gave thee such a tender voice**，并按 **Enter**。

按 **Tab**。插入了另一个制表符。输入行 **Making all the vales rejoice?**

您的缓冲区现在看起来应该与图 2 所示类似。

图 2. 在 **Text** 模式中输入制表符

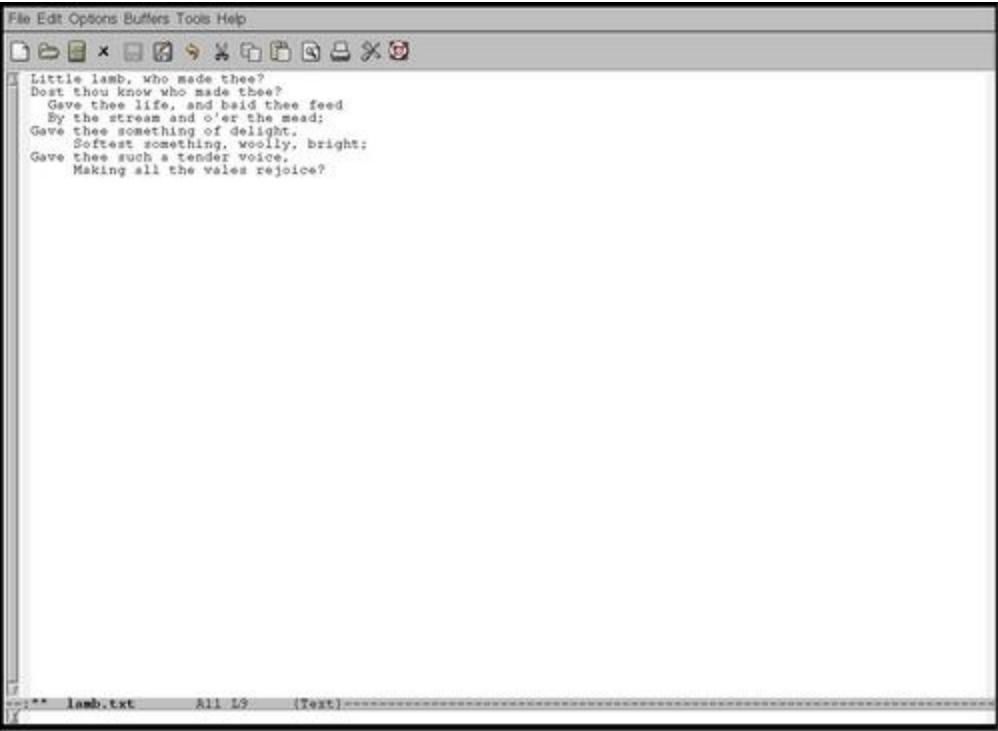


表 2 显示了 **Text** 模式所设置的键绑定。

表 2. **Text** 模式键绑定

键	描述或者函数
Esc	作为 mode-specific 命令的前缀

Esc Tab、M-Tab	ispell-complete-word
Esc S、M-S	center-paragraph
Esc s、M-s	center-line

Abbrev 模式

Emacs 的缩写 是用一个特定的字符串定义的特殊单词。当在缓冲区中输入一个缩写时（并且 **Abbrev** 模式已打开），将对这个缩写进行扩展 或者使用定义它的字符串来替换它。

Abbrev 模式（一种次要模式）使得您可以对较长的字符串或者短语进行速记，但是您可能还会想到一些其他的使用方式。

定义一个缩写

添加缩写的简单方式是，运行一个 **inverse-add** 缩写函数 **inverse-add-global-abbrev** 或者 **inverse-add-local-abbrev**。这些函数允许您为缓冲区中的某个单词定义一个缩写，第一个函数可以对当前 **Emacs** 会话中打开的任何缓冲区应用这一缩写，而第二个函数仅对与当前缓冲区具有相同的主要模式的缓冲区定义这一缩写。后者可用于定义仅适用于某些模式的缩写，例如对包含程序源代码的缓冲区中的长变量名进行定义。

尝试定义一个缩写，并使其应用于所有的缓冲区：

在下一新行中，输入一个缩写单词 **li**，以使得光标位于这个单词的末尾（即在 **i** 之后）。

通过输入 **C-x a i g**，运行 **inverse-add-global-abbrev** 函数。

在小缓冲区中给出提示的地方定义您的缩写：输入 **Little lamb** 并按 **Enter**。

请注意，您在缓冲区中输入的缩写使用定义它的字符串进行了替换，并且光标移到了这个定义的头。

现在，将光标移动到一个新行，并按您刚刚用过的方法生成一个新的缩写，通过输入 **x**（缩写不区分大小写），并使用这个字符串 **He is** 来定义它。

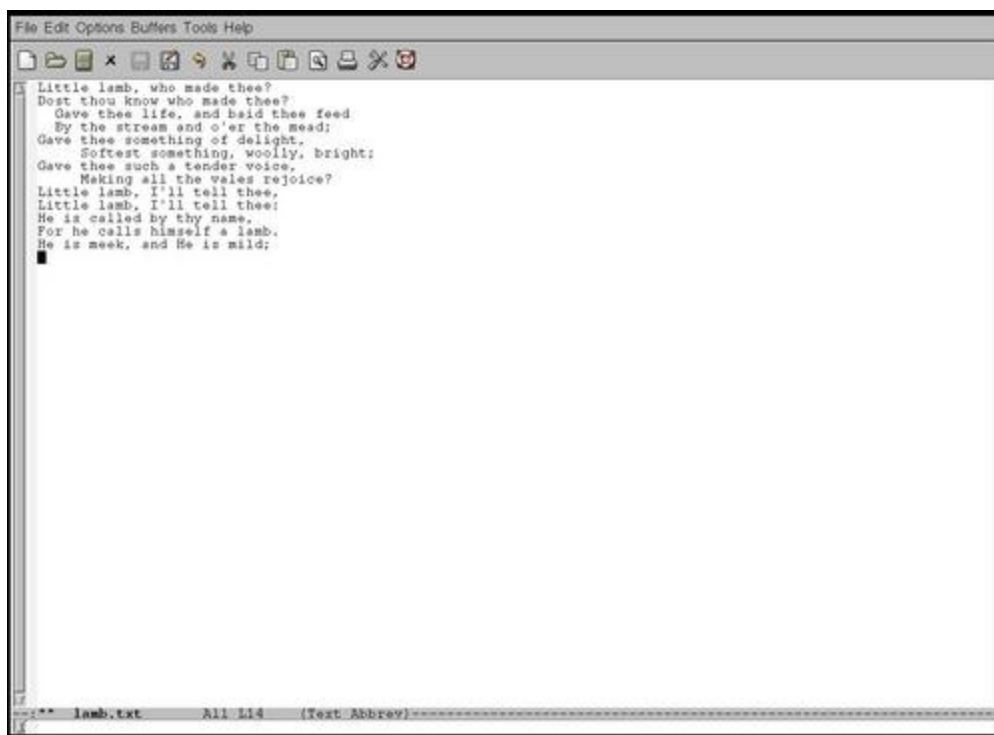
现在您已经定义了两个缩写。然而，正如您在模式行里看到的，**Abbrev** 模式是关闭的。请打开它：输入 **M-x abbrev-mode**。用您刚刚生成的定义删除这两行，然后输入清单 1 中的代码。

清单 1. 带有缩写的示例行

Li, I'll tell thee,
Li, I'll tell thee:
x called by thy name,
For he calls himself a lamb.
x meek, and x mild;

li 和 **x** 的缩写按照您的输入进行了扩展，当您完成以上操作后，缓冲区看起来就应该与图 3 所示类似。

图 3. Emacs 缓冲区中的缩写扩展



这个示例显示了如何定义在所有缓冲区中都可以使用的缩写。要定义一个缩写，以使其仅应用于当前模式的缓冲区，可以使用 **C-x a l**。

使用一个单词作为缩写的定义

您还可以为缓冲区中的单个单词定义缩写。当您在编写程序源代码并且刚输入了一个较长的变量时，这种方式是特别有用的。

要为一个单词定义缩写，可以在光标位于这个单词之后时使用 **C-x a g**。在完成了这个操作之后，当 **Abbrev** 模式开启时，小缓冲区将提示您用缩写来替代那个单词。

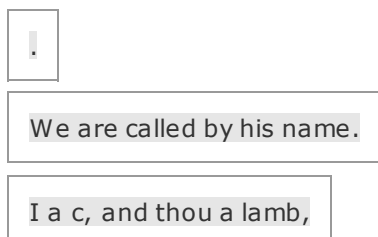
现在，请尝试下面的操作：

在新行中输入 **He became a little child**。

当光标紧跟在 **child** 这一单词之后时，输入 **C-x a g**。

在小缓冲区中输入 **c** 并按 **Enter**。

用句点结束当前行，然后输入更多的行以观察所发生的扩展：



请注意，**Emacs** 将逗号前面的字母 **c** 作为缩写，而不是单词 **called** 中的字母 **c**。

同样地，要定义一个仅应用于当前主要模式的缩写，可以使用 **C-x a l**。

删除缩写

要删除您在会话中定义的全部缩写，可以使用 **kill-all-abbrevs** 函数。

请尝试下面的操作：键入 **M-x kill-all-abbrevs**。

现在，在任何缓冲区（无论其处于何种模式）中都没有使用您所定义的缩写（**li**、**x** 和 **c**）对其进行扩展。再输入两行内容作为结束：

Little lamb, God bless thee!

Little lamb, God bless thee!

文本操作

在这一部分中，将学习一些用于编辑文本的特殊命令和键绑定，无论当前的主要模式是何种模式，其中大多数的命令和键绑定都是可用的。

缩进和填充文本

可以自动对区域按不同的方式进行缩进。通过输入 **C-x C-i** 来运行 `indent-rigidly` 函数，该函数会将区域中所有行向右缩进一个空格。

尝试下面的操作：

输入 **C-Space** 以便对缓冲区底部的标记进行设置。

将光标移到“Little lamb, I'll tell thee,”这一行的开头，以将缓冲区中最后的 10 行标记为区域。

输入 **C-x C-i** 以使这一区域缩进一个空格。

再次输入 **C-x C-i** 以使这个区域再缩进一个空格。

正如您可以多次运行 `indent-rigidly`，通过在该函数的前面加上 **C-u** 和一个数字，您还可以按指定的空格数来进行缩进，使用负数可以将区域向左移动。

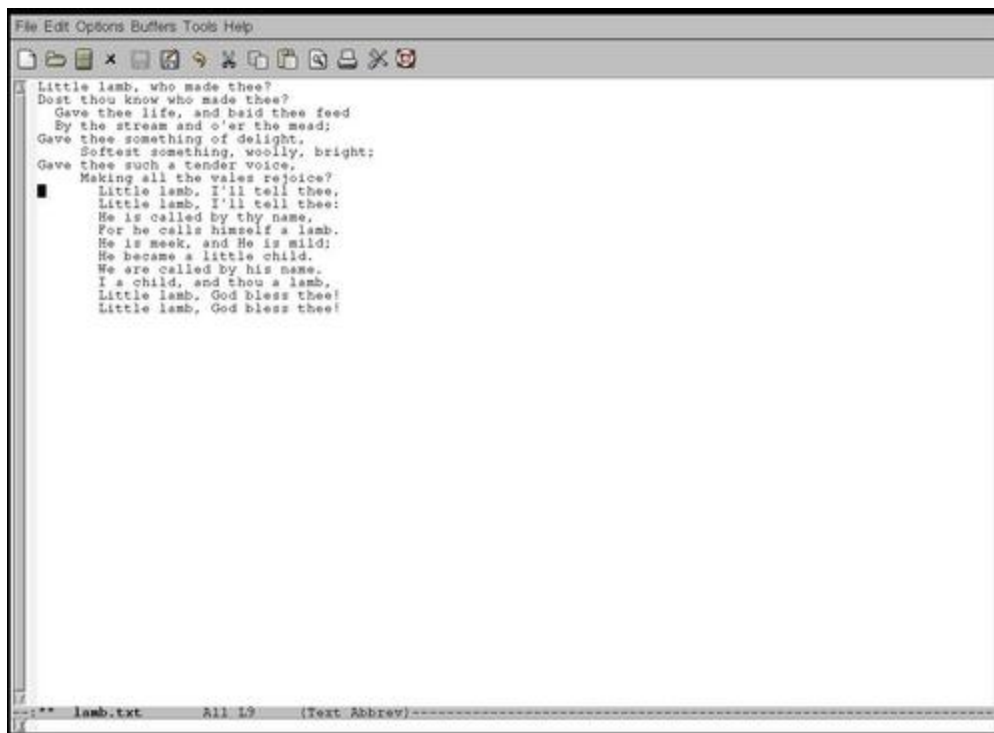
尝试下面的操作：

要再对这一区域缩进 10 个空格：输入 **C-u 10 C-x C-i**。

要使缩进回退 5 个空格：输入 **C-u -5 C-x C-i**。

在完成该操作之后，您的缓冲区应该与图 4 所示类似。

图 4. 插入严格缩进



`indent-rigidly` 函数也绑定到 **C-x Tab**。

要填充这一区域，调整右边距参差不齐的文本，并运行 `fill-region` 函数。还有一个类似的函数 `fill-paragraph`，它适用于当前段落。它与 **M-q** 的键盘绑定是相同的。

尝试下面的操作：输入 **M-x fill-region**。请注意，在填充了该区域之后，光标将移动到这个区域之后。

和任何 Emacs 格式化命令一样，可以使用 `undo` 函数对缩进和填充命令进行撤消，本系列文章的第一部分教程中对其进行了描述。现在立即尝试下面的操作以撤消区域填充：输入一次 `C-_`。

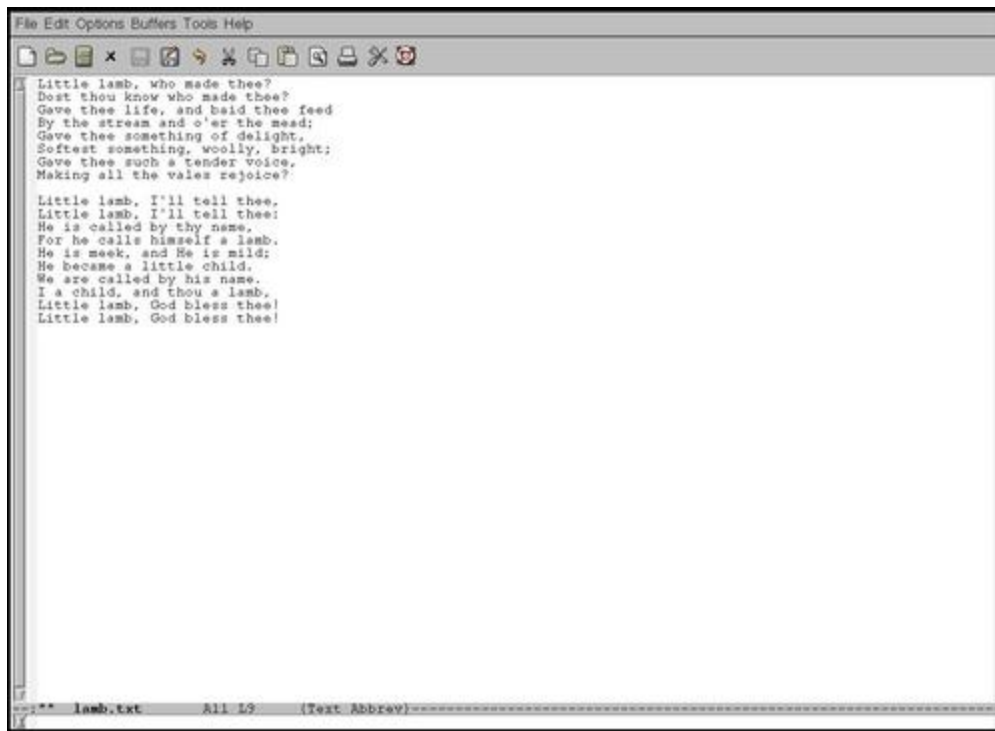
正如您可以使用缩进来添加或删除垂直的间距，您还可以去掉任何现有的水平间距。要进行这项操作，可以使用 `delete-horizontal-space` 函数，`M-`（在 `Meta` 键后面加上 `-` 键）。它将删除位于光标左边的第一个非空格字符和光标右边的第一个非空格字符之间所有的前导空格，而无论光标是否恰好位于空格处。

继续尝试下面的操作。要删除缓冲区中的所有前导空格，可以将光标移到每一行的空格处，并输入 `M-`，然后光标将移动到下一个有空格的行。

要添加一行的垂直间距，您始终可以按 `Enter`，但这样做将会移动光标。要添加垂直间距而不移动光标，可以运行 `open-line` 函数，它绑定到 `C-o`。

尝试下面的操作：将光标从缓冲区的顶端向下移动 8 行，以使其位于“Little lamb, I'll tell thee,”这一行的开始处，并输入 `C-o`。现在，您的缓冲区应该与图 5 所示类似。

图 5. 删除水平间距并插入垂直间距



文本调换

Emacs 有一些用于调换 的命令，这些命令允许您交换紧挨在光标前面的和紧挨在光标后面的字符、单词或者行。对于实现文本中的快速更正，这些命令是非常合适的。

使用 `C-t`, `transpose-chars` 函数，以调换光标前的单个字符和光标后的单个字符；使用 `M-t`, `transpose-words` 函数，以调换光标前后的单个单词。

现在，请尝试下面的操作：

将光标移动到单词 `For` 中的字母 `o` 处，然后输入 `C-t` 以调换字符 `o` 和 `F`。

再次输入 `C-t` 以调换字符 `r` 和 `F`。

将光标移回到 `F` 处，并输入一次 `C-t`，以将 `F` 回退一个字符。

将光标再次移回到 `F` 并再一次输入 `C-t`，以将 `F` 移动到这一行的开头处。

输入 `M-t` 以调换单词 `For` 和 `he`。

再次输入 `M-t` 以调换单词 `For` 和 `calls`。

将光标移动到 `he` 和 `For` 之间，并再次输入 `M-t` 以调换它们。

将光标移动到 `calls` 和 `For` 之间，并再次输入 `M-t`，以调换它们。

最后一次输入 `M-t`，以调换 `calls` 和 `he`。

transpose-lines 函数，**C-x C-t**，可以调换光标所在的行和光标前面的行。

尝试下面的操作：将光标移动到以“I a child”开头的行的开头处，并输入 **C-x C-t**。

请注意，您还可以使用通用参数在这些命令之前加上一个数值，**C-u: C-u 2 C-x C-t** 会将光标所在的行与光标前面的两行进行调换。

另一个有用的命令是 **delete-indentation**，它可以将光标所在的行与前面的行连接起来，并用空格进行分隔。它绑定到 **M-^**，通过按住 **Meta** 键并使用 **Shift** 键输入一个脱字符号 (^) 来输入该命令。另一种方法是，**C-1 M-^** 可以将当前行与紧跟其后的行连接起来。

转换大小写

有几个 **Emacs** 函数可以用于转换大小写。**uppercase-word** 函数（绑定到 **M-u**）可以将光标处到光标所在词的词尾之间的所有文本都转换为大写字母。类似地，**downcase-word**（绑定到 **M-l**）可以将光标处到光标所在词的词尾之间的所有文本转换为小写字母。

在缓冲区中尝试下面这些操作：

将光标移动到以“For he calls”开头的行，并输入 **M-c M-c M-l M-c M-l M-c** 以使这一行正确地变为大写。

向下移动光标到以“We are called”开头的行，并且输入 **M-u M-u M-u M-u M-u M-u** 以使整行内容转换为大写字母。

输入 **C-a** 以便将光标移动到这一行的开头，并输入 **C-c C-l C-l C-l C-l C-l C-c C-l** 以使这一行正确地变为大写。

通过这些命令前面加上带负参数的函数（如 **M--**），您可以对光标之前的单词进行操作，可以通过按住 **Meta** 键然后按连字符 (-) 来输入这个命令。通过使用 **downcase-region** 和 **upcase-region** 函数，您可以对区域应用改变大小写的命令，这两个函数分别绑定到 **C-x C-l** 和 **C-x C-u**。

文本操作命令汇总

表 3 列出了您刚刚学过的各种文本操作命令，并对它们的含义进行了描述。

表 3. Emacs 文本操作命令

绑定	命令或者函数	描述
C-x C-i 、 C-x Tab	indent-rigidly	这一命令对区域中的（或光标所在的）行进行缩进。
	fill-region	这一命令填充区域里的所有段落。
M-q	fill-paragraph	这一命令填充光标所在的单个段落。
M-	delete-horizontal-space	这一命令删除光标左右的任何水平间距。
C-o	open-line	这一命令以垂直间距在光标下方打开一个新行，而不移动光标。
C-t	transpose-chars	这一命令调换光标左右的单个字符。
M-t	transpose-words	这一命令调换光标左右的单个单词。
C-x C-t	transpose-lines	这一命令调换光标所在的行与光标前面的行。
M-^	delete-indentation	这一命令将光标所在的行和前面的行连接起来。以 C-1 作为开始，以连接光标所在的行和下一行。
M-u	uppercase-word	这一命令将光标处到其所在词的词尾之间的文本转换为大写字母。
M-l	downcase-word	这一命令将光标处其所在词的词尾之间的文本转换为小写字母。

C-x C-l	downcase-region	这一命令将区域中的内容转换为小写字母。
C-x C-u	upcase-region	这一命令将区域中的内容转换为大写字母。

搜索和替换文本

Emacs 中的搜索和替换命令可以在所有模式中使用，本部分将对这些命令进行描述。

增量搜索

Emacs 中最基本和最常用的搜索方式是使用增量搜索，之所以称为增量搜索，是因为当您输入第一个字符时，将立即开始搜索工作。这一方式在您输入每个字符时进行增量搜索，以搜索文本 **lamb** 为例，您输入字符 **l** 时，光标将移动到缓冲区中的下一个 **l**，接着您输入字符 **a**，光标将移动到 **la** 的第一个实例处（这与找到的第一个 **l** 的位置不一定相同），依此类推。

这是一种很有效的文本搜索方式，因为与带有搜索框工具的应用程序不同，在这些应用程序中，您需要输入一个术语，然后按 **Enter**（或者更糟，您必须单击 **OK** 按钮），而在您输入关键词的第一个字母时立即开始进行增量搜索。一旦找到正确的匹配项，您即可停止输入。

最常用的增量搜索是 **isearch-forward** 函数，它绑定到 **C-s**。它从缓冲区中光标处开始前向搜索您给出的文本。要重复搜索或移到下一个匹配项，可以再次输入 **C-s**。

当您到达了缓冲区的末尾，**Emacs** 会发出提示声，并在小缓冲区中显示一则消息，即已到达缓冲区末尾。如果此后您再次输入 **C-s**，搜索工作将回绕到缓冲区的开头（小缓冲区将给出提示信息）。

尝试下面的操作：

输入 **C-s**。请注意小缓冲区中如何提示您提供要进行搜索的字符串。

输入 **l** 并观察缓冲区中光标如何移动到下一个 **l**。如果您使用的是最新版本的 **Emacs**，那么将突出显示缓冲区中所有的 **l** 字符。

输入 **e** 并观察光标如何向前移动和突出显示下一个 **l**，以及紧跟其后的 **e** 字符。

输入一个空格并且观察光标如何移动到下一行并突出显示其中的 **le**。

输入 **C-s** 重复搜索 **le**，并观察光标如何移动到下一行。

输入 **C-s** 重复搜索 **le**，并注意 **Emacs** 如何告诉您搜索工作失败，即搜索工作已到达缓冲区末尾，而没有找到您要搜索的 **le** 的另一个匹配项。

输入 **C-s** 以便从头开始重复搜索 **le**，并观察光标如何移动到在缓冲区顶部找到的第一行。

Emacs 还提供了一种搜索缓冲区中可见文本的搜索方式：**C-s C-w** 将光标处到其所在词的词尾之间的字符串放入搜索缓冲区，而 **C-s C-y** 将光标处到其所在行的行尾之间的所有内容放入搜索缓冲区。

增量搜索通常是不区分大小写的；然而，如果您在搜索中指定搜索全部小写字母之外的任何内容，那么将仅匹配您所给出的大小写。

后向增量搜索

要从光标处开始后向搜索，可以使用 **isearch-backward**（绑定到 **C-r**），它将逆向搜索整个缓冲区。

与向前增量搜索相同，两次输入这个命令将开始搜索最近搜索过的文本。当您到达缓冲区的顶部，再次输入 **C-r** 则将回绕到缓冲区的底部。

尝试下面的操作：输入 **C-r lit** 以进行逆向搜索，然后多次输入 **C-r**，用以逆向回绕到缓冲区的底部。

非增量搜索

Emacs 也提供了非增量搜索的方式。例如，您希望搜索缓冲区中可见的某个特定短语或者字符串，那么您希望进行粘贴而不是输入，在这种情况下，非增量搜索是非常有用的。

无论是进行前向和逆向搜索，非增量搜索的工作方式都是相同的。要完成这项操作，可以在输入 **C-s** 或者 **C-r** 之后按 **Enter**，输入要搜索的完整字符串，并再次按 **Enter**。

尝试下面的操作：

输入 **C-s** 以开始前向搜索，并按 **Enter** 以指定通过非增量搜索来完成。

输入单词 **little** 并按 **Enter**。

单词搜索

有时您可能希望搜索缓冲区中的单词或短语，而不管它们是如何分隔或者如何进行格式化的，它们甚至可能位于不同的行。

例如，如果您希望在缓冲区中匹配 **feed by**，而这个短语刚好被换行符分隔开了，那又应该如何操作呢？

尝试下面的操作：输入 **C-r** 并给出 **feed by** 作为要搜索的文本。

因为这两个单词之间有一个换行符，所以 **Emacs** 很快地发出提示音并报告“**Failing I-search**”。

要对短语进行匹配，而不管单词之间存在何种分隔，可使用 **Emacs** 的单词搜索。无论使用前向或者逆向搜索都可以这样操作，按 **Enter**，输入 **C-w**，然后给出要搜索的单词或者短语。

再次尝试这一搜索：

将光标移动到缓冲区的顶端。

输入 **C-s** 以开始前向搜索。

按 **Enter**。

输入 **C-w** 以指定单词搜索。

输入 **feed by** 并按 **Enter**。

即使这个字符串跨行，也能够进行匹配，并且光标移动到这个短语第二行的 **By** 之后。

正则表达式搜索

您还可以在 **Emacs** 编辑器中搜索正则表达式。要完成这项操作，可以运行 **isearch-forward-regexp** 或者 **isearch-backward-regexp** 函数。这两个函数分别绑定到 **C-M-s** 和 **C-M-r**。然后，给出一个正则表达式作为参数。这些搜索是增量的。

尝试前向搜索，并注意随着您所构建的正则表达式的不同，其匹配项也会发生变化：

输入 **C-M-s** 以开始前向正则表达式搜索。

给出正则表达式 **l.*e**，并注意对于您输入的每个字符，其匹配结果是如何变化的。

替换文本

Emacs 中提供了几种替换文本的方式。

replace-string 函数将提示您输入进行匹配和进行替换的字符串，它将替换从光标处到缓冲区末尾的所有实例。

尝试将缓冲区中的全部 **something** 替换为 **clothing**：

输入 **M-x replace-string** 并按 **Enter**。

输入 **something** 并按 **Enter**。

输入 **clothing** 并按 **Enter**。

在该命令运行结束之后，**Emacs** 将在小缓冲区中向您报告执行了多少次替换，在这个示例中是两次。

要在整个缓冲区中删除 某个单词或者短语，可以运行这个命令，并且将其替换为空字符串。

另一个用于替换文本的功能强大的函数是 **replace-regexp**，这个函数使用正则表达式作为要搜索的字符串，并且将其替换为一个文本字符串。

最后，您可以运行 **query-replace** 函数，它会对每个需要替换的实例提出询问。它绑定到 **M-%**。表 4 对每次匹配询问的选择进行了描述。

表 4. **Emacs** 的 **query-replace** 函数的选项

键	描述
空格、y	替换这个匹配。
Del、n	跳过这一匹配到下一个匹配。
Enter、q	退出 query-replace 。

.	进行本次替换，然后退出 <code>query-replace</code> 。
,	进行本次替换，将光标移动到此处，然后退出 <code>query-replace</code> 。
C-r	指定递归编辑。
C-w	删除这个匹配并递归编辑。
C-l	重绘屏幕，并使这一行位于屏幕正中位置。
!	继续进行所有的替换，而无需再次询问。
E	编辑替换的字符串。
^	退回到前一次替换。

`replace-regexp` 和 `query-replace-regexp` 函数的工作方式是类似的，但是它们使用正则表达式作为要进行替换的字符串。

搜索和替换命令汇总

表 5 汇总了您在前面刚刚学习到的各种 Emacs 搜索和替换命令。

表 5. Emacs 搜索和替换命令

绑定	命令或者函数	描述
C-s [字符串] [C-w] [C-y]	isearch-forward	前向增量地在整个缓冲区中搜索字符串（在缺省情况下，将搜索您上一次给出的搜索字符串，如果存在），C-w 使用从光标处到光标所在单词的词尾之间的文本，以及 C-y 使用从光标处到光标所在行的行尾之间的全部内容。
C-r [字符串] [C-w] [C-y]	isearch-backward	后向增量地在整个缓冲区中搜索字符串（在缺省情况下，将搜索您上一次给出的搜索字符串，如果存在），C-w 使用从光标处到光标所在单词的词尾之间的文本，C-y 使用从光标处到光标所在行的行尾之间的全部内容。
C-s Enter C-w 单词或者短语	word-search-forward	在整个缓冲区中前向搜索给定的单词或者短语（不管它们之间如何分隔）。
C-r Enter C-w 单词或者短语	word-search-backward	在整个缓冲区中后向搜索给定的单词或者短语（不管它们之间如何分隔）。
C-M-s	isearch-forward-regexp	在整个缓冲区中前向增量搜索给定的正则表达式。
C-M-r	isearch-backward-regexp	在整个缓冲区中后向增量搜索给定的正则表达式。
	replace-string	从光标处到缓冲区末尾搜索给定的字符串，并使用给定的字符串来替换它。
	replace-regexp	从光标处到缓冲区末尾搜索给定的正则表达式，并使用给定的字符串来替换它。
M-%	query-replace	从光标处到缓冲区末尾搜索给定的字符串，对于搜索到的每个实例，询问（如表 4 中所述）是否使用给定的字符串来进行替换。

C-M-%	query-replace-regexp	从光标处到缓冲区末尾搜索给定的正则表达式，对于搜索到的每个实例，询问（如表 4 中所述）是否使用给定的字符串来进行替换。
-------	----------------------	--

使用拼写检查器

IsPELL 是一个交互式的 **UNIX** 拼写检查器，并且 **Emacs** 中内置了这个拼写检查器，对于检查缓冲区中拼写错误的单词，它的功能非常强大并且易于使用。本部分内容对各种 **ispell-** 函数进行了描述。

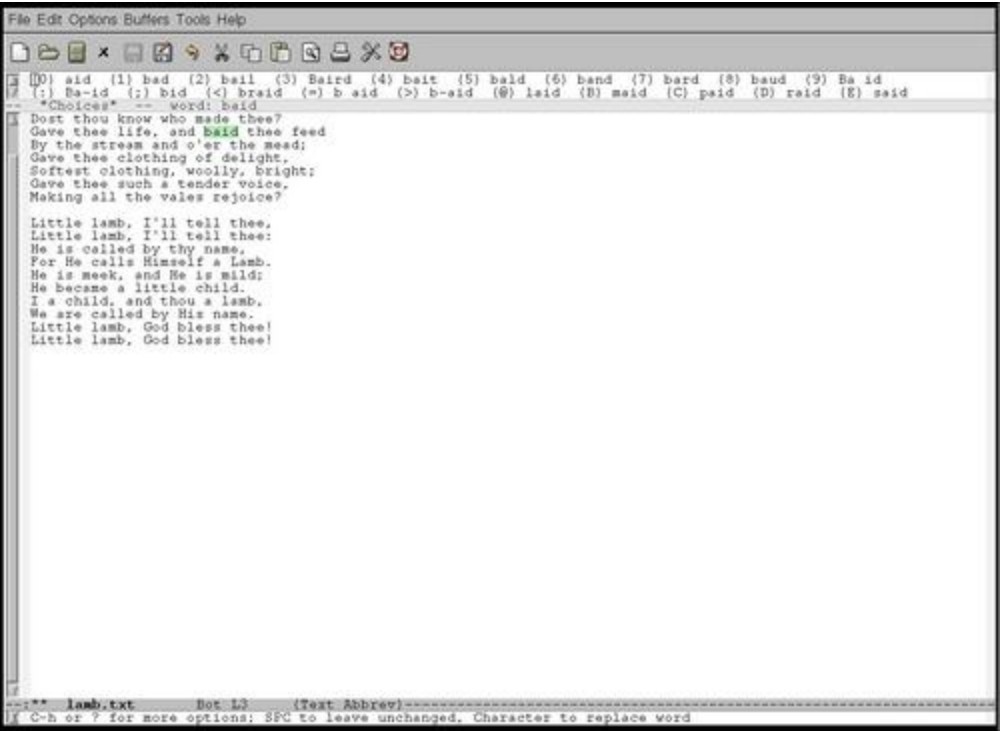
单词拼写检查

使用 **ispell-word** 函数（或者它等价的键绑定 **M-\$**）来检查光标处单词的拼写。

单词 **baid** 看起来似乎不正确。将光标移动到这个单词，并通过输入 **M-\$** 来检查这个单词的拼写。

Emacs 不认识这个单词，所以将其突出显示为拼写错误；缓冲区窗口上方将出现一个新窗口，其中给出了建议列表，如图 6 所示。每个建议的前面都有一个字符，您可以输入这个字符以使用相应的建议来替换拼写错误。（您也可以输入 **C-g** 以取消这个替换。）

图 6. 在 **Emacs** 中使用 **IsPELL** 纠正拼写



找到您所需要的单词 **bid**，然后按与之相应的键。**IsPELL** 退出，同时用您所选择的单词替换拼写错误的单词。

检查一个区域中的拼写

您还可以使用 **ispell-region** 函数检查某个区域中的拼写。

尝试使用鼠标突出显示缓冲区里的最后两行，在拖动鼠标时按住 **B1**（如本系列文章的第一个教程中所述）。然后输入 **M-x ispell-region** 以检查被突出显示的区域中文本的拼写。

IsPELL 在小缓冲区中向您报告拼写检查完成，且没有发现拼写错误。

检查缓冲区的拼写

要检查整个缓冲区里的拼写，可以使用 **ispell-buffer** 函数。

尝试进行以下操作：键入 **M-x ispell-buffer**。

对于所有的 **IsPELL** 命令，当在缓冲区的拼写检查中发现了拼写错误时，可以对其进行各种处理。表 6 对这些选项进行了描述。

表 6. `ispell` 单词替换命令

键	描述
字符	使用以（字符）开头的替换建议。
空格	在此上下文中接受这个单词作为更正。
i	接受这个单词作为更正并将其插入到个人字典文件。
a	仅对于此次 Emacs 会话接受这个单词作为更正。
A	仅对于此次 Emacs 会话中的缓冲区，接受这个单词作为更正。
r	使用您输入的字符串替换这个单词（ Ispell 会再次进行拼写检查）。
R	使用您输入的字符串替换这个单词（ Ispell 会再次进行拼写检查），并对整个缓冲区的剩余部分运行 <code>query-replace</code> 。
l	使用给定的字符串替换这个单词，并在给定的字典文件中查找这个新的字符串。
u	将这个单词的小写形式插入到个人字典文件。
m	使用给定的字符串替代这个单词，将其保存到个人字典，然后再次对该单词进行拼写检查。
C-l	使当前行位于屏幕中心位置。
C-r	进入一个递归编辑。
C-z	挂起 Emacs 。（In X 窗口系统中，这个操作将对 Emacs 客户端窗口进行图标化。）
x	退出拼写检查，并将光标移回到它的原始位置。
X	退出拼写检查，并使光标位于它当前所在的位置。
q	立刻结束拼写检查。
?	显示选项菜单。

在发生拼写错误时进行捕获

Flyspell 模式是一个特殊的次要模式，该模式可以在您输入拼写错误的单词时，突出显示这些错误。当您在编写快速文档（如电子邮件）或者第一次编写需要即刻投入使用的草案初稿时，这一模式特别有用。当 **Emacs** 发现了拼写错误时，它并不会让您停止输入，您可以继续输入，但将在缓冲区中突出显示这一拼写错误的单词。

Flyspell 模式的工作原理是在后台运行 **Ispell**；在您打开 **Flyspell** 模式后，将检查缓冲区中现有的文本，并将突出显示其中所有的拼写错误。

尝试下面的操作：

打开 **Flyspell** 模式：键入 `M-x flyspell-mode`。

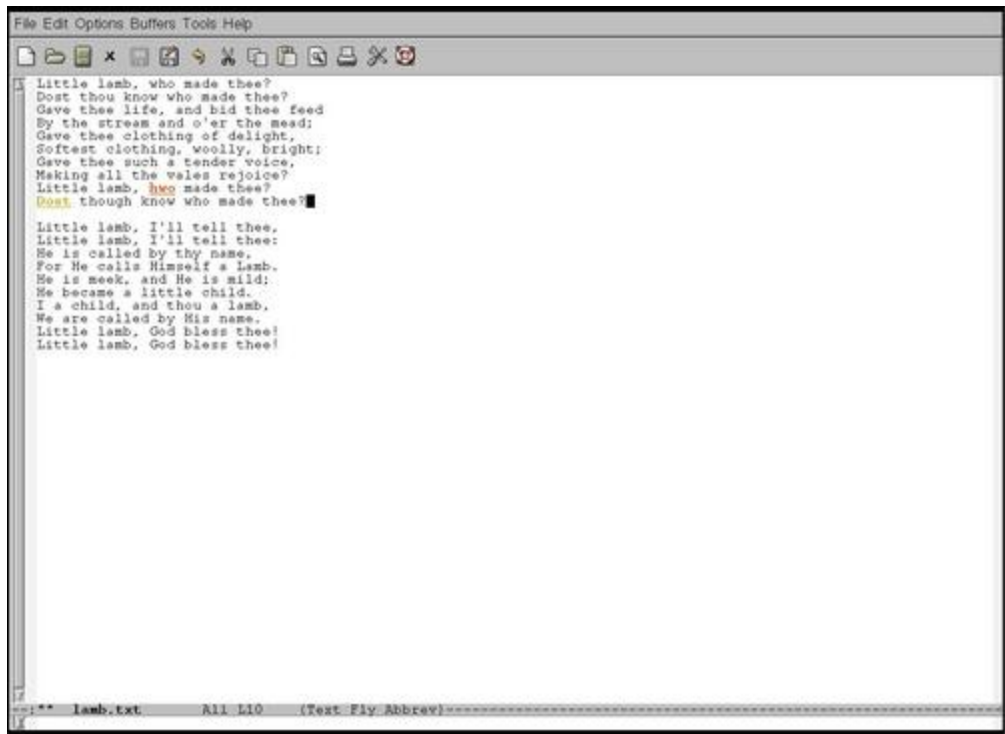
向下移动光标到第一节的末尾，输入下面的两行内容，且有意地在第一行中出现拼写错误：

```
Little lamb, hwo made thee?
```

```
Dost thou know who made thee?
```

请注意，**Emacs** 无法识别 Blake 古语 **Dost**，尽管您知道这一拼写是正确的。它和拼写错误的单词 **hwo** 都被突出显示，如图 7 所示。

图 7. 使用 Emacs 的 Flyspell 模式



要纠正这些突出显示的单词，可以使用鼠标指针在每个错误的单词上单击中键，这时会出现建议单词的菜单。对于 **hwo**，选择正确的 **who**；对于 **Dost**，选择 **Accept (buffer)** 以在该缓冲区中将此拼写作为正确的拼写而接受。

除了鼠标控制之外，Flyspell 模式还绑定了更多的命令；表 7 给出了相应的列表。

表 7. Flyspell 模式的键绑定

绑定	描述
M-\$	使用 Ispell 纠正最近一个拼写错误的单词。
M-x flyspell-auto-correct-word, M-Tab	根据 Ispell 给出的修改建议，自动地纠正最近一个拼写错误的单词。
M-x flyspell-auto-correct-previous-word	根据 Ispell 给出的修改建议，自动地纠正前面拼写错误的单词。
M-x flyspell-correct-word, B2	显示单词建议的弹出菜单。

Emacs 拼写命令汇总

表 8 汇总了您刚刚学习过的各种 Emacs 拼写函数，对其含义进行了描述，并给出缺省键绑定（如果存在）。

表 8. Emacs 的拼写命令

绑定	命令或者函数	描述
M-\$	ispell-word	调用 Ispell 以检查光标处单词的拼写。
	ispell-region	调用 Ispell 以检查某一区域中的拼写。
	ispell-buffer	调用 Ispell 以检查从光标处到缓冲区末尾的所有单词的拼写。
	flyspell-mode	在您输入内容的同时，调用 Ispell 以便在后台检查缓冲区中所有单词的拼写，

总结

总结

在完成本教程的学习之后，您对 **Emacs** 中的许多重要编辑概念有了很好的理解，并且在学习过程中，您还了解到了许多功能强大的技术。您还了解了什么是 **Emacs** 模式，以及如何使用这些模式，您了解了各种文本操作诀窍，包括进行缩进的方法、改变大小写、以及搜索和替换，您了解了如何使用 **Emacs** 拼写检查器和缩写工具。

您已开始获知 **Emacs** 的基本用法，但对于 **Emacs** 值得知道的还有很多。请注意看本系列文章的第三部分，以获得一些高级概念。

第 3 部分：高级 Emacs 文本操作

修改 **Emacs** 的命令执行

当您开始研究一些使用 **Emacs** 进行文本编辑的高级技术时，您首先需要了解如何在 **Emacs** 中改变常规命令的执行，包括那些您已经掌握的命令。接下来所介绍的技术填补了您在本系列文章的前两个教程中所了解到的内容的一些空白，并向您介绍了与改变命令（在它们运行之前、期间、之后）相关的一些技术。

指定一个数字前缀

正如本系列文章的第 1 部分教程中所描述的（请参见参考资料），通用参数可以在命令的前面使用一个数字。但是当您仅希望指定一位的数字时，还有另一种可能更加快捷的方法：使用 **digit-argument** 功能，即同时键入 **Meta** 键与一位的数字：**M-5 C-n**。

前面的命令将运行 **next-line** 功能 **C-n** 五次。

要指定一位数字，可以使用 **digit-argument**。这种方法比使用通用参数命令（在这一示例中，它将是 **C-u 5 C-n**）更加快捷，因为它比使用通用参数命令少按一个键。

现在，尝试针对本系列文章的第 1 部分教程中的练习文件（您所保存的名为 **practice.b** 的文件）应用这种方法：

使用该文件启动 **Emacs**：

```
$ emacs practice.b
```

通过执行下面的命令，将光标向下移动三行：

```
M-3 C-n
```

将光标向后移动五个字符：

```
M-- M-5 C-f
```

请注意，负数将使得给定的命令按照相反的方式执行。（您可以使用 **M-5 C-b** 得到相同的结果。）

通过输入以下的命令，移动到缓冲区尾：

M-9 M-9 M-9 C-f

请注意，您可以通过在一行中组合使用多个 **Meta** 按键，以指定更长的、多位数字，这甚至比使用通用参数更加快捷。

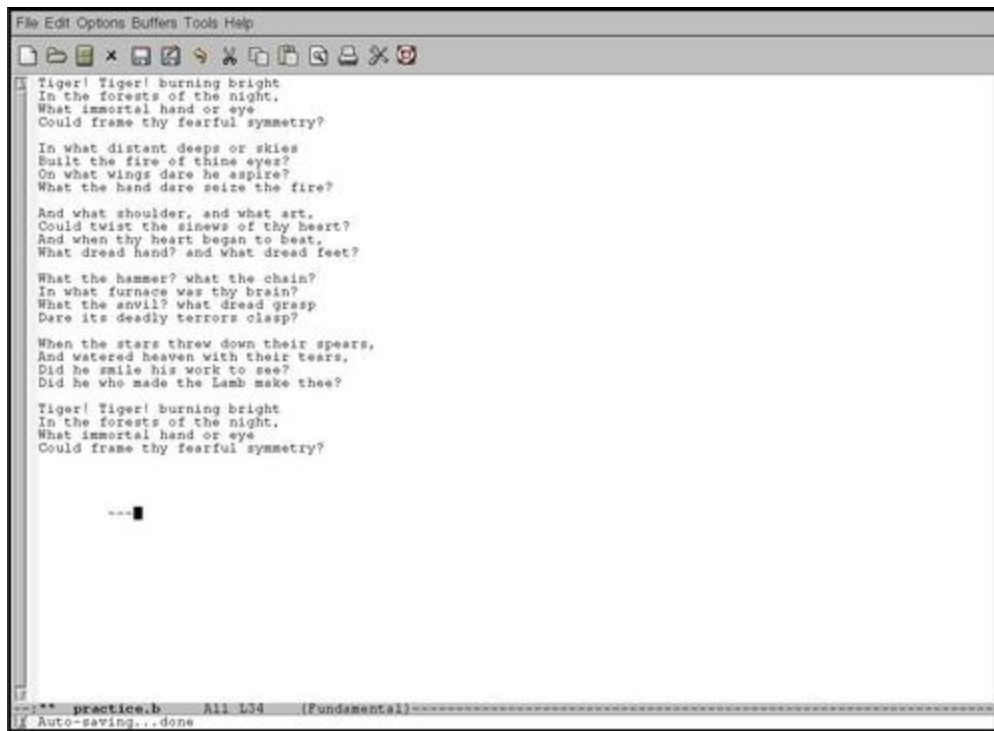
插入几个空行：键入 **M-4**，然后按 **Enter** 键。

缩进八个空格：输入 **M-8**，然后按空格键。

使用 **M-3 -** 插入三个破折号。

现在，您的缓冲区应该与图 1 中所示类似。

图 1. 在 Emacs 中使用数字参数功能进行编辑



重复上一条命令

repeat 功能对于密集型编辑工作来说是非常有用的。顾名思义，它可以重复执行您最近键入的命令。

尝试运行一个命令，并重复执行它：

键入 **C-b** 将光标向后移动一个字符。

使用 **C-x z** 重复执行一次。

您还可以反复地执行一个命令——通过再次键入 **z**，以便再次重复执行一次该命令；不断地键入 **z** 就可以反复地执行该命令。

尝试下面的操作：

键入 **z** 将光标向后移动一个字符。

多次键入 **z**，将光标向后移动多个字符。

键入 **C-f** 将光标向前移动一个字符。

键入 **C-x z** 以重复执行 **forward-char** 一次，然后键入 **z** 以重复执行该命令，直到 Emacs 发出提示声表示您已经到达缓冲区尾。

如果您在 **repeat** 命令之前使用一个数字（既可以是使用 **Meta** 键给定的数字，也可以是通过通用参数功能给定的数字），那么您就可以多次重复执行最近的命令。

通过键入 **C-b M-5 M-0 C-x z** 将光标向后移动 50 个字符，以便尝试这种方法。

执行递归编辑

您已经在本系列文章的第 2 部分教程中，即“Emacs 编辑环境，第 2 部分：学习 Emacs 的基本模式和编辑特性”（请参见参考资料），了解了各种用于搜索和替换文本的 Emacs 命令。正如在该教程中所描述的，`replace-string` 功能可以将给定的字符串替换为另一个字符串；`query-replace`（绑定到 `M-%`）可以完成相同的工作，但它在每一次替换前都会提示您，允许您判定是否应该进行给定的替换。

当您运行 `query-replace` 进行给定的替换时，可以选择使用的方法之一是递归编辑，它是一种特殊的 Emacs 功能，该功能允许您挂起当前的编辑任务，以便当该任务在后台等待的时候，您能够以交互的方式编辑缓冲区。为了说明您已经进入到递归编辑方式，使用方括号将模式行中的当前模式括起来。通过键入 `C-M-c` 退出递归编辑，即 `exit-recursive-edit` 功能；当您执行该操作时，将从模式行中删除方括号，并且您退回到原始编辑任务或者进入递归编辑之前的上下文。

您可以通过运行 `recursive-edit` 功能，在任何时候进入递归编辑。顾名思义，您可以嵌套进行任何次数的递归编辑；您每次在进行递归编辑时，都会对模式行中的当前模式加上一对新的括号。

在使用 `query-replace` 的情况下，您在询问提示符处键入 `C-r`，以进入递归编辑。这允许您在进行特定的替换之前，停止您将要进行的替换，以便您能够编辑缓冲区；当您退出递归编辑时，您将会返回到该替换操作。

现在，请尝试下面的操作：

键入 `M-%` 以运行 `query-replace` 功能。

回答迷你缓冲区中的询问：

Query replace: o

Query replace o with: a

在第一个替换的位置，当受到询问时，可以通过键入 `C-r` 进行递归编辑：

询问是否使用 `a` 替换 `o`：（? 用于帮助）`C-r`

通过键入 `M->` 移动到缓冲区尾，然后在三个破折号后键入 `William Bloke`。

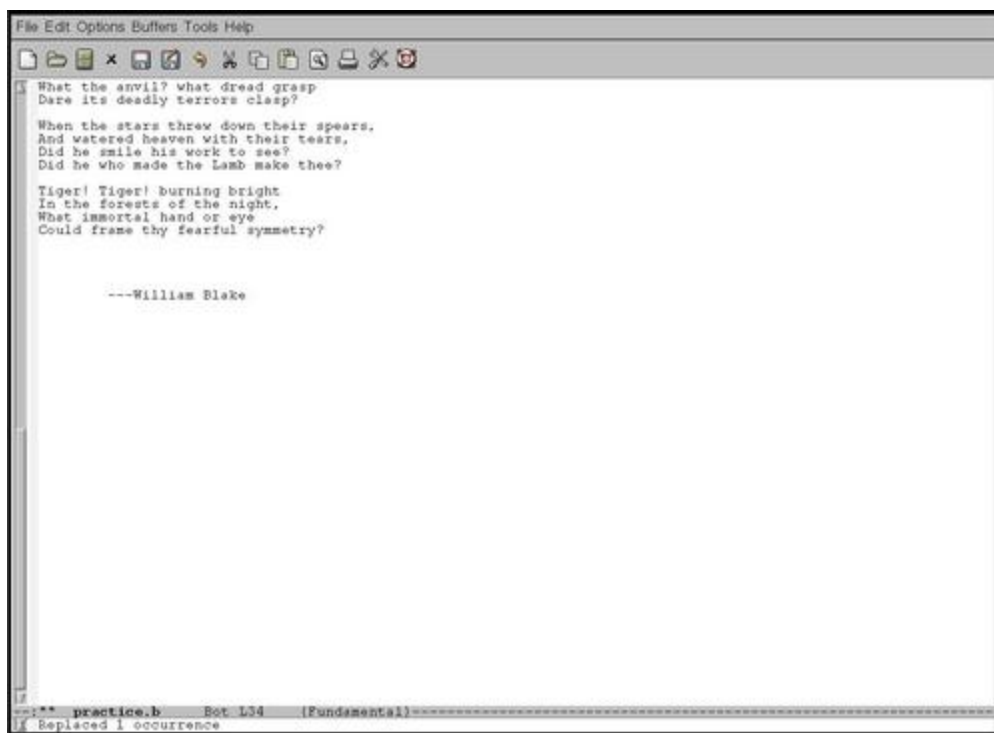
键入 `C-M-c` 以退出递归编辑。

键入 `n` 以拒绝对第一个匹配项进行替换。

所找到的第二个匹配项是 `Bloke` 中的 `o`，该单词是您在 `query-replace` 功能运行之后键入的；键入 `y` 以替换它。

当退出这个功能时，您的缓冲区的最后一部分内容应该与图 2 中所示类似，且 Emacs 应该在迷你缓冲区中报告，它已经替换了给定字符串的一个匹配项。

图 2. 在 Emacs 询问替换期间，进行递归编辑之后



矩形区块

您是否曾经希望能够从文档中选择文本区块，以对其进行复制、剪切或者粘贴？您的确可以这样做。在 **Emacs** 中，我们将通过文本的四个角中任何两个相对的角所指定的文本选择称为矩形区块；接下来的部分将向您介绍如何有效地使用矩形区块。

标记一个矩形区块

要指定一个矩形区块，您只需要对矩形区块的四个假想的角中的任何一个设置标记，然后将光标移动到与该角相对的角。当您完成以上操作时，选定的两个角所包围的虚构文本块就是当前矩形区块。

当您运行下面的矩形区块命令之一时，将针对您所选择的文本执行相应的操作。**Emacs** 所有的矩形区块命令都是以 **C-x r** 开头的，本文在接下来的部分中对它们进行了说明。

删除一个矩形区块

有几种方法可以删除标注为矩形区块的文本。尽管这里描述的所有功能都适用于删除当前矩形区块，但其中每一个都具有不同的效果。

剪切一个矩形区块

使用 **kill-rectangle** 功能，**C-x r k**，可以剪切当前矩形区块。它与您所熟悉的常规 **kill-line** 功能相似：它删除该矩形区块中的所有字符，并且不用任何其他字符来替换它们。位于该矩形区块右边的字符将自动地向左移动。

这一功能可以将矩形区块的内容保存到某个特殊的矩形区块剪切区域中，而不是保存到一般的剪切环中（请参见使用剪切环）。

现在，请尝试下面的操作：

将光标移动到以 **What the hammer** 开始的节的开头，并键入 **C-space** 以设置标记。

移动光标，使其恰好位于单词 **terrors** 之前，以指定一个包含四行的矩形区块（顺便提一下，该区块中每行包括三个单词）。

剪切您选择的矩形区块：键入 **C-x r k**。

您所选择的矩形区块已经不存在了。请注意，该矩形区块右边的所有文本是如何进行移动，以填充该矩形区块所在位置的。

删除一个矩形区块

要删除 您所标记的矩形区块中的字符，而不保存它们，可以使用 **delete-rectangle**，它绑定到了 **C-x r d**。这一命令用于删除矩形区块的区域，因此其效果看起来与剪切的矩形区块的效果相同（正如刚刚描述的）；它们的区别在于，在执行该命令后，您无法再粘贴回所剪切的文本。

（然而，正如您在第一部分教程中所了解到的，您始终可以撤消 刚执行的操作。如果您在刚刚删除或者剪切了一个矩形区块之后键入 **C-_**，那么将恢复该矩形区块的原始文本。）

清除一个矩形区块

刚介绍的这两个用于删除矩形区块的命令都会删除该矩形区块所占据的整个空间。但是，您还可以清除 它，通过运行 **clear-rectangle** 功能，用空格字符替换整个区域，该功能与 **C-x r c** 进行了绑定。

尝试下面的操作：

移动光标，使其恰好位于行 **On what wings dare he aspire?** 中的单词 **dare** 之前，并通过键入 **C-space** 来设置标记。

要指定由两行中的单词 **dare** 组成的小的矩形区块，可以移动光标，使其恰好位于下面一行的单词 **dare** 之后。

键入 **C-x r c** 以清除您刚刚定义的矩形区块。

请注意，光标移动到该矩形区块的左下角。

打开一个矩形区块

最后，您可以打开 一个矩形区块，使用该矩形区块以指定一个将要添加空格的区域。要完成这项操作，可以标记一个矩形区块，然后运行 **open-rectangle** 功能，**C-x r o**。执行以上的操作，将使用空格字符填充整个矩形区块，并将原矩形区块中的所有文本向右移动。

现在，请尝试下面的操作：

将光标移动到缓冲区顶部，并通过键入 **C-space** 设置标记。

移动光标，使其恰好位于底部节中 **Could** 中的 **Co** 的后面。

通过键入 **C-x r o** 打开该矩形区块。

粘贴一个矩形区块

要在光标处粘贴上一次剪切的矩形区块的内容，可以运行 **yank-rectangle**，该操作与 **C-x r y** 进行了绑定。这一命令用于在光标处插入上一次剪切的矩形区块。在执行插入操作时，将该矩形区块中所有行的所有现有文本移动到右边。

尝试下面的操作：

将光标移动到文本中包含 **what the chain** 的节的开头：键入 **M-< M-1 M-5 C-n M-2 C-f**。

键入 **C-x r y** 以粘贴该矩形区块。

请注意，您刚刚清除的小矩形区块并不是被粘贴的那个区块；当您清除 一个矩形区块时，并没有对其进行保存，仅保存那些剪切 的矩形区块，并且您只能够粘贴上一次剪切的矩形区块。

另外请注意，光标移动到该矩形区块右下角的后面，并且迷你缓冲区报告已经设置了标记。当您粘贴一个矩形区块时，标记设置到您所粘贴的矩形区块的左上角。

要使用空白字符替换一个矩形区块，但需要保存它的原始内容，以便您稍后可以对其进行粘贴，那么可以首先剪切它，再粘贴它，然后清除它：

通过键入 **M-< M-2 M-2 C-n M-2 C-f C-space**，在倒数第二节中的第一个 **Did he** 之前设置标记。

键入 **M-6 C-f C-n** 以移动光标，使其恰好位于下一行中的 **he** 之后。

键入 **C-x r k** 以剪切矩形区块。

键入 **C-p** 将光标移动到您刚刚剪切的矩形区块的左上角。

使用 **C-x r y** 粘贴它。

使用 **C-x r c** 清除它。

此时，您可以将这个矩形区块粘贴到其他地方。矩形区块的原始位置现在已经填满了空白字符。

尝试下面的操作：

通过键入 **M-1 M-6 C-p M-1 M-4 C-f** 移动光标，使其恰好位于单词 **wings** 之后。

使用 **C-x r y** 粘贴该矩形区块。

请注意，光标移动到您刚刚粘贴的矩形区块的右下角。已经将标记设置到该矩形区块的左上角，正如迷你缓冲区中所报告的。

矩形区块的命令表

表 1 列出了您刚刚了解的各种 **Emacs** 矩形区块命令，给出它们的功能名称，并描述它们的含义。

表 1. 使用矩形区块的 **Emacs** 命令

键盘输入	功能	描述
C-space	set-mark-command	标记矩形区块的一个角（光标标记其相对的角）。
C-x r k	kill-rectangle	剪切当前的矩形区块，并将其保存在一个特殊的矩形区块缓冲区中。
C-x r d	delete-rectangle	删除当前的矩形区块，并不为粘贴而保存它。
C-x r c	clear-rectangle	清除当前的矩形区块，使用空白字符替换整个区域。
C-x r o	open-rectangle	打开当前的矩形区块，使用空白字符填充整个区域，并将该矩形区块的所有文本移动到右边。
C-x r y	yank-rectangle	在光标处，粘贴上一次剪切的矩形区块的内容，将所有的现有文本移动到右边。

操作标记

设置标记和进行选择都是非常重要的，在本系列文章的第 1 部分教程（请参见参考资料）中您了解了相关的 **Emacs** 概念。但是需要了解的内容远不止这些，本部分将描述与这些主题相关的一些高级编辑技术。

移动到标记处

您知道 **C-space**, **set-mark-command** 功能，可以在光标处设置标记。但是当您继续编辑时，确切地了解在缓冲区中的哪个位置设置了标记，这有时是很有用的，而且同样可以很容易地回到标记处，如果您在光标处设置它，并且希望出于某种原因而引用它。**Emacs** 目前还无法对设置标记的位置进行可视化表示，但是有一些方法可以将光标移动到标记处，这就可以解决前面的两个问题。

您可以通过几种不同的方法来完成这一任务。其中一种方法是，在 **C-space** 之前使用通用参数 **C-u** 命令。使用这种方法，可以更改 **C-space** 的含义，不再设置标记，而是移动到标记处。

在您的练习缓冲区中，您所执行的最后一项操作是粘贴一个矩形区块（请参见前面的部分），所以光标移动到该矩形区块的右下角，并且标记设置于其左上角。现在通过输入以下命令，尝试移动到标记处：**C-u C-space**。

这是一种很好的、快速的方法，但使用起来却并不是很方便，尤其是如果在您移动到标记处之后，希望返回到光标原来所在的位置，但这个位置已经丢失了。但是还有另一种方法可以移动到标记处，即记住光标的位置，对于上述的情况，这种方法是很有用的。可以使用 **exchange-point-and-mark** 功能，它与 **C-x C-x** 进行了绑定。它的工作方式正如其名称所表示的：它将光标移动到标记处，但是同时，它在光标处设置一个新的标记。

现在，请尝试下面的操作：

键入 **M-< C-space** 以在缓冲区的顶部设置标记。

键入 **M-5 C-n** 以从标记处移开。

交换光标和标记：键入 **C-x C-x**。

要移回到光标的最近位置——并且在缓冲区的顶部设置标记——再次运行该命令：键入 **C-x C-x**。

再次将标记设置到缓冲区的顶部。如果您再次运行这个命令，而不设置新的标记，那么无论您将光标移动到了什么位置，都将返回到缓冲区的顶部。通过移动光标，然后再次运行这一命令，以尝试这种方法：键入 **M-9 C-f C-x C-x**。

通过两次键入该命令，您可以将这个命令作为一种快速的、功能强大的方式来使用，以“查看”标记设置的位置：第一次，您键入 **C-x C-x**，将光标移动到标记处，但同时还在光标处设置了标记。第二次，您键入 **C-x C-x**，将标记设置回它原来所在的位置，并且将光标移动到开始所处的位置。

标记缓冲区中特殊的部分

Emacs 提供了一些特殊的功能，用于标记文本单元的某些类型。这些内容值得深入研究，并且在这里对它们进行了相应的描述。

可以在所有这些标记命令之前加上数字，以修改它们进行标记的单元数目以及它们的位置，既可以在当前光标的右边（正数），也可以在其左边（负数）。

标记单词

mark-word 功能用于标记从光标处到当前词尾，并且它与 **M-@** 进行了绑定（按住 **Meta** 键，同时键入 **@** 字符）。

对于快速地剪切一个单词来说，它并不是很有价值（您仍然应该使用 **M-d** 来完成这一任务，正如您在前面的教程中所了解到的），但对于对单个单词进行标记，以对其执行某些其他操作来说，这是一个很好的命令，如使用 **upcase-region** 和 **downcase-region** 功能进行大小写转换，它们分别将区域中的所有文本转换为大写字符和小写字符。

尝试下面的操作：

移动到缓冲区中的最后一行，恰好处于 **W** 的前面：键入 **M-> M-b M-b**。

键入 **M-2 M-@** 对光标右边的两个单词进行标记。（如果您没有移动到这些单词的左边，您可以使用 **M-- M-2 M-@** 实现相同的目的。）

运行下面的功能，将区域中所有的字符转换为大写字母：键入 **M-x upcase-region** 并按 **Enter**。

请注意，这时打开了一个新的窗口，它告诉您禁用了 **upcase-region** 功能；这是因为它可能让新用户感到不知所措。现在您已经掌握了这个内容，因此键入 **y** 以便为这一会话启用它。然后，如果您愿意，可以在提示是否同时希望为将来的会话启用它时，再次键入 **y**。

标记段落

mark-paragraph 功能，**M-h**，将当前段落标记为区域。当您键入这个命令时，光标可以位于该段落中的任何位置。这个功能在段落尾设置标记，并且移动光标，使其恰好位于该段落的前面。

尝试标记一个段落：

使用 **M-5 C-p** 移动到最后一节，该命令将光标移动到最后一行中的倒数第二个单词。

键入 **M-h** 以选择这一段。

请注意，光标移动到该节前面的空白行。标记设置到该节前面的行，键入 **C-x C-x** 以验证它。

标记整个缓冲区

要立刻为整个缓冲区设置标记，可以运行 **mark-whole-buffer** 功能，**C-x h**。

现在，请尝试下面的操作：**C-x h C-w C-y** 将整个缓冲区标记为一个区域，剪切该区域（因而剪切整个缓冲区），然后粘贴它。

对于完整地将一个现有的缓冲区复制到另一个缓冲区，这是一种很好的方法。通过将缓冲区标记为区域，然后键入 **M-w** 以运行 **kill-ring-save** 功能，可以完成这一任务。然后，您可以将它粘贴到其他地方，甚至粘贴到另一个缓冲区中。

使用标记环

您已经了解了，键入 **C-u C-space** 可以返回到最近设置的标记（请参见“移动到标记处”）。但是 **Emacs** 并不仅仅只能够记住最后一个标记，它可以记住您最近设置的 16 个标记，并且将它们保存在一个称为标记环的特殊位置。

当您多次键入 **C-u C-space** 时，**Emacs** 将遍历标记环中的这 16 个位置。在您键入该命令 16 次之后，**Emacs** 将返回到该标记环中的第一个标记处。

每个缓冲区都具有自己的标记环，标记环用于保存特定缓冲区中最近设置的 16 个标记。

尝试多次键入 **C-u C-space** 以遍历在您的练习缓冲区中最近设置的一些标记。

设置临时标记

只要在缓冲区中的某处设置了一个标记，就存在一个区域（正如您所了解的，当前区域 是位于标记和当前光标位置之间的区域），但该区域是不可视的，因此您通常无法看到它。查看区域的一种方法是，当您第一次通过拖动鼠标选择它时，仅持续到您按某个键之前。

但是通过运行 **transient-mark-mode** 功能，您可以使得这个区域成为可见的，该功能可以进行切换。在设置了这一辅助模式后，无论您何时设置标记，都会突出显示该区域，并且当您调整它的时候，如通过使用光标移动键，它仍然保持突出显示的状态。一旦您键入更改缓冲区的另一个键，而且无需与该区域有关，比如当您键入一个常规字母数字字符，以将其插入到光标处（或者如果您键入 **C-g**）时，突出显示将会消失，并且清除该区域，尽管标记仍然存在。

如果您再次设置标记，或者如果您键入一个与该区域有关的命令（如交换光标和标记），那么将设置该区域，并且它在缓冲区中再次突出显示。

现在，请尝试下面的操作：

键入 **M-x transient-mark-mode** 以启用这一模式。

移动到缓冲区尾，并设置标记：键入 **M-> C-space**。

通过键入 **C-p** 几次，向上移动光标，并观察标记的移动情况。

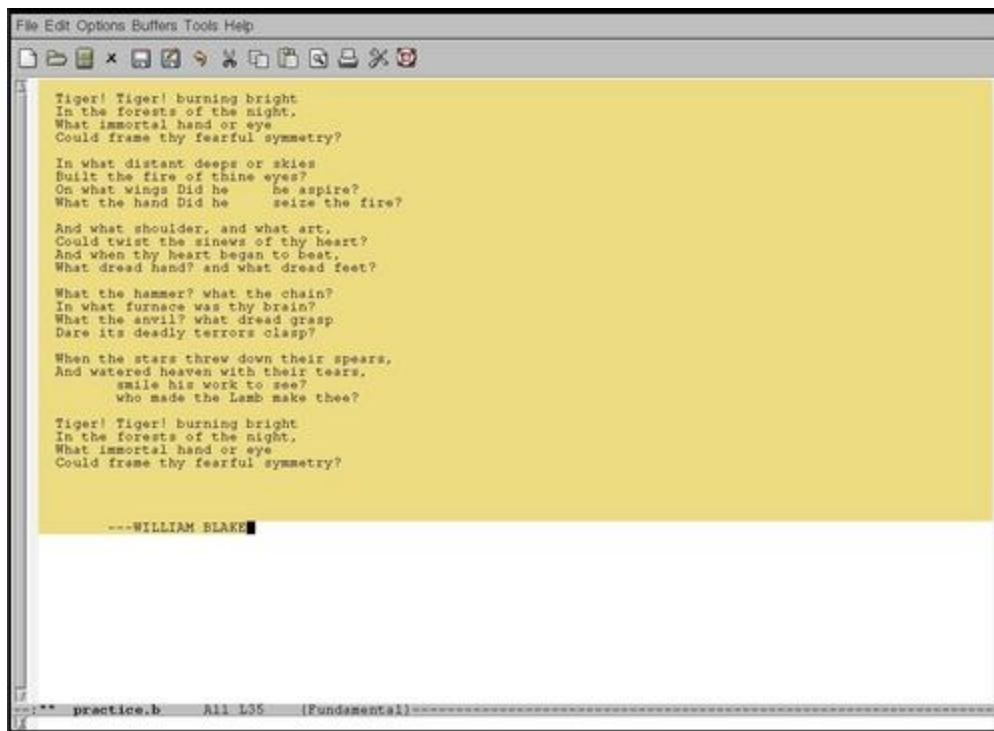
移动到缓冲区的顶部，并插入一个空格：键入 **M-< C-o**。请注意，该区域不再突出显示。

键入 **C-w** 以剪切这一区域，并且注意，没有剪切任何内容，相反地，**Emacs** 发出提示声，并且在迷你缓冲区中向您报告“**The mark is not active now**”。

但是仍然设置了一个标记，键入 **C-x C-x** 以证明这一点，并再次观察如何快速地突出显示该区域。

您的 **Emacs** 会话应该与图 3 中所示类似。

图 3. 临时标记模式中的可见区域



请注意，这是一个辅助模式，当激活它时，在模式行中不显示。键入 **M-x transient-mark-mode** 以禁用它。

使用二次选择

回到本系列文章的第 1 部分教程中，您学习了如何设置标记以及操作区域，同时您还学习了如何使用鼠标来进行选择。但是当您在使用 X 时，还有另一种方法，即使用鼠标进行选择 and 剪切文本，而无需使用标记和光标，或者甚至根本无需设置标记或移动光标。这是通过使用二次选择 来实现的，二次选择是所选文本的一个连续区域（没有通过标记和当前光标来表示其结尾）。当您使用鼠标进行选择时，可以通过按下 Meta 键进行二次选择。

您将注意到，二次选择与普通的 X 选择相比，使用了不同的颜色进行突出显示。

现在尝试进行二次选择：

按住 Meta 键。

在第一节的开始处按下鼠标的 左键，然后将它拖曳到末尾，然后放开 Meta 键和鼠标按钮。

当您进行该操作时，并没有将光标从所处的位置移开。

不使用鼠标拖曳，也可以进行二次选择。要完成这一任务，按住 Meta 键，在开始处单击鼠标的左键，在末尾处单击鼠标的右键，然后释放 Meta 键。

有一些用于选择单词和行（作为二次选择）的其他快捷方式，它们类似与用于进行常规选择的鼠标操作：当鼠标指针位于一个单词上，以选择整个单词时，按住 Meta 键，然后双击鼠标左键（M-B1-B1）；当您三击一行中的某处（M-B1-B1-B1）以选择整行时，按住 Meta 键。

现在，请尝试下面的操作：按住 Meta 键，然后双击缓冲区中出现的单词 Tiger 中的任何一个。请注意，同样的，光标并没有移动。

要对二次选择进行粘贴，请按住 Meta 键，然后在希望将二次选择粘贴到缓冲区的目标位置单击 鼠标中键。当您在进行该操作时，光标移动到您所粘贴的选择内容的末尾。您还可以将它粘贴到另一个 X 客户端窗口。

通过单击 WILLIAM 和 BLAKE 之间的 鼠标左键，按空格键以插入一个额外的空格；然后按住 Meta 键，并单击 鼠标右键 以便将二次选择粘贴到此处，以尝试这项操作。

标记和选择命令表

表 2 为刚介绍的高级标记和选择命令提供了汇总信息。表中给出这些命令的功能名（在适用的情况下）。

表 2. 高级 Emacs 标记和选择命令

键盘输入	功能	描述
C-u C-space		移动到标记环中的前一个标记。
C-x C-x	exchange-point-and-mark	交换光标和标记的位置。
M-@	mark-word	标记从光标到当前词尾的所有文本。
M-h	mark-paragraph	标记当前段落，不管光标处在什么位置。
	transient-mark-mode	切换临时标记模式。
	mark-whole-buffer	标记整个缓冲区，不管光标处在什么位置。
M-B1		设置二次选择的开始，拖曳鼠标以进行二次选择。
M-B3		设置二次选择的结尾。
M-B1-B1		将一个单词标记为二次选择。
M-B1-B1-B1		将一行标记为二次选择。

高级剪切和粘贴命令

您已经了解了有关剪切和粘贴文本的内容，这在本系列文章的第 1 部分教程中已进行了描述（请参见参考资料），但对于 Emacs 来说，并不仅限于这些基本的操作。本部分将描述一些高级技术。

高级剪切命令

您已经了解，**C-k** 可以剪切从光标到行尾的全部内容，但是还有一些其他有趣的剪切命令，可用于缓冲区中的其他部分。

剪切超过当前行的内容

如果您在 **C-k** 之前加上一个数字参数，既可以使用 **Meta** 键（正如在“指定一个数字前缀”中所描述的），也可以使用通用参数，那么您就可以剪切多行内容。

如果您在 **C-k** 之前使用 **0** 作为参数，那么您将剪切从光标到该行开始处的所有文本。

尝试剪切当前行中位于光标左边的全部内容：键入 **M-0 C-k**。

现在，当您在进行粘贴时，将重新得到该行原来的内容。尝试下面的操作：**C-y**。

您还可以进行向后操作。以当前行中光标之前的内容作为开始，负的参数将向后剪切多行内容。例如，**C-u -5 C-k** 剪切从光标到该行开头以及它前面的五行内容。

剪切句子

使用 **kill-sentence** 功能，**M-k**，可以剪切从光标到当前句尾的所有文本。

尝试下面的操作：

移动光标到以 **In what distant** 开头的行：键入 **M-< M-6 C-n M-2 C-f**。

键入 **M-k** 以剪切该句子。

与 **C-k** 一样，这个命令从光标处开始剪切操作，并且它还可以接受一个数字参数。所以，**M-0 M-k** 将剪切从光标处到句子开始的全部内容。

删除到一个字符

zap-to-char 功能，与 **M-z** 进行了绑定，对于删除一定范围的字符（从光标直到一个特定的字符，包括该字符），该功能是很有用的。通过键入 **M-z**，然后给出要删除到的字符，以便执行该命令。

现在，请尝试下面的操作：

键入 **M-z**。

键入 **D** 以删除自光标到第一个 **D** 字符（包括该字符）之间的全部内容。

与本文中介绍的在前面加上数字参数的其他命令一样，这个功能将删除直到给定字符的出现次数恰好等于数字参数的所有文本，既可以从光标处向前（正整数），也可以向后（负整数）删除。

尝试删除从光标到其后的第五个 **e** 字符间的全部内容：键入 **M-- M-5 M-z e**。

使用剪切环

正如存在标记环以保存所设置的标记（正如使用标记环中所描述的）一样，还存在一个剪切环，用于保存每次您进行剪切操作时所剪切的文本。每次在剪切某些文本时，都将会在剪切环的一个槽位中保存它。仅有的例外情况是矩形区块，最后剪切的矩形区块保存在其本身的特殊位置，而不在剪切环中（请参见矩形区块）。在缺省情况下，每个缓冲区的剪切环都有 30 个槽位。

当您键入 **C-y** 以粘贴您上一次剪切的文本时，事实上您是从剪切环的顶部进行粘贴。要在剪切环中回退一个槽位，可以键入 **M-y**。在进行这个操作时，将使用剪切环中前面的文本替换您刚刚粘贴的文本。每当您再次键入 **M-y**，您就会在剪切环中回退一个槽位。

您还可以在一般的 **C-y** 前面加上一个数字参数，以粘贴剪切环中特定位置的文本，在剪切环中，最近的内容编号最小，所以 **C-u 1 C-y** 与 **M-1 C-y** 是相同的，这与简单的 **C-y** 是相同的。

此时，尝试粘贴该剪切环中的第三个条目：键入 **M-3 C-y**。

但是 **C-u C-y**（不带数字参数），它具有特殊的含义。通常，**C-y** 在光标处粘贴位于剪切环顶部条目的文本，然后光标移动到被粘贴的文本末尾。但是使用 **C-u C-y**，光标将仍然位于粘贴文本的开头处。对于编辑您将要粘贴的文本，这是一种很好的方法。

现在，通过在该行的开头粘贴文本来尝试这一操作：键入 **C-a C-u C-y**。

通过观察 **kill-ring** 变量，您可以查看剪切环中的内容。尝试下面的操作：键入 **C-h v kill-ring**。这个命令打开了一个新的窗口，其中显示了剪切环中的内容。键入 **C-x 1** 以关闭该窗口。

剪切和粘贴命令表

表 3 对这一部分中描述的高级剪切和粘贴命令提供了汇总信息。

表 3. 高级 Emacs 剪切和粘贴命令

键盘输入	功能	描述
integer C-k	kill-line	剪切行的整数 数目。如果是 0，剪切从光标到该行开头的全部内容；如果是负数，则反向剪切。
M-k	kill-sentence	剪切从光标到句子的结尾处的内容。
M-z	zap-to-char	删除从光标到指定的字符之间的所有文本。
M-y	yank-pop	移动到剪切环中的下一个槽位。
integer C-y	yank	粘贴剪切环中指定槽位的内容。

总结

关于使用 Emacs 的系列文章的第 3 部分教程到此就结束了。到现在为止，您可能已经认识到，Emacs 提供了各种各样的特性和功能（甚至有些特性和功能是您在任何其他编辑器或者应用程序中可能从未听说过的），它也因此赢得了“编辑器之王”的美名。

尽管还有更多的 Emacs 概念和特性需要了解，但您已经掌握了 Emacs 的一些更高级的文本操作方法，您应该可以成为一位充满信心的 Emacs 用户，并且准备好了在您的日常工作中使用它。当使用它时，您将会发现它带来了高效、事半功倍的体验。

第 4 部分：选项、寄存器和书签

此外，本教程的某些部分（介绍 Emacs 操作的图形元素）特别涉及到 X Window System。为了完成这些部分的学习，您应该拥有一台正常运行的 X 服务器。

了解 Emacs 运行时和显示选项

Emacs 提供了许多可以影响其行为的命令行选项。您通常在启动 Emacs 时没有使用任何选项——或许是一个或两个文件名参数——但是使用选项您可以执行许多高级操作，包括以批处理模式对文件运行 Emacs Lisp 代码。在很多时候，使用这些选项既方便又实用，它们非常适合许多特定的情况，在这一部分中将对这些情况进行描述。（从带命令行选项的命令行中调用 Emacs，这种特定的情况非常有用并且十分常见，特别是在 Shell 脚本或者启动文件中。）

Emacs 使用 GNU 风格的“长”选项，这种选项以两个破折号开头，可以带一个传统的空格字符，或者在选项及其参数之间使用一个等号。这些长选项旨在易于理解和记忆，例如，以反转显示方式启动 Emacs 的长风格的选项是 **--inverse-video**，但是作为结果，与大多数其他工具和应用程序中的选项相比，要键入更长的选项。然而，大多数 Emacs 选项都具有一个传统的单连字符等效部分。本教程在稍后的内容中将给出一个完整的选项列表 表 2，其中包括了这两种类型。

修改文件操作

一些最常见且最有用的 Emacs 选项是那些用于修改或者指定文件的选项。您可以使用这些选项执行许多非常方便的技巧。

自动地打开某些文件

Emacs 所接受的唯一参数是一个文件规范。当您给定一个文件、一个文件列表或者另一个文件规范作为参数时，**Emacs** 将启动并打开指定的文件到它们各自的缓冲区中，就如同您通过调用一系列 **find-file** 功能 (**C-x C-f**)，并依次指定其中的每一个文件：

```
$ emacs /usr/local/share/newdev/inputs/*.txt
```

这个命令启动 **Emacs**，打开 `/usr/local/share/newdev/inputs/` 目录中具有 `.txt` 文件扩展名的所有文件，并将其放入到它们自己的新的缓冲区中。每个缓冲区的名称都与其相应的文件名相同。

如果存在某些始终需要进行编辑的文件，如任务列表、日程安排或者首要任务文件，可以将它们添加到您的启动文件中所使用的 **emacs** 命令行中，这样做是非常有意义的。通过将它们放入到启动文件的命令行中，在 **Emacs** 启动时，就会自动地打开这些文件，并将其放入到相应的缓冲区中：

```
emacs plans tasks
```

Emacs 按照给定的顺序打开文件，所以您最后指定的文件总是当前缓冲区。在前面的示例中，当 **Emacs** 启动时，名为 **tasks** 的文件是当前缓冲区。如果您指定了多个文件，那么 **Emacs** 将窗口从中间分成两半，在其顶部显示一个文件的缓冲区，在其底部显示另一个文件的缓冲区。键入 **C-x 1** 以删除第二个窗口，以便使得整个 **Emacs** 窗口只显示一个缓冲区。

如果您指定三个 或者更多的文件，那么 **Emacs** 将在顶部的窗口中显示最后的文件，并将其作为当前缓冲区；它在其下的第二个窗口中显示一个缓冲区列表，该列表显示了所有打开的缓冲区的名称、它们的大小、它们的主要模式以及它们相对应的文件的名称。同样，您可以键入 **C-x 1** 以删除这个窗口。

您还可以在文件规范的前面使用长的 **--visit** 和 **--file** 选项。例如，清单 1 中的命令行是等价的。

清单 1. 用于指定 **Emacs** 文件名的等价命令行

```
emacs myfile
```

```
emacs --visit myfile
```

```
emacs --visit=myfile
```

```
emacs --file myfile
```

```
emacs --file=myfile
```

当您在为批处理操作构造长的 **Emacs** 命令行时（请参见执行批处理操作部分），这些长选项有时是非常有用的。

开始编辑一个新的文件

如果您所给定的文件在文件系统中并不存在，那么 **Emacs** 将打开一个新的缓冲区，并在模式行中指出它是一个新的文件。例如，如果当前目录是一个名为 `/usr/local/src/projx/inputs/` 的空目录，并且您键入 **emacs README**，那么您将打开一个名为 **README** 的新缓冲区。如果您将该缓冲区保存到磁盘，则写入到一个名为 `/usr/local/src/projx/inputs/README` 的文件。

尝试在您的 **home** 目录中生成一个新的文件：

```
$ emacs myfile
```

请注意，**Emacs** 如何告诉您（在迷你缓冲区中）**myfile** 是一个新的文件。

在该缓冲区中键入一些文本：

Enough! or Too much!

现在，将该缓冲区保存到磁盘：键入 **C-x C-s C-x C-c**，以便将缓冲区保存到您的新文件，然后退出 **Emacs**。请注意，您没有必要给出文件名（如果您访问一个新的文件，您通常会这样做），正如在前面的教程中所描述的，这是因为作为命令行参数您指定的是一个文件的名称，而不是一个缓冲区。当该文件不存在时，**Emacs** 会创建它。（如果您退出而不保存该文件，那么 **Emacs** 将要求您进行确认，尽管该文件将不复存在，但仍然会将您所编辑的内容写入到一个自动保存文件。）

打开文件并定位到某个位置

如果您在一个文件名参数之前使用 **+number**，那么 **Emacs** 将打开该文件，并将光标移动到 **number** 相对应的行。当您启动 **Emacs** 并希望对文件中的某个特定位置进行编辑的时候，这种方法是非常有用的。例如，假设您正在调试一个 **C** 程序 **myprog.c**，并且编译器已经向您发出了警告，告诉您在第 315 行存在一个错误；为了在编译器发现错误的准确行上进行编辑，您可以键入 **emacs +315 myprog.c**，以便打开源文件到一个缓冲区中。

如果您已经阅读了本系列文章中的这些教程，并键入了所提供的示例，那么您应该拥有一个名为 **practice.b** 的示例文件，该文件中包含 William Blake 的“The Tiger”；如果您没有保存它，或者在前面教程的文本操作已经毁坏了副本，那么在本教程的下载部分中提供了可用的副本。

尝试打开 **practice.b**，并使光标定位到第十五行：

```
$ emacs +15 practice.b
```

（使用 **C-x C-c** 退出。）

对于在文件的结尾处打开一个缓冲区，这一选项同样是非常合适的。要完成这项任务，可以给出一个相对于该文件的长度来说非常大的数字。

尝试打开 **practice.b**，并使光标定位于文件的结尾处，：

```
$ emacs +999 practice.b
```

（使用 **C-x C-c** 退出。）

您还可以指定在文件中的水平位置。如果您将垂直的行看作是行，那么跨越每行的水平位置就是列，并且通过在行编号（行位置）后面加上一个冒号字符（:）和表示列的数值，就可以指定水平位置。

尝试打开 **practice.b**，并使光标定位于第三行上的第十二个字符（第十二列）：

```
$ emacs +3:12 practice.b
```

当您执行这个操作时，光标应该定位于 **immortal** 中的 **a** 字符上。（然后，使用 **C-x C-c** 退出。）

如果指定行所拥有的列比您指定的列要少，那么光标不会绕到下一行，而是停留在该行的结尾处。

如果您指定了多个文件，那么您可以在每个文件的前面加上各自的选项，以指定希望光标定位到的行和列。对于每个文件，缺省的行为是在一个缓冲区中打开它，并使光标定位于该文件的第一行和第一列，这与在每个文件名的前面加上 **+1:1** 是等价的。

插入文件

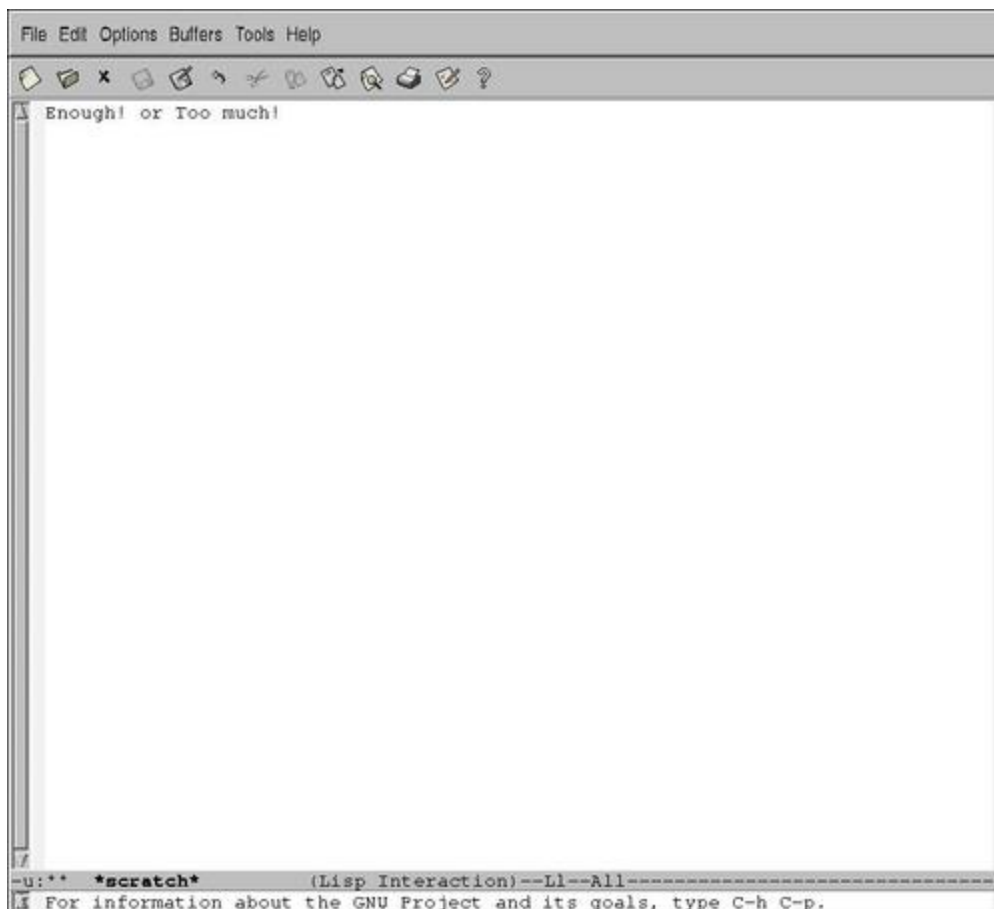
使用 **--insert** 选项插入您作为参数给出的文件的内容。在您希望插入到的文件参数之后，给出这一选项；如果它的前面不使用任何文件参数，那么该选项会将给定的文件插入到一个临时缓冲区内。

尝试下面的操作：

```
$ emacs --insert myfile
```

这个操作将使用一个临时缓冲区启动 **Emacs**，该缓冲区应该与图 1 中所示类似，其中插入了新的文件 **myfile**，并且光标定位于缓冲区的开头处。

图 1. 从命令行插入一个文件到 **Emacs** 缓冲区中



（使用 **C-x C-c** 退出。）

现在，尝试将您的新文件插入到 **practice.b** 中：

```
$ emacs practice.b --insert myfile
```

这个操作将启动 **Emacs**，同时在一个新的缓冲区中打开 **practice.b** 文件，并且在其开头处插入 **myfile** 的内容。

（使用 **C-x C-c** 退出，键入 **n**，然后键入 **yes** 以确定执行退出操作，而不进行保存。）

您可以将一项插入操作与一个位置选项组合在一起，并且您可以同时给出一些附加的参数。尝试下面的操作：

```
$ emacs +10:5 practice.b --insert myfile anotherfile
```

这个操作将启动 **Emacs**，同时使 **practice.b** 的内容处于一个新的缓冲区中，在第十行的第五列插入 **myfile**，然后在一个新的缓冲区中打开名为 **anotherfile** 的新文件，并将其作为当前缓冲区。

（使用 **C-x C-c** 退出，键入 **n**，然后键入 **yes** 以确定执行退出操作，而不进行保存。）

改变显示属性

除了指定文件操作的选项之外，其次最常用的 **Emacs** 选项可能就是那些与 **X Window System** 相关的选项。您可以使用这些选项更改 **Emacs** 在 **X** 中的显示方式。这些选项中的大多数是标准的 **X Window System** 选项，它们适用于所有的 **X** 客户端。

在 **xterm** 中运行 **Emacs**

与 **X** 相关的最有用的 **Emacs** 命令行选项之一是 **-nw**，您在本系列文章的第 1 部分教程（请参见参考资料）中曾看到过该选项的示例。这个选项在当前终端中打开 **Emacs**，而不是创建一个它

自己的新的 X 客户端窗口。在许多情况下，该选项是非常有用的，例如，当您正通过一个慢速的连接运行远程终端窗口，并且不希望启动本地的 X 客户端窗口时；或者当您正在一个终端中进行许多操作，并且希望对一个文件进行快速编辑，而不启动一个新的窗口以打开 Emacs 时。

如果您的 X 正在运行，那么请在一个终端窗口中尝试执行下面的操作：

```
$ emacs -nw
```

当您键入这一命令时，Emacs 将以终端模式启动，并且出现在该终端中，而不是产生一个新的 X 客户端窗口。（键入 C-x C-c 以退出。）

指定服务器

使用 --display 选项，以指定应该在什么位置显示 Emacs X 客户端窗口。它接受 X 服务器的主机名作为一个参数，后面紧跟着一个冒号和显示编号，这个显示编号表示该主机上特定的 X 服务器的编号（它通常是 0，但根据运行多个 X 服务器的系统的不同，它的值也有所不同）。

例如，这一命令在本地主机的缺省服务器上显示客户端，并且它与不给出任何选项的操作是等价的：

```
emacs --display :0
```

要在名为 rs6000 的远程主机的缺省服务器上打开 Emacs 窗口，请键入：

```
emacs --display rs6000:0
```

指定窗口大小

--geometry 选项是另一个标准的 X 选项。采用以下格式，给出客户端窗口的宽度、高度以及可选的 X 和 Y 偏移量：

```
WIDTHxHEIGHT[{+-}XOFFSET{+-}YOFFSET]
```

从屏幕的左上角开始计算偏移量，它既可以是正值，也可以是负值。

如果您正在一个 X 服务器上运行本教程，那么现在您可以尝试它。要在您的桌面的左上角打开一个 25 个字符宽和 250 个字符高的长窗口，请键入：

```
$ emacs --geometry 25x250+0+0
```

（使用 C-x C-c 退出。）

设置 Emacs 的字体和颜色

您可以使用其他的标准 X 选项以便使用指定的文本字体、颜色（前景），以及其后的空白区（背景）启动 Emacs。

要指定一种字体，可以给出一种 X 字体名作为 --font 选项的引号括起来的参数。可以通过大量的选项（请参见表 1）来指定颜色。

表 1. 用于指定颜色的 Emacs 命令行选项

选项	描述
--foreground-color color -fg color	设置前景颜色为 color。
--background-color color -bg color	设置背景颜色为 color。
--border-color color -bd color	设置边框颜色为 color。

<code>--cursor-color color</code> <code>-cr color</code>	设置光标颜色为 <code>color</code> 。
<code>--mouse-style color</code> <code>-ms color</code>	设置鼠标指针颜色为 <code>color</code> 。

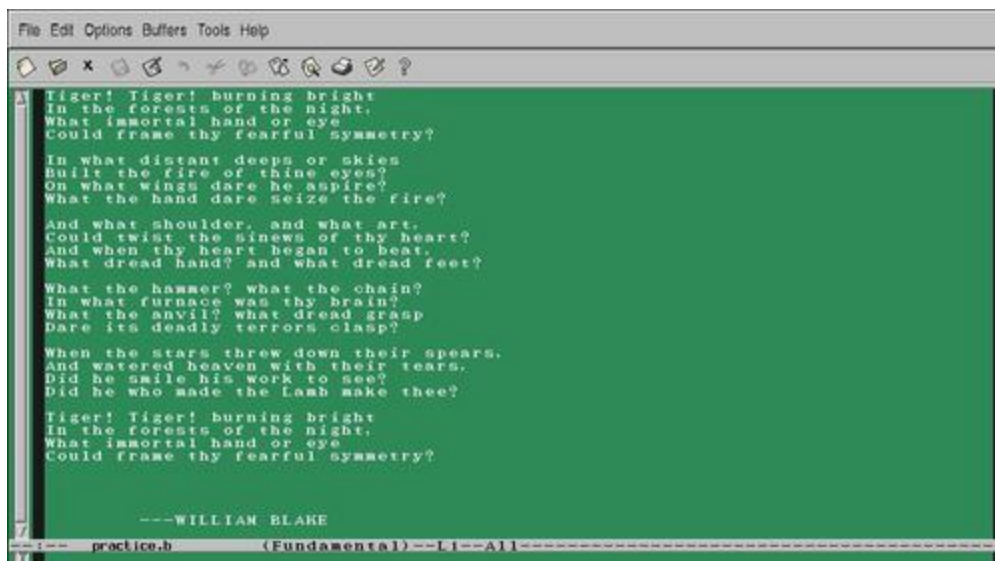
您还可以组合使用字体和颜色选项。尝试一次设置所有这些选项：

```
$ emacs -fn "-misc-fixed-medium-r-normal-jp-13-120-75-75-c-120-iso10646-1"
```

```
-bg "sea green" -fg "white" -cr "brown1" -ms "royalblue1" practice.b
```

这一命令在一个 X 客户端窗口中启动 Emacs，它与图 2 中所示类似。

图 2. 指定 Emacs 的颜色和字体



您可以使用哪些特别的字体和颜色，这取决于您的 X 服务器及其配置。

控制启动文件

Emacs 允许您运行一个启动文件，该文件也称为初始化文件，它可以包含要执行的 Emacs Lisp 代码，并且通常用于设置各种属性和变量，您可以使用它来自定义您的编辑环境。它是一个名为 `.emacs` 的隐藏文件，位于您的 `home` 目录中。（自定义您的 `.emacs` 文件是本系列文章中后续教程的一个主题。）

通过给定 `-q` 选项，您可以指定 Emacs 不运行任何现有的启动文件。当您希望运行一个 Emacs 会话，而不使用任何已有的自定义设置时；当您正在使用一个新的 Emacs 功能时；当某处出现了奇怪的行为时；或者您不确定是否您自己的某个自定义设置是导致问题的原因时，对于调试新添加到您的启动文件中的内容，这个选项是非常有用的：

```
emacs -q
```

因为每个人的 `.emacs` 文件是不同的，所以您可能希望尝试使用其他人的启动文件，以实现相应的更改，并观察其他人如何自定义他们的编辑环境。您可以使用 `-u` 选项来完成这一任务，给定该用户的名称作为这个选项的参数。这对于试验本地 Emacs 向导的自定义设置是非常有用的，例如用户 `joe`：

```
emacs -u joe
```

只要这个 `.emacs` 启动文件对您来说是可读的，那么该命令将使用用户 `joe` 的 `home` 目录中 `.emacs` 启动文件来启动 Emacs。如果用户 `joe` 并不存在，且没有这个 `.emacs` 文件，或者该文件是不可读的，那么 Emacs 在启动时将不使用任何初始化文件。

Emacs 还有一个全局启动文件 `site-start.el`，它通常保存在 `/etc/emacs/` 目录树中。
`--no-site-file` 选项将启动 Emacs，而不执行该文件中的 Emacs Lisp。

最后，`--debug-init` 将启用 Lisp 调试器，以便检查您的启动文件，当您正在向它添加某些内容并且出现了问题时，这一操作是非常有用的。您甚至可以使用它来调试另一个用户的启动文件：

```
emacs -u joe --debug-init
```

为 Emacs Lisp 的执行使用选项

Emacs 提供了一些用于 Lisp 代码执行的命令行选项；当您正以批处理模式运行 Emacs 时，这些选项特别有用，如下所述。

使用 `-l` 加载一个 Lisp 文件，给出该文件的名称作为参数。Emacs 首先在当前目录中查找该文件，然后在 `EMACSLOADPATH` 环境变量（如果设置了的话）中的任何目录中查找。要从文件系统的其他地方加载一个 Lisp 文件，可以给出完整的路径名作为参数。

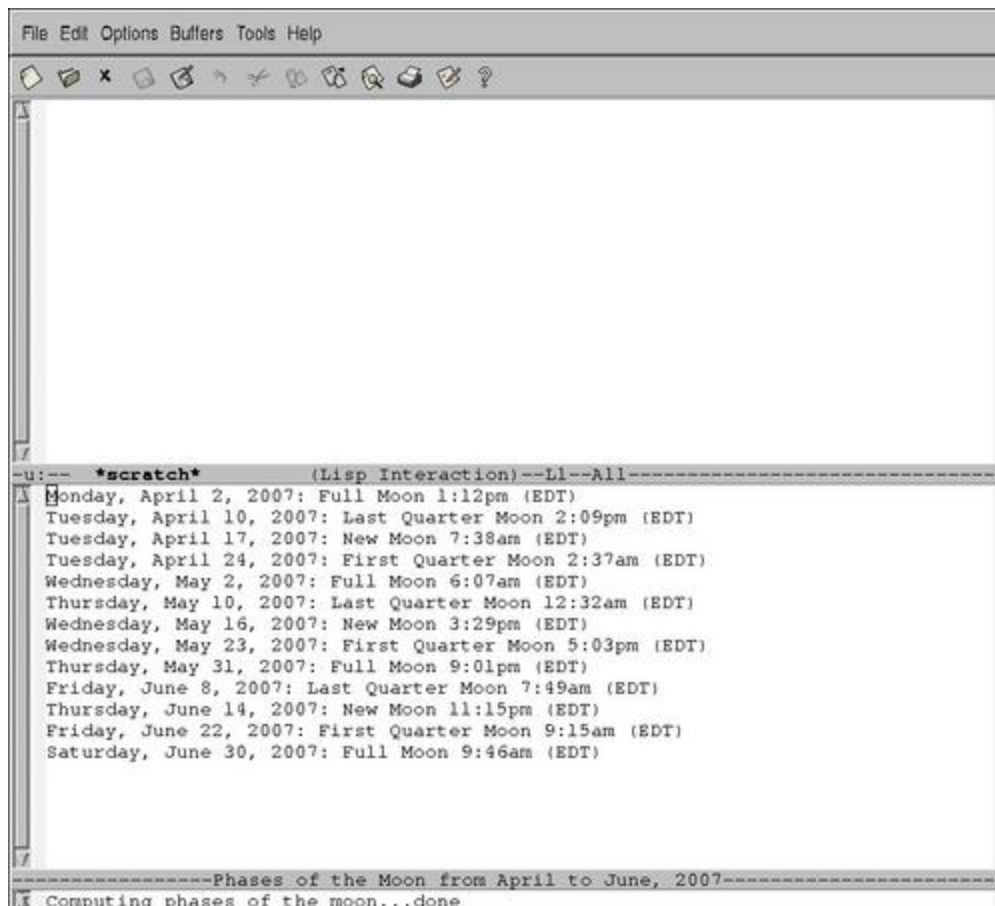
要从命令行运行 Emacs Lisp 函数，可以使用 `-f` 选项，并给出 Lisp 函数的名称，以将其作为一个选项来执行。如果该函数带有任何参数，那么请确保提供这些参数。

例如，尝试启动 Emacs，并运行 `phases-of-moon` 函数：

```
$ emacs -f phases-of-moon
```

当您在执行这一操作时，您的 Emacs 会话应该与图 3 中所示类似，在 Emacs X 客户端中有两个窗口：一个临时缓冲区和一个包含当前月相的日期的窗口。（使用 `C-x C-c` 退出。）

图 3. 使用 `phases-of-moon` 函数启动 Emacs



要启动 Emacs 并执行一个 Lisp 表达式，可以使用 `--eval` 选项；并在其后紧跟一个用引号括起来的 Lisp 表达式。

例如，尝试使用您的新文件启动 Emacs，移动到缓冲区的末尾，并执行 Lisp 代码 (`insert (current-time-string)`)，它会将当前时间和日期插入到当前缓冲区中：

```
$ emacs +999 myfile --eval "(insert (current-time-string))"
```

（使用 `C-x C-c` 退出，键入 `n`，然后键入 `yes` 以确定执行退出操作，而不进行保存。）

执行批处理操作

您可以采用批处理模式运行 Emacs，以实现非交互处理。当您给出 `--batch` 选项时，Emacs 将不对标准输出进行写操作，并且它将从标准输入（而不是迷你缓冲区）中获得输入。（它仍然会对标准错误进行写操作。）

通常，您可以使用 `-l` 或者 `-f` 选项，正如刚才所介绍的，使用 `--batch` 以执行 Lisp 代码或者以非交互方式运行相应的函数。如果您使用 `--batch` 调用 Emacs，并且没有使用任何其他选项，那么 Emacs 将立刻退出。

否则，Emacs 将执行给定的命令，直到一个导致退出的命令、或者直到它碰到一个 `--kill` 选项，该选项将立刻退出 Emacs，可以将它放在命令行的结尾，以便在完成批处理操作时退出 Emacs。

尝试采用批处理模式运行 Emacs，以便将 `myfile` 的内容插入到 `practice.b` 的第三十五行，使用 `save-buffer` 函数保存它，然后退出：

```
$ emacs --batch +35 practice.b --insert myfile -f save-buffer --kill
```

现在，查看您的 `practice.b` 文件，它应该包含额外的一行内容。

您可以简单地使用另一个 Emacs 单行程序删除该行。尝试下面的操作：

```
$ emacs --batch +35 practice.b -f kill-line -f save-buffer --kill
```

不要忘记查看 `practice.b`，并确保已经删除额外的行。

命令行选项表

表 2 总结了到目前为止所介绍的众多 Emacs 命令行选项，给出了它们的长、短选项名（如果适用的话），并描述了它们的功能。

表 2. Emacs 命令行选项

选项	描述
<code>--visit=filespec</code> <code>--file=filespec</code> <code>filespec</code>	打开 <code>filespec</code> 到各自的缓冲区，以进行编辑。
<code>+row[:column]</code>	将光标移动到该文件中的 <code>row</code> 行和（可选的）水平位置 <code>column</code> 列（缺省值是 <code>+1:1</code> ）。
<code>--insert file</code>	将文件 <code>file</code> 插入到缓冲区的开头。
<code>--debug-init</code>	对 <code>.emacs</code> 启动文件使用 Lisp 调试器。
<code>--no-init-file</code> <code>-q</code>	不运行任何 <code>.emacs</code> 启动文件。
<code>--no-site-file</code>	不运行全局 <code>site-start.el</code> 文件。
<code>-u user</code>	使用用户 <code>user</code> 的 <code>.emacs</code> 启动文件。

--user user	
--funcall function -f function	执行 Emacs Lisp 函数。
--eval expression --execute expression	执行 Emacs Lisp 表达式 expression。
--load file -l file	执行文件 file 中的 Emacs Lisp 指令。
-batch --batch	使用批处理（非交互）模式。
-kill --kill	当处于批处理模式时，退出 Emacs。
--name name	使用 name 作为 Emacs X 客户端窗口的名称（缺省值是“emacs”）。
-T title --title title	使用 title 作为 Emacs X 客户端窗口的标题（缺省值是 name@FQDN，其中 FQDN 是该主机的完全限定域名）。
--reverse-video -r	使用反向显示方式，交换前景和背景的颜色。
--iconic -iconic	启动 Emacs，并将其作为一个图标，而不是一个活动窗口。
--icon-type -i	当图标化 Emacs 窗口时，使用 Emacs 图标（通常是 /usr/share/emacs/version/etc/gnu.xpm），而不是窗口管理器的任何缺省值。
-fn name -font name	使用 name 作为 Emacs 窗口字体。
--border-width width -bw width	将窗口边框设置为 width 像素。
--internal-border width -ib width	将窗口内部边框设置为 width 像素。
--g dimensions --geometry dimensions	根据给定的 X 窗口尺寸 dimensions（生成窗口的缺省值是 80x40 个字符）设置窗口的宽度、高度和位置。
--foreground-color color -fg color	将前景色设置为 color。
--background-color color -bg color	将背景色设置为 color。
--border-color color -bd color	将边框颜色设置为 color。

<code>--cursor-color color</code> <code>-cr color</code>	将光标颜色设置为 <code>color</code> 。
<code>--mouse-color color</code> <code>-ms color</code>	将鼠标指针颜色设置为 <code>color</code> 。
<code>-d name</code> <code>--display name</code>	在与 <code>name</code> 相对应的 X 显示器上打开 Emacs 窗口。
<code>-nw</code> <code>--no-windows</code>	在 X 中，不使用 X 客户端窗口，而是在当前终端窗口中打开。这一选项不影响控制台会话。
<code>-t file</code> <code>--terminal file</code>	将标准 I/O 重定向到文件 <code>file</code> ，而不是终端。

使用 Emacs 寄存器

您已经知道，使用您在本系列文章的上一个教程中所学的技术（请参见参考资料），您可以返回到标记处，并且您可以返回到标记环中的任何位置。但是除此以外，您还可以通过另一种 Emacs 工具，在缓冲区中设置任意多个位置，并在任何时候移动到这些位置：寄存器。

Emacs 寄存器 是通用的存储机制，它可以存储很多内容中的一项，包括文本、矩形区块、缓冲区中的位置，或者某些其他值或设置。每个寄存器都有一个标签，您可以使用单个字符来引用寄存器。可以重定义寄存器，但是它一次只能包含一项内容。一旦您退出 Emacs，将清空所有的寄存器。

您可以插入保存在寄存器中的文本。当您重新获得寄存器的设置或者配置时，我们称其为恢复该寄存器；如果寄存器包含一个您希望返回的缓冲区中的位置，我们称其为跳转 至该寄存器中保存的位置。

所有的 Emacs 寄存器命令都是以 `C-x r` 开头的。

本文接下来的部分将向您介绍用于设置、查看、恢复和跳转至寄存器的命令。

保存一个寄存器

要在一个寄存器中保存当前光标，可以运行 `point-to-register` 功能，它与 `C-x r space` 进行了绑定，同时给出寄存器的名称，它可以是任何字母数字字符。寄存器的名称是区分大小写的，`x` 和 `X` 指的是两个不同的寄存器。

要将一个区域 复制到寄存器，可以使用 `copy-to-register` 功能，它与 `C-x r s` 进行了绑定。

要将一个矩形区块复制到寄存器，使用 `copy-rectangle-to-register` 功能，它与 `C-x r r` 进行了绑定。

尝试将当前光标保存到寄存器 `X` 中：

启动 Emacs，并使光标定位于练习文件的第二行的第九个字符，即单词 `forests` 中的字符 `o` 处：

```
$ emacs +2:9 practice.b
```

键入 `C-x r space X` 以便在寄存器 `X` 中保存光标位置。

查看一个寄存器

使用 `view-register` 功能查看寄存器中的内容。

尝试查看寄存器 `X` 中的内容：

键入 `M-x view-register` 并按 `Enter`。

当在迷你缓冲区中提示您时，键入 `X` 作为要进行查看的寄存器。寄存器的名称始终是单个字符，所以您无需按 `Enter`。

当您运行这一功能时，将打开一个新的窗口，以显示寄存器 X 中所包含的内容，在这个示例中，是您的 `practice.b` 缓冲区中的一个光标位置。键入 `C-x 1` 以关闭该窗口。

跳转至一个寄存器

要跳转至您保存在一个寄存器中的光标，或者要恢复一个窗口或框架配置，可以使用 `jump-to-register` 功能，它与 `C-x rj` 进行了绑定。

尝试跳转至您已保存在寄存器 X 中的光标位置：

键入 `M->`，以移动到缓冲区的末尾，远离您已经保存的光标位置。

键入 `C-x rjX` 以跳转至保存的光标位置。现在，您返回到了 `forests` 中。

恢复一个寄存器

从一个寄存器恢复文本，无论是区域还是矩形区块，其操作方式是不同的。要从寄存器插入一个文本区域或者矩形区块，可以使用 `insert-register` 功能，它与 `C-x ri` 进行了绑定。它在光标处插入寄存器中的文本，并保持光标位于插入的内容之前。如果您给出的寄存器是一个光标位置，那么将在光标处插入该位置编号（缓冲区中字符的数目）。

尝试下面的操作：

单击并拖曳鼠标左键，选中 `and what art`，以便将这三个单词标记为一个文本区域。

将该区域保存到寄存器 `q`：键入 `C-x r s q`。

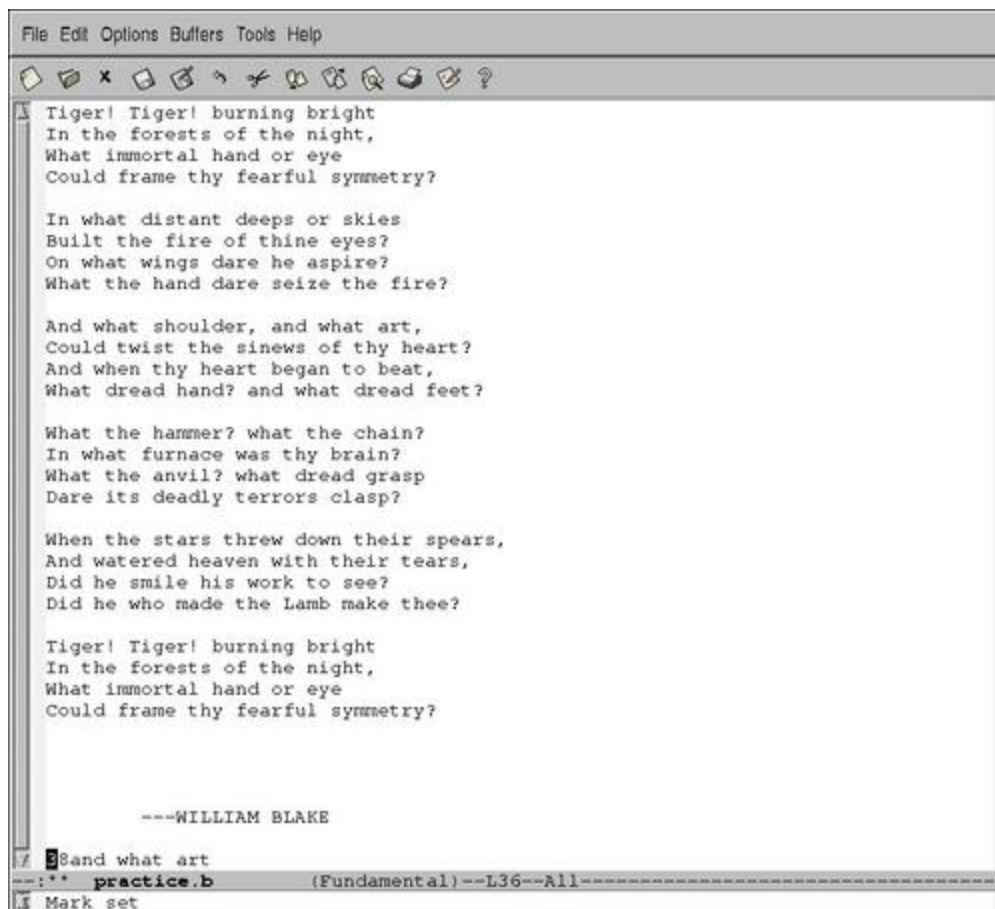
移动到缓冲区尾，并插入一个空白行：键入 `M->` 并按 `Enter`。

插入您刚刚保存的寄存器的内容：键入 `C-x r i q`。请注意，光标设置到您所插入的文本的开头，而不是该文本的结尾。

插入早些时候保存的寄存器 X 中的内容，其中包含光标的位置：键入 `C-x r i X`。

如果您已经执行了本教程中的所有示例，那么您的 `Emacs` 会话应该与图 4 中所示类似。

图 4. 恢复文本内容和位置寄存器



所恢复的编号 38 表示 `forests` 中的 `o`（寄存器 X 的光标位置）是缓冲区中的第三十八个字符。通过在该文件中从第一个位置开始将光标向前移动 37 个字符，您可以验证它：

```
M-< M-3 M-7 C-f
```

Emacs 寄存器命令表

表 3 列出了使用 Emacs 寄存器的各种命令和键，给出了它们的功能名，并描述了它们的含义。

表 3. 使用寄存器的 Emacs 命令

键盘输入	功能	描述
C-x r space X	point-to-register	将光标保存到寄存器 X。
C-x r s X	copy-to-register	将区域保存到寄存器 X。
C-x r r X	copy-rectangle-to-register	将选定的矩形区块保存到寄存器 X。
未定义	view-register	查看一个给定的寄存器的内容。
C-x r j X	jump-to-register	将光标移动到寄存器 X 中给定的位置。
C-x r i X	insert-register	在光标处插入寄存器 X 的内容。

使用 Emacs 书签

Emacs 提供了保存缓冲区中位置的另一种工具。这些 Emacs 书签 的工作方式与寄存器相同，但是它们的标签可以超过一个字符长，而且它们比寄存器更为持久：如果保存了书签，那么您可以在两个不同的会话之间使用它们。它们将一直保留下来，直到您删除它们。正如它们的名称所表示的，对于保存您在缓冲区中的位置，以便您稍后可以返回到该位置（通常是在以后的 Emacs 会话期间），使用书签是非常方便的。

这个部分将向您介绍使用、设置、列出、保存和删除 Emacs 书签。

设置一个书签

对于保存光标位置，书签与寄存器是很相似的，不同之处在于书签能够保持到当前会话之外。对于标记您在文件中的位置，并在稍后返回到该位置，书签是非常方便的，您可以为很多文件设置书签，并且您可以在单个文件中设置很多书签。

要在当前缓冲区中，为您正在访问的文件的当前光标设置一个书签，可以运行 **bookmark-set** 功能，它与 **C-x r m** 进行了绑定。

这一命令后面紧跟您的书签的名称；在缺省情况下，它就是当前缓冲区的名称。

现在，尝试保存一些书签：

启动 Emacs，并使光标定位于您的练习文件中的第 20 行：

```
$ emacs +20 practice.b
```

键入 **C-x r m** 以便将这个光标作为书签保存。

当 Emacs 请求为这一书签提供相应的名称时，可以按 **Enter** 以使用该缓冲区的名称 (practice.b)。

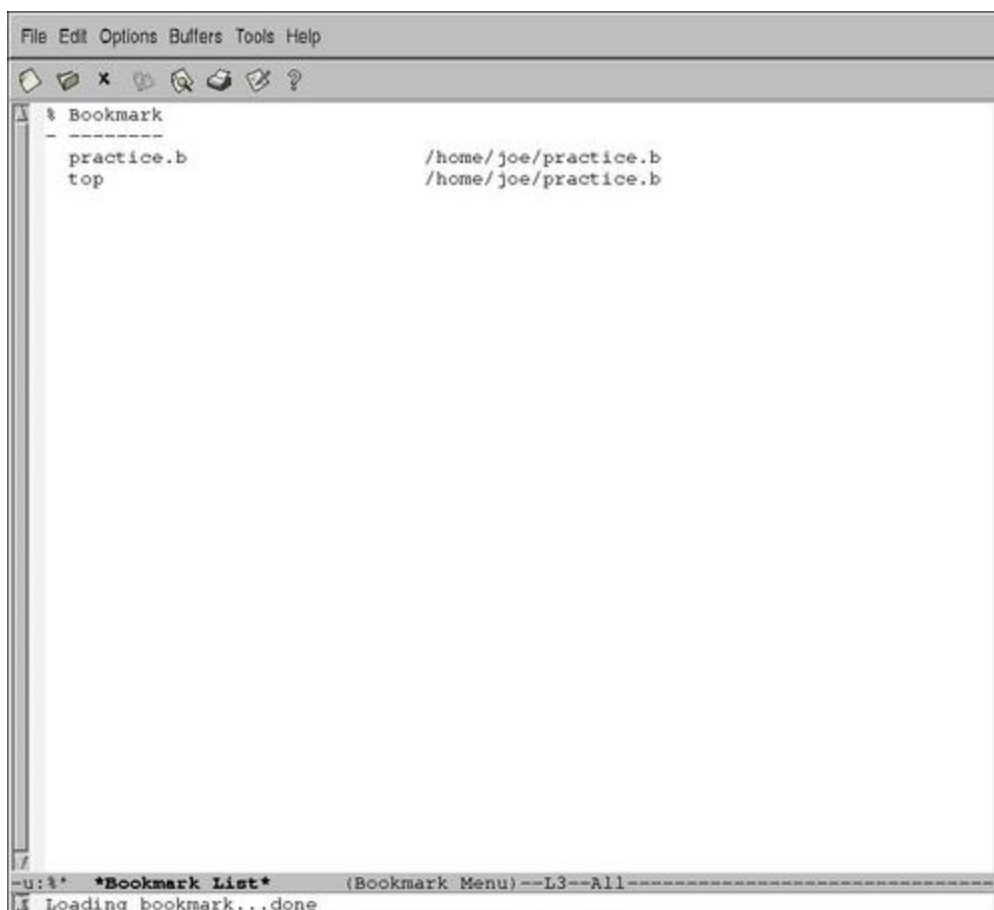
键入 **M-<** 以移动到缓冲区的顶部。

保存一个名为 **top** 的新书签：键入 **C-x r m top**。

列出您的书签

bookmarks-bmenu-list 功能可以列出一个由已设置的所有书签组成的菜单。通过键入 **C-x r l** 运行该功能，而您的会话应该与图 5 中所示类似。

图 5. Emacs 的书签菜单



您可以通过按 **Enter** 从该列表中选择一个书签。**Emacs** 将在一个新的缓冲区（如果尚未在一个缓冲区中打开它）中访问该文件，并将其作为当前缓冲区，同时将光标移动到该书签的位置。

现在，尝试运行这一功能，并使用箭头键选择您的“**top**”书签。

跳转至一个书签

您可以通过直接跳转 至某个书签来访问它，而无需从您的书签主列表中选择它。要跳转至某个特定的书签，可以使用 **bookmark-jump** 功能，**C-x r b**。这一命令将移动到特定文件中的给定位置；如果尚未在缓冲区中打开这个带书签的文件，那么这一命令将打开它。

在迷你缓冲区中会提示您输入要跳转到的书签。这个提示使用了自动完成功能，所以您只需键入该书签名前面足够的字母，以使其能够区别于其他的书签即可，然后按 **Tab** 以选择该书签。

尝试跳转到您的 **practice.b** 书签：键入 **C-x r b prac**，按 **Tab** 键，然后按 **Enter**。

删除一个书签

当您使用一个现有书签的标签来设置新书签（请参见设置一个书签部分）时，您将使用新的书签取代原来的书签。这是一种“删除”书签的方式，即使用新的值来取代其原来的值。当您在阅读一个大文件时，如果您仅仅保留一个书签以标记您的位置，上述方法是很常见的，因为您每次设置该书签时都替换了它先前的值。

但是您可以同时删除一个书签，那么该标签不再指向任何文件中的任何位置。要实现这一点，可以运行 **bookmark-delete** 功能，并在被询问时，给出要进行删除的书签的名称。

尝试删除您的“**top**”书签：

键入 **C-x bookmark-delete** 并按 **Enter**。

当被询问时，给出要删除的书签的名称：键入 **top** 并按 **Enter**。

通过使用 **C-x r l** 调出书签列表，您可以验证已经删除了该书签。

保存您的书签

在 Emacs 的新版本中，一旦您生成了任何书签，那么您为当前 Emacs 会话所设置的书签将自动地保存到您的永久书签文件中。您的书签文件是您的 home 目录中的一个名为 .emacs.bmk 的隐藏文件。

如果没有配置 Emacs 以使其自动保存您的书签，那么通过运行 bookmark-save 功能，可以将它们保存到您的 .emacs.bmk 文件中。如果当您退出时没有保存任何新的书签，那么 Emacs 将询问您是否想要保存它们。

Emacs 书签命令表

表 4 列出了使用 Emacs 书签的各种命令和键，给出了它们的功能名，并描述了它们的含义。

表 4. 使用书签的 Emacs 命令

键盘输入	功能	描述
C-x r m Bookmark	bookmark-set	设置一个名为 Bookmark 的书签。
C-x r l	bookmarks-bmenu-list	列出所有已保存的书签。
	bookmark-delete	删除一个书签。
C-x r b Bookmark	bookmark-jump	跳转至名为 Bookmark 的书签中所设置的位置。
未定义	bookmark-save	将所有的书签保存到书签文件 ~/.emacs.bmk 中。

总结

本教程是关于使用 Emacs 编辑环境的系列文章的第 4 部分，在本教程中您了解了如何使用 Emacs 控制编辑会话某些方面的三个部分：启动 Emacs 时许多可用的命令行选项、功能强大的 Emacs 寄存器、用于设置和保存位置以及数据的书签工具。您还了解了许多功能强大的编辑技术，这些技术将使您成为一名工作效率更高的 Emacs 用户。

本系列文章的后续教程建立于您在本教程中所学到的知识的基础之上，并且将引导您更深入地掌握这一功能全面的编辑应用程序。

第 5 部分：确定您的 Emacs 视图的形状

因为本教程专门研究对 X 窗口系统环境中 Emacs 图形元素的操作，所以您应该拥有一台正常运行的 X 服务器。

本教程使用了一个由两个文件组成的示例数据集，并在一个存档文件中提供了该数据集（请参见下载部分以获取相应的链接）。

对您的 Emacs 会话进行划分和分区

您可以使用许多命令来控制 Emacs 窗口，它是您所看到的、以 Emacs X 客户端窗口为框架的一个缓冲区的视图，这个 X 客户端窗口中包含该缓冲区本身及其模式行（在它的下方）。这些命令通过各种方式对您的窗口进行分区，以允许您同时查看多个缓冲区。实际上，这些都是用于 Emacs 客户端窗口操作的所有命令中最常用的一些命令。

本教程中的示例使用了一个示例数据集，并在一个压缩的存档文件中提供了该数据集（请参见下载部分）。这个存档文件中包含一个 **tar** 存档文件，而 **tar** 存档文件包含了两个纯文本文件 **innocence** 和 **experience**，它们的内容分别是 **William Blake** 的 **The Songs of Innocence** 和 **The Songs of Experience** 的完整文本，其中您将看到来自以前的教程的一些文本。要开始学习本教程，首先将这两个文件解压缩到您自己的示例目录中。

垂直地划分一个窗口

对 **Emacs** 窗口进行分区的最常见的方式可能是从屏幕的中间将其分为两半。这一操作将生成两个新的窗口，其中每个窗口都具有它自己的模式行，并且每个窗口的高度都大约为原始高度的一半。您以水平方向对 **Emacs** 屏幕进行了切分，所以这两个新的窗口在一个垂直的列中相互堆叠；因而，我们称这种窗口操作为垂直划分。

要进行这样的划分，可以运行 **split-window-vertically** 功能，它与 **C-x 2** 按键进行了绑定。

尝试使用示例文件进行这一操作：

通过在包含这两个示例文件副本的目录中键入 **emacs**，启动 **Emacs**。

打开这些文件：

C-x C-f innocence Enter C-x C-f experience Enter

对窗口进行划分：

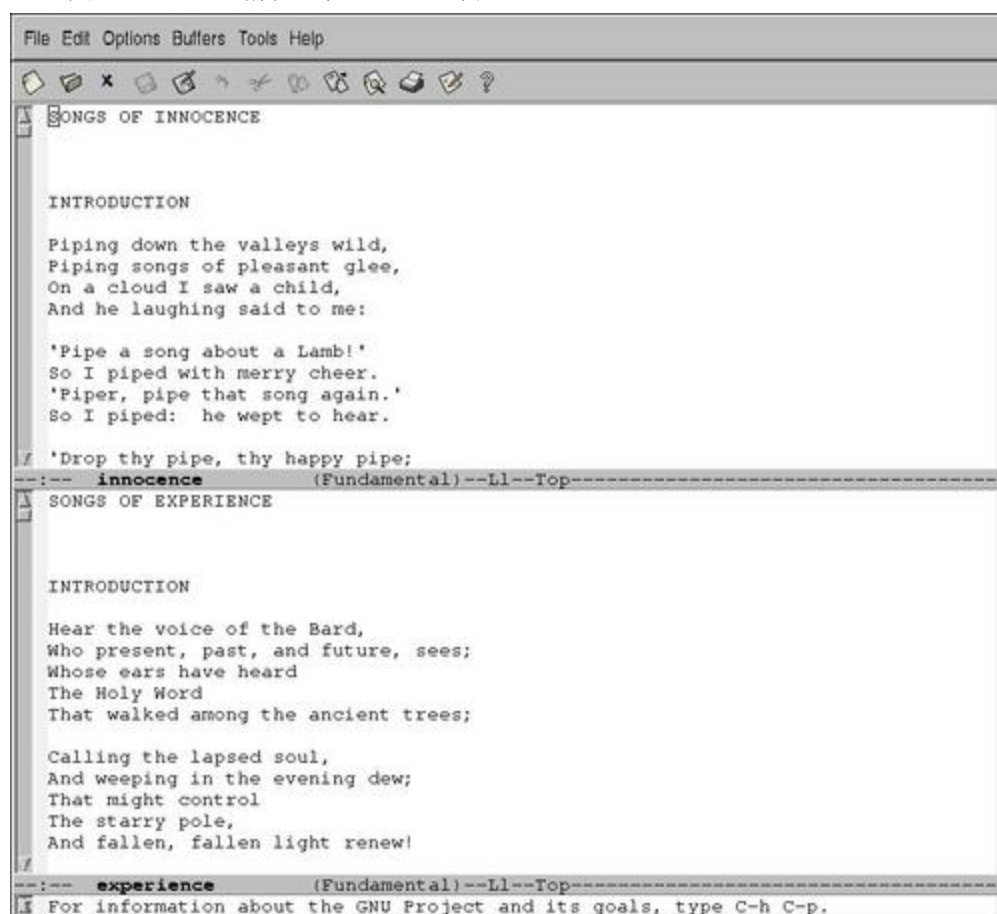
C-x 2

请注意，在对窗口进行划分时，活动的缓冲区（名为 **experience**，是您最后打开的一个缓冲区）将出现在这两个窗口中。在顶部窗口中的缓冲区现在是活动缓冲区，并包含一个活动光标。在这个顶部窗口中切换到 **innocence** 缓冲区：

C-x b Enter

您的 **Emacs** 会话应该与图 1 中所示类似。

图 1. 垂直地划分一个 **Emacs** 窗口



顺便提一下，当您使用两个文件作为参数启动 **Emacs** 时，将得到相同的两个垂直划分的缓冲区，这是由 **Emacs** 自动划分的，正如本系列文章第 4 部分教程“**Emacs** 选项、寄存器和书签”中所描述的。键入 **C-x C-c** 以退出 **Emacs**，然后针对示例文件尝试这一操作：

```
$ emacs innocence experience
```

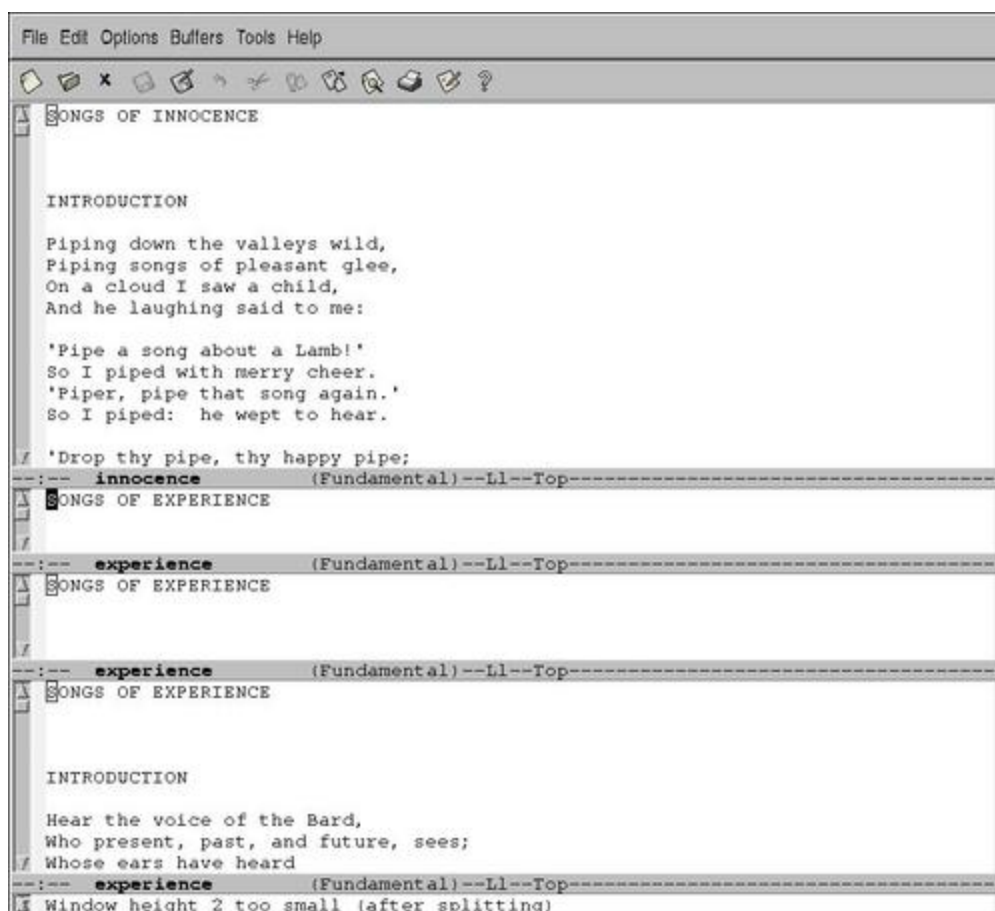
请注意其中的一处区别：当您两个文件指定作为命令行参数时，底部的窗口（其内容为第二个参数）将成为活动窗口。

您并不限于同时使用两个窗口，**Emacs** 可以根据屏幕的大小显示许多窗口。但请记住，当您进行一次划分时，**Emacs** 将对当前窗口进行划分，而不会涉及到所有其他窗口。

例如，在这两个窗口中，底部的窗口现在是活动的；键入 **C-x 2**，以便仅划分下面的这个窗口。

现在，将您的 **Emacs** 屏幕划分为了三个窗口，并且中间的窗口是活动的。键入 **C-x 2**，以便再次划分中间的窗口，在 **Emacs** 屏幕中一共生成了四个窗口，然后再次键入 **C-x 2**，以尝试对第二个、很小的那个窗口进行划分。根据 **Emacs X** 客户端主窗口的尺寸，对分割的窗口所进行的这种递归划分迟早将会中止，即当任何新的窗口太小，以至于不能够显示至少两行文本和模式行时；这时候，**Emacs** 将发出蜂鸣声，并在迷你缓冲区中报告，无法再继续对给定的窗口进行划分。您的会话应该与图 2 中所示类似。

图 2. 多个 **Emacs** 垂直划分



您可以使用这些划分的窗口同时查看多个缓冲区，但是它同样有助于在多个窗口中查看相同的缓冲区，更确切的说，在不同的窗口中查看缓冲区的不同部分。在您查看缓冲区中另一个部分的同时，对缓冲区中的某个部分进行编辑，这是强大的编辑功能中最有用的技巧之一。它是非常有用的。

现在，请尝试下面的操作：

通过键入 **C-x C-c** 退出 **Emacs**，然后使用一个文件启动它：

```
$ emacs experience
```

键入 **M->**，以移动到缓冲区的末尾。

键入 **C-x 2** 以垂直地划分窗口，并且这两个窗口中显示了相同的缓冲区。

移动到活动窗口中的缓冲区的顶部：键入 **M-<**。

使用向下箭头键，将光标向下移动几十行。当该缓冲区出现在顶部的窗口中时，您可以在其中进行移动，但是 **Emacs** 总是在底部的窗口中显示相同的缓冲区的尾部。

当您在划分一个窗口时，**Emacs** 将自动地决定划分的尺寸，它通常从中间将缓冲区划分为两半，所以垂直划分所创建的两个新的缓冲区，其中每一个都恰好是原始窗口大小的一半。但是您也可以使用行数来指定其大小，即通过在划分命令之前使用一个数字，指定垂直划分的顶部窗口应该具有的行数（包括它的模式行）。

尝试将该缓冲区一分为二，并使顶部的缓冲区为九行内容的高度：

通过键入 **C-x C-c** 退出 **Emacs**，然后使用一个文件启动它：

```
$ emacs experience
```

键入 **M-9 C-x 2** 以划分该缓冲区，并使顶部的缓冲区为九行内容的高度。

要为底部的缓冲区而不是顶部的缓冲区指定行数，可以使用一个负数。

尝试将该缓冲区一分为二，并使底部的缓冲区为四行内容的高度：

通过键入 **C-x C-c** 退出 **Emacs**，然后使用一个文件启动它：

```
$ emacs experience
```

键入 **M-- M-4 C-x 2** 以划分该缓冲区，并使底部的缓冲区为四行内容的高度。

C-x 2 可以进行垂直划分，但是用于划分窗口的许多重要的键绑定都是使用 **C-x 4** 前缀作为开始的。

要在一个新的缓冲区中以及一个新的窗口中打开一个新的文件，可以使用 **find-file-other-window** 功能，它与 **C-x 4 f** 进行了绑定。然后，给出该文件的文件名。要打开只读文件，则可以使用 **find-file-read-only-other-window**，并给出其文件名。它与 **C-x 4 r** 进行了绑定。

尝试在其他窗口中的一个新的缓冲区中打开 **innocence** 文件：键入 **C-x 4 f innocence Enter**。

尽管当您指定一个四行的窗口时，另一个窗口已经存在，但是请注意这个命令如何调整它的大小，以便活动窗口和新的窗口具有大致相同的尺寸。

switch-to-buffer-other-window 功能与 **C-x 4 b** 进行了绑定，它用于垂直地划分窗口，并允许您选择要在新的窗口中显示哪一个缓冲区，而该窗口也将成为活动窗口。

尝试使用这个命令，将另一个窗口中的缓冲区切换到 **innocence** 缓冲区：键入 **C-x 4 b innocence Enter**，请注意，它不会对您输入该命令的窗口进行更改，现在，它也显示了 **innocence** 缓冲区。

要在另一个垂直的窗口中显示一个新的缓冲区，但保持活动光标仍然位于当前窗口中，可以运行 **display-buffer** 功能，它与 **C-x 4 C-o** 进行了绑定。它提示输入需要在另一个窗口中显示的缓冲区的名称，但是当前窗口仍然是活动的。如果该显示仅有一个窗口，那么这个功能将对其进行划分，并创建一个新的窗口，但是如果该显示已经有两个或者更多个窗口，那么不进行新的划分。

尝试使用这一命令，将底部窗口中的缓冲区切换到 **experience** 缓冲区：键入 **C-x 4 C-o experience Enter**。

在窗口中进行移动

当您的 **Emacs** 会话中具有多个窗口时，不会影响光标的移动，您可以在当前活动窗口中正常地移动，就如同在您的会话中只有一个窗口一样。并且当您滚动这个窗口时（使用滚动条，或者使用各种用于滚动的键），其他的窗口都不会滚动，即使这些窗口显示了相同缓冲区的副本。

要滚动另一个窗口，而不是光标所在的窗口，可以使用 **scroll-other-window** 功能，它与 **C-M-v** 进行了绑定。（**Emacs** 将您所有的窗口保存在一个排序的列表中，所以如果您打开了两个以上的窗口，这一命令将滚动列表中的下一个窗口。）

现在，尝试键入 **C-M-v**，以滚动下面的窗口（该窗口中包含 **experience** 缓冲区），直到看到标题“A Little Boy Lost”为止。

通过启用 **scroll-all** 模式，您可以同时滚动所有的打开了相同缓冲区的窗口。这一功能是可以进行切换的，当它处于活动状态时，模式行将显示 ***SL***。它将相同缓冲区的所有窗口一同移动，即使这些窗口显示了缓冲区中完全不同的部分，即将您键入的滚动命令应用于包含该缓冲区的所有窗口。

现在，请尝试下面的操作：

键入 **C-x b experience Enter**，以便在顶部窗口中切换到 **experience** 缓冲区。

通过键入 **M-x scroll-all-mode** 打开 **scroll-all** 模式。

当顶部的窗口仍然处于活动状态时，通过键入 **PgDn**，同时滚动两个窗口。

通过按向上箭头键，在两个窗口中向上移动光标；多次按向上箭头键，以便该缓冲区的顶部出现在顶部窗口中，继续按向上箭头键，这时可以观察到光标在底部窗口中移动，而不在顶部窗口中移动。

通过多次按向下箭头键，在两个窗口中向下移动光标。

移动到另一个窗口

要在两个窗口之间移动，可以运行 **other-window** 功能，**C-x o**，它将移动到下一个窗口。当您重复地运行该命令时，它循环地遍历所有的窗口。当您移动到另一个窗口时，将在该缓冲区的当前位置绘制光标。

通过使用 **windmove** 命令直接指定光标，您也可以将光标移动到其他窗口，如表 1 所述。

表 1. Emacs 窗口移动命令汇总

功能	描述
windmove-up	移动到正好位于当前窗口上方的窗口，如果该窗口存在的话。
windmove-down	移动到正好位于当前窗口下方的窗口，如果该窗口存在的话。
windmove-left	移动到正好位于当前窗口左边的窗口，如果该窗口存在的话。
windmove-right	移动到正好位于当前窗口右边的窗口，如果该窗口存在的话。

现在，请尝试下面的操作：

键入 **C-x o**，以移动到底部的窗口。

关闭 **scroll-all** 模式：键入 **M-x scroll-all-mode**，然后再次向下滚动到“A Little Boy Lost”，请注意顶部的窗口不再滚动。

通过键入 **C-x o**，移回到顶部的窗口。

使用 **C-x 2**，对这个窗口进行划分。

通过键入 **M-x windmove-down**，向下移动到您刚刚划分的窗口。

删除窗口

有几种方式可以删除 Emacs 窗口。

要删除当前窗口（即活动光标当前所在的窗口），可以运行 **delete-window** 功能，它与 **C-x 0** 进行了绑定。

现在，尝试删除您刚刚创建的第三个窗口：键入 **C-x 0**。

要删除除了当前窗口之外所有的窗口，可以运行 **delete-other-windows** 功能，它与 **C-x 1** 进行了绑定。

尝试下面的操作：键入 **C-x 1**。

当您删除一个窗口时，您并不会关闭它所显示的缓冲区，该缓冲区在 Emacs 中仍然保持打开。要删除当前窗口，并且同时关闭其缓冲区，可以运行 **kill-buffer-and-window** 功能，它与 **C-x 4 0** 进行了绑定。

尝试下面的操作：

垂直地将窗口划分为两半：键入 **C-x 2**，以便在您的 Emacs 会话中得到两个窗口。

将光标移动到这首诗的标题中的 **BOY** 和 **LOST** 之间的空格处，并键入 **M-t**，以调换这两个单词。

将当前缓冲区 **experience** 写入到一个名为 **new.experience** 的新的文件：键入 **C-x C-w new.experience**。

使用 **C-x 4 0** 关闭该缓冲区和窗口。键入 **y** 以进行证实。

当您进行这一操作时，关闭了 **new.experience** 缓冲区，并且您的 Emacs 会话再次只包含一个窗口。

水平地划分一个窗口

对应于刚刚描述的 **split-window-vertically** 功能的是 **split-window-horizontally**，它与 **C-x 3** 进行了绑定。这一功能将当前 Emacs 窗口从中间划分为两半，并将这两个新的窗口沿水平方向排成一行，因此它们相互紧靠在一起。对于并列地查看类似的缓冲区，这种方法是特别合适的。尝试下面的操作：

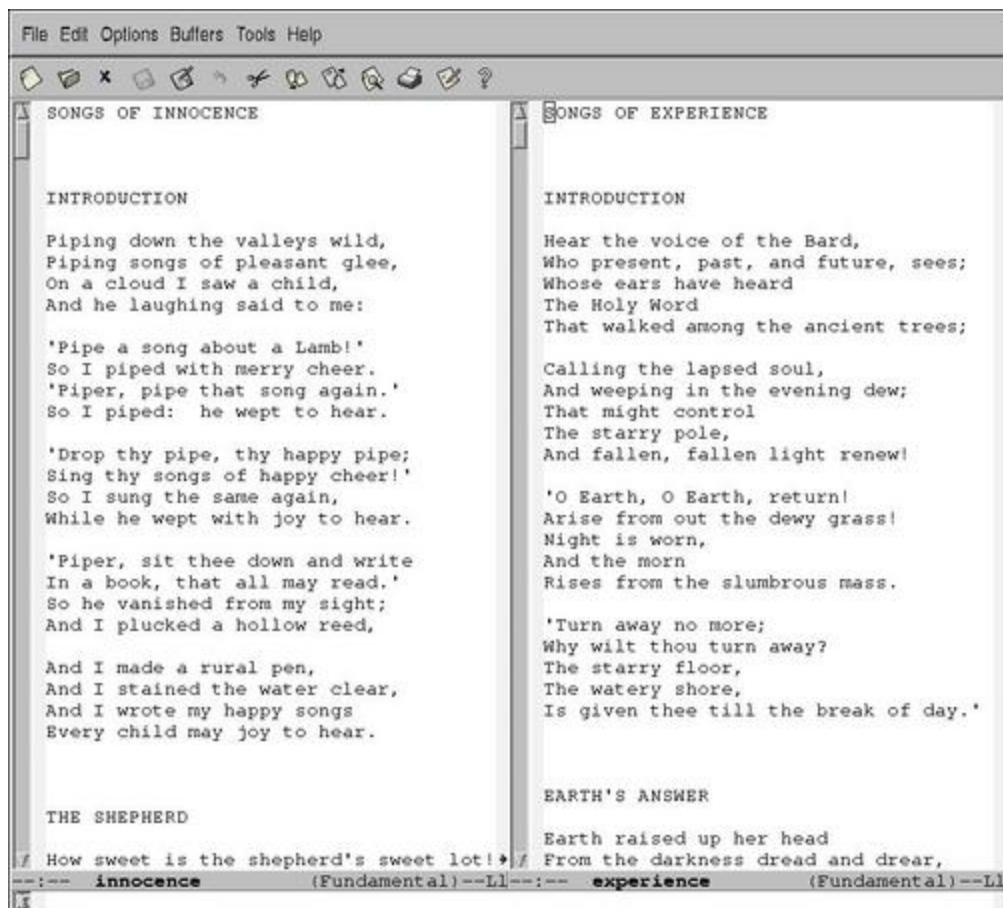
在一个新的缓冲区中打开 **experience** 文件：键入 **C-x C-f experience Enter**。

键入 **C-x 3**，以水平方向将该窗口划分为两半。

键入 **C-x b Enter**，以便将最左边窗口的内容更改为 **innocence** 缓冲区。

您的 Emacs 会话应如 图 3 所示。

图 3. 在水平 Emacs 窗口中的示例文件



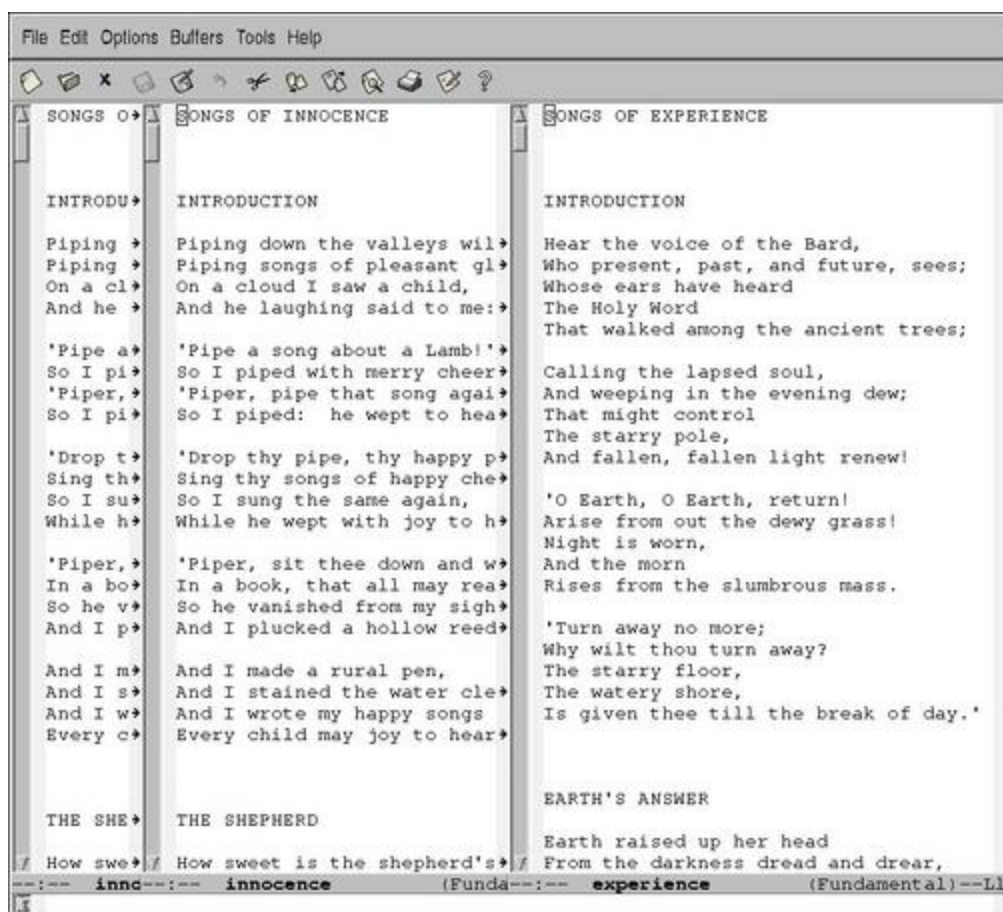
与垂直划分一样，您可以在这个功能的前面使用一个数值参数以指定左边 窗口应该具有的字符数目宽度；如果您给出了一个负数，那么它将指定右边 窗口的宽度。

尝试下面的操作：通过键入 **C-u 11 C-x 3**，以水平方向将当前窗口划分为两半，并使最左边的窗口具有 11 个字符宽度。

请注意，您给出的数值包括用于最左边窗口的滚动条的一个字符、用于滚动条和缓冲区的内容之间的空白列的一个字符、缓冲区本身的列、用于箭头图形（指示一行内容是否超出了它的显示宽

度) 的一列, 以及该划分生成的第二个窗口的滚动条。在这一示例中, 您刚刚创建的最左边的窗口具有可显示七个字符的宽度, 如图 4 中所示。

图 4. Emacs 中的多个水平窗口



与相同缓冲区的垂直划分一样, 可以分别地处理相同缓冲区的任何水平划分。在一个窗口中的滚动操作和光标移动不会影响其他窗口, 所以当您在一个窗口中移动光标时, 它不会影响任何包含相同缓冲区的其他窗口。但是您可以使用另一个技巧。如果您打开了 **follow-mode**, 那么您就可以根据相同缓冲区的多个窗口生成一个较大的“虚拟”窗口。

通过这种方式, 当您窗口进行水平划分时, 这些窗口将显示该缓冲区中不同的、但又相连的部分, 从而为该缓冲区生成一个较大的虚拟窗口。将光标移动到一个窗口的末尾, 这样可以使其进入到一个窗口的开始处。

尝试下面的操作:

键入 **C-x 1** 以删除所有的窗口, 并键入 **C-x 3**, 以便水平地将单个窗口划分为两半。现在您的 Emacs 会话包含两个窗口, 每个窗口都包含了 **innocence** 缓冲区, 并且每个窗口的光标都位于缓冲区的顶部。

键入 **M-x follow-mode** 为该缓冲区启用这种模式。请注意, 单词 **Follow** 出现在两个窗口的模式行中。第二个窗口的内容发生了更改: 现在, 它恰好接着左边窗口中的内容, 以该窗口的下一行内容作为开始。

通过按向下箭头键, 在左边的窗口中, 将光标在缓冲区中向下移动, 并观察当到达该窗口的底部时所发生的情况: 光标将移动到右边窗口的顶部。

在右边的窗口中向下移动光标, 并观察如何重绘这两个窗口。使用向上箭头键, 在右边的窗口中向上移动光标, 并观察当到达该窗口顶部时所发生的情况: 光标移动到左边窗口的底部。

这个功能是可以进行切换的。再次运行它, 就可以在这个缓冲区中关闭它: 键入 **M-x follow-mode**, 并注意 **Follow** 不再出现在模式行中。

尽管 **follow-mode** 同样可以工作于垂直划分的缓冲区中，但是在大多数情况下，使用这一模式没有任何优势。

调整窗口大小

到目前为止，您要么为窗口使用缺省大小，要么在创建它们时指定它们的大小。但是您可以在任何时候调整任何 **Emacs** 窗口的大小。

要使当前窗口变得更高些，可以运行 **enlarge-window** 功能，它与 **C-x ^** 进行了绑定。要使它变得更矮些，可以在这一功能之前使用 **M--**。要按照特定的行数缩小或者增大窗口，可以给出具体的数值（使用通用的参数 **C-u**）。

您还可以更改窗口的宽度。要使当前窗口变得更窄，可以运行 **shrink-window-horizontally** 功能，它与 **C-x {** 进行了绑定。要使它变得更宽，可以运行 **enlarge-window-horizontally** 功能，它与 **C-x }** 进行了绑定。

尝试确定您的 **Emacs** 会话中某些窗口的形状：

键入 **C-x 1** 以关闭任何现有的窗口，并键入 **C-x b innocence Enter**，以便在这一窗口中打开 **innocence** 缓冲区。

使用 **C-x 3** 水平地将这个窗口划分为两半。

通过键入 **C-x 4 C-o experience Enter**，在最右边的窗口中切换缓冲区。

通过键入 **C-x 2 C-x o C-x b experience Enter**，垂直地将最左边的窗口划分为两半，并使 **experience** 缓冲区处于底部的窗口中。

通过键入 **C-s sunf** 搜索标题，将光标移动到“Ah, Sunflower”诗的开头，然后键入 **C-u 6 C-l** 以重绘该窗口，并使这首诗呈现在眼前。

通过键入 **M-- M-5 C-x ^ C-l**，将这一窗口收缩五行，并重绘它。

因为窗口的宽度不足以显示它们，所以隐藏了一些行。通过键入 **C-x } C-x }** 使该窗口增加两个字符的宽度。

如果在调整大小时，您将该窗口设置得过小，以至于其中的行数不足以绘制窗口及其模式行，那么将关闭该窗口。尝试下面的操作：

移动到这个窗口上方的窗口：键入 **windmove-up**。

键入 **C-u -5 C-x ^** 将该窗口收缩五行。

多次尝试这一操作：键入 **C-u -5 C-x ^**，并再次键入它以继续收缩这一窗口。继续进行这一操作直到该窗口消失。

您可以进行更多调整。要缩小该窗口的大小（如果可能的话），可以运行 **shrink-window-if-larger-than-buffer** 功能，它与 **C-x -** 进行了绑定。通过运行 **balance-windows** 功能，您还可以平衡所有可见窗口的大小。它使得所有的窗口都具有大致相同的尺寸。它与 **C-x +** 进行了绑定。

尝试下面的操作：键入 **C-x 2 C-x 2** 以生成更多的窗口，并键入 **C-x + C-x +** 以平衡它们的高度。

除了使用这些命令之外，您还可以使用鼠标调整窗口大小，正如您将在“使用鼠标调整窗口大小”部分中学习到的。

比较两个窗口

通常，可以使用多个窗口以比较缓冲区的内容。**compare-windows** 功能允许您完成这项任务：它可以比较两个窗口中的文本，从两个窗口中光标处的字符开始，并将光标移动（在两个窗口中）到第一个不同的字符处。如果这两个文件是完全一样的，那么将光标移动到这两个缓冲区的末尾。

现在，尝试针对名为 **experience** 的示例文件和您以前编辑的 **new.experience** 文件运行它：

通过键入 **C-x C-c** 退出 **Emacs**，然后使用一个文件启动它：

```
$ emacs experience
```

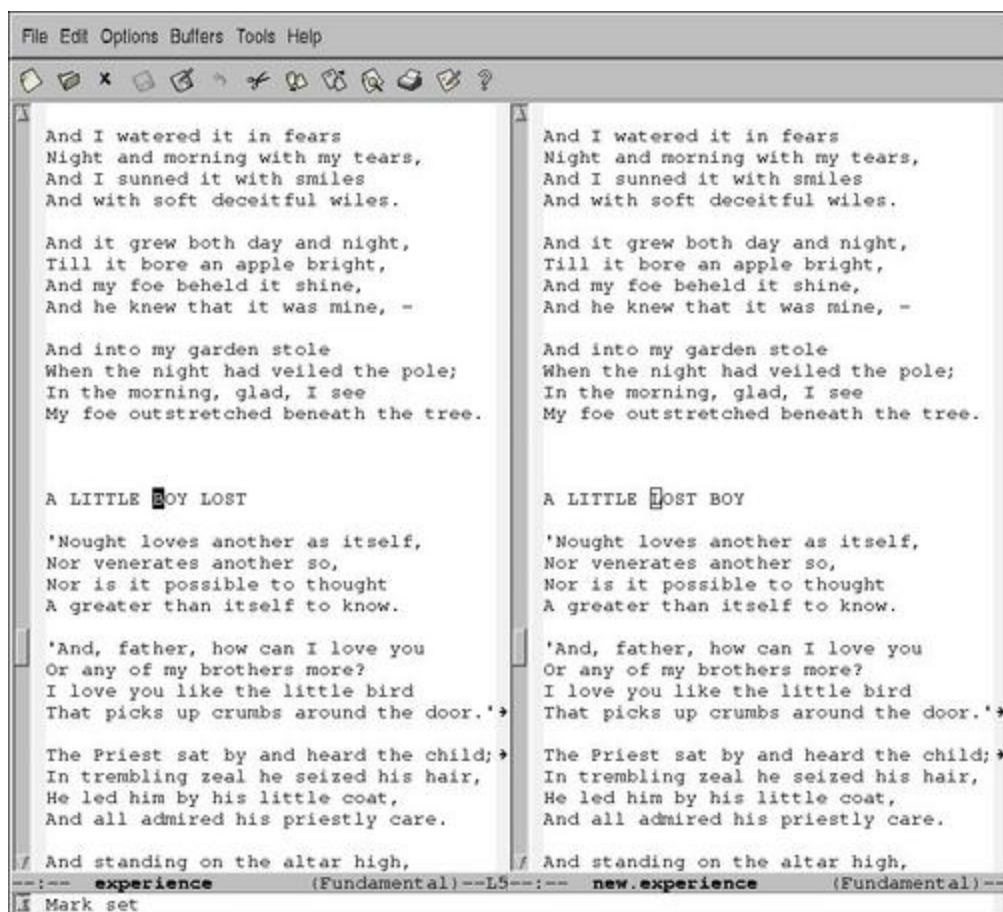
键入 **C-x f new.experience**，以便在一个新的缓冲区中打开您编辑过的文件。

通过键入 `C-x 3 C-x b Enter`，垂直地划分屏幕，并使得每个缓冲区都位于它自己的窗口中。

键入 `M-x compare-windows` 以运行比较操作。

两个窗口中的光标移动到文件中第一个不同的位置：这首诗的标题（您在前面对其进行了更改）。您的 Emacs 会话应该与图 5 中所示类似。

图 5. 比较两个 Emacs 窗口的内容



Emacs 窗口命令表

表 2 中包含了用于操作窗口的常见 Emacs 命令的列表，给出了它们的键绑定（如果适用的话），并描述了它们的功能。

表 2. 常见的 Emacs 窗口操作命令

功能	绑定	描述
split-window-vertically	C-x 2	从中间将当前窗口划分为两半，垂直地堆叠新的缓冲区。
switch-to-buffer-other-window	C-x 4 b	垂直地将当前窗口划分为两半，提示输入缓冲区以使用底部的窗口，并将其作为活动窗口。
display-buffer	C-x 4 C-o	在另一个窗口中显示一个缓冲区，提示输入缓冲区以使用另一个窗口，但保持当前窗口为活动窗口。（如果仅存在一个窗口，那么垂直地划分该窗口以显示另一个缓冲区。）
find-file-other-window	C-x 4 f	在新的缓冲区中打开新的文件，在新的垂直窗口中绘制它。
find-file-read-only-other-window	C-x 4 r	在一个新的只读缓冲区中打开新的文件，在新的垂直窗口中绘制它。
scroll-other-window	C-M-v	滚动到下一个由 C-x o 切换到的窗口。

scroll-all		切换 scroll-all 次要模式。当它处于打开状态时，将同时滚动显示当前窗口中的缓冲区的所有窗口，并滚动均等的相应距离。
other-window	C-x o	将光标移动到下一个窗口，并使其成为活动窗口。
windmove-up		移动到恰好位于当前窗口上方的窗口，如果它存在的话。
windmove-down		移动到恰好位于当前窗口下方的窗口，如果它存在的话。
windmove-left		移动到恰好位于当前窗口左边的窗口，如果它存在的话。
windmove-right		移动到恰好位于当前窗口右边的窗口，如果它存在的话。
delete-window	C-x 0	删除当前窗口，并将光标移动到使用 C-x o 将切换到的下一个窗口。
delete-other-windows	C-x 1	删除当前窗口之外的所有窗口。
kill-buffer-and-window	C-x 4 0	删除当前窗口，并剪切它的缓冲区。
split-window-horizontally	C-x 3	将当前窗口从中间划分为两半，水平地堆叠新的缓冲区。
follow-mode		切换 follow 次要模式。当它在缓冲区中处于打开状态时，将所有显示该缓冲区的窗口连接为一个较大的虚拟窗口。
enlarge-window	C-x ^	使当前窗口增加一行的高度；在其之前使用一个负数，将使得当前窗口减少一行的高度。
shrink-window-horizontally	C-x }	使当前活动窗口减少一列的宽度。
enlarge-window-horizontally	C-x {	使当前活动窗口增加一列的宽度。
shrink-window-if-larger-than-buffer	C-x -	将当前活动窗口的大小缩小到对于它所包含的缓冲区来说可能的最小尺寸。
balance-windows	C-x +	平衡所有窗口的尺寸，使它们的大小大致相等。
compare-windows		将当前窗口与下一个窗口进行比较，在两个窗口中从光标处开始比较，并在两个缓冲区中将光标移动到第一个不同的字符处，直到到达缓冲区的末尾为止。

移动并操作 Emacs 框架

在 X 中，通常将一个在它自己的 X 客户端窗口中运行的应用程序称为一个窗口。但是因为 Emacs 对于单词窗口有它自己的定义，正如在前面的部分中所描述的，Emacs 为整个 Emacs X 客户端窗口使用了另一个术语：称其为框架。

Emacs 支持为相同的 Emacs 会话打开多个框架。当您在多个框架中打开一个缓冲区时，更改将出现在所有框架的缓冲区中。关闭一个框架并不会影响到其他框架，但使用通常的 **save-buffers-kill-emacs** 功能 (C-x C-c) 退出一个框架，将保存所有框架的缓冲区，并退出所有的框架。

如果您处于一个控制台窗口中，那么这些命令仍然可以起作用，尽管一个控制台一次只能显示一个框架。在控制台中，通过给定的框架编号区分各个框架，该编号出现在模式行中（并且前面使用了一个 F 字符）以区分每个框架。

生成新的框架

make-frame-command 功能, **C-x 5 2**, 可以生成一个新的框架, 并使其处于活动状态:

通过键入 **C-x C-c** 退出 Emacs, 然后使用一个文件 (您创建的新的文件) 启动它:

```
$ emacs new.experience
```

键入 **C-x 5 2** 以生成一个新的框架。(它出现在您桌面上的确切位置, 取决于您的窗口管理器。) 它也包含了 **new.experience** 缓冲区的一个副本。

编辑一行: 键入 **C-u 381 C-n M-f M-f M-f M-t**。对于两个框架中的这个缓冲区, 将在模式行中出现星号。

C-x 4 命令用于 Emacs 窗口, **C-x 5** 命令用于 Emacs 框架。对于用于生成新窗口的所有 **C-x 4** 命令, 正如在垂直地划分一个窗口部分中所描述的, 都存在 **C-x 5** 等价操作: 例如, **switch-to-buffer-other-frame** 功能是 **C-x 5 b**。

尝试下面的操作:

键入 **C-x C-f experience Enter**, 以便在一个新的缓冲区中打开 **experience** 文件的副本。

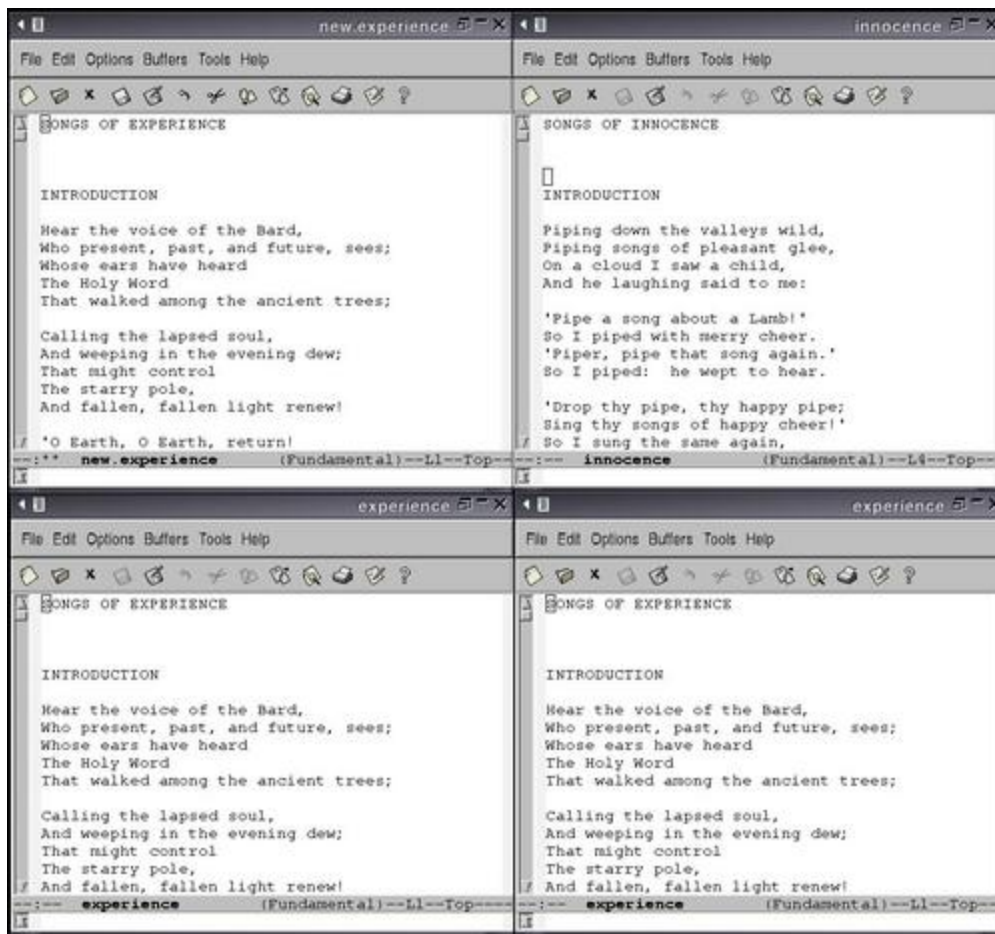
键入 **C-x 5 b experience Enter**, 以便在一个新的框架中打开这个缓冲区的副本。

find-file-other-frame 功能, 它与 **C-x 5 f** 进行了绑定, 提示输入一个文件名, 并在新的框架中打开给定的文件。同样地, **find-file-read-only-other-frame**, 它与 **C-x 5 r** 进行了绑定, 在新的框架中打开给定的文件, 并将其作为一个只读缓冲区。

尝试使用单步操作查找一个文件, 并在一个新的框架中打开它, : 键入 **C-x 5 f innocence Enter**。

现在您的 Emacs 会话应该与图 6 所示类似, 其中包括四个不同的 X 客户端窗口。

图 6. 运行多个 Emacs 框架



在框架之间移动

使用 **other-frame** 功能，**C-x 5 o**，可以在框架之间进行移动。

与它的 **Emacs** 窗口等价操作 **C-x o** 一样，这个功能在所有的当前 **Emacs** 框架之间循环，提升每个框架，以使其获得焦点并位于任何其他窗口之上，且选择其作为当前框架。

尝试在已经打开的四个框架之间循环：**C-x 5 o C-x 5 o C-x 5 o C-x 5 o C-x 5 o**。

删除框架

如果您使用 **X** 控件以关闭一个框架，或者使用 **C-x C-c** 命令以退出 **Emacs**，那么您将退出所生成的所有框架。然而，用以销毁一个框架的 **X** 控件，仅仅销毁特定的框架；它既不会关闭在该框架中打开的任何缓冲区，也不会销毁其他 **Emacs** 框架。

您还可以从 **Emacs** 删除框架。要删除当前框架，可以使用 **delete-frame** 功能，**C-x 5 0**。在删除了当前框架之后，将使下一个框架成为活动框架和当前框架。如果您尝试在会话中仅有的框架上运行这一操作，那么 **Emacs** 将发出蜂鸣声，并报告一个错误。现在尝试这一操作，以删除您所在的框架。

要删除除了当前框架之外的所有框架，可以使用 **delete-other-frames** 功能，**C-x 5 1**。将删除当前框架之外的所有框架，如果它们存在的话。现在尝试这一操作以删除剩余的两个框架，以便回到一个 **Emacs** 框架。

请注意，这些命令并不会关闭缓冲区，在完成删除操作之后，在所删除框架的窗口中显示的任何缓冲区仍然可用于当前框架。

图标化框架

在控制台中，**C-z** 通常将 **Emacs** 挂起到后台；在 **X** 中，它将运行 **iconify-or-deiconify-frame** 功能。这一操作将图标化当前框架；但是如果已经图标化了当前框架，它将取消该框架的图标化。

尝试下面的操作：在您已经打开的当前框架中键入 **C-z**。根据您正在运行的 **X** 的版本和窗口管理器，以及您系统中的桌面软件，应该对 **Emacs** 窗口进行图标化。然后，对图标化的框架再次按 **C-z**，以取消它的图标化，并使得图标化的框架重新获得焦点。

Emacs 框架命令表

表 3 中包含了用于操作框架的常见 **Emacs** 命令列表，给出了它们的功能名及其缺省的键绑定（如果适用的话），并描述了它们的功能。

表 3. 常见的 **Emacs** 框架操作命令

功能	绑定	描述
make-frame-command	C-x 5 2	生成一个新的 Emacs 框架，并使其成为活动框架。
switch-to-buffer-other-frame	C-x 5 b	在另一个框架中打开指定的缓冲区。如果不存在其他框架，则创建一个新的框架。
find-file-other-frame	C-x 5 f	在另一个框架中打开指定的文件。如果不存在其他框架，则创建一个新的框架。
find-file-read-only-other-frame	C-x 5 r	在另一个框架中的只读缓冲区中打开指定的文件。如果不存在其他框架，则创建一个新的框架。
other-frame	C-x 5 o	移动到下一个框架，并使其成为活动框架。
delete-frame	C-x 5 0	删除当前框架，并使下一个框架成为活动框架。
delete-other-frames	C-x 5 1	删除当前框架之外的所有框架。
iconify-or-deiconify-frame	C-z	图标化当前框架。如果该框架已经图标化了，那么取消它的图标化。（在控制台中，这个绑定将挂起 Emacs 。）

Emacs 窗口和鼠标

在 **Emacs** 窗口中使用鼠标，有一些非常有价值的技术。

在模式行中使用鼠标

您可以使用鼠标执行在对您的 **Emacs** 会话进行划分和分区部分中所描述的许多功能。这些窗口操作可以在模式行中完成。

在 **Emacs** 的新版本中，模式行的特定区域具有它们自己的鼠标绑定。例如，如果您的模式行显示了您是否有邮件（它通过书写单词 **Mail** 来完成这一任务），然后以某种方式使用鼠标在模式行中单击该单词，将您的邮件启动一个新的缓冲区；单击模式行中该缓冲区的名称，会将该窗口中的缓冲区切换到缓冲区列表中的下一个缓冲区。当您鼠标位于模式行中特定区域的上方时，将使用一个弹出框显示相应的内容，这些内容称为工具提示。除了这些特殊的工具提示之外，下面的命令也可以用于模式行中的任何位置。

使用鼠标划分窗口

在模式行上以 **C-B2** 组合方式进行单击，将在您单击的位置对窗口进行水平地划分。如果您太靠近窗口的上下边缘，那么该划分是可能的最小尺寸。（通过按住 **Ctrl** 键，然后单击中间的鼠标按钮，就可以输入这个组合。）

在滚动条上输入 **C-B2** 组合，将在您单击鼠标的位置垂直地划分窗口。如果您太靠近窗口的上下边缘，那么该划分是最小尺寸。（请注意，当前这一操作并不使用实现滚动条的某些 **X** 工具包。）

尝试使用鼠标划分窗口：

如果 **Emacs** 正在运行的话，通过 **C-x C-c** 退出它。使用示例文件之一，再次启动它：



```
$ emacs innocence
```

在一个新的垂直窗口中打开第二个示例文件：键入 **C-x 4 f experience Enter** 。

您的 **Emacs** 会话应该与图 1 中所示类似。尝试在顶部模式行的中间（即恰好位于单词 **Top** 之后的某处）使用 **C-B2** 鼠标组合，以便在该位置处水平地划分顶部的窗口。

在模式行上大约相同的位置使用 **C-B2** 鼠标组合，水平地划分底部的窗口，同样是在恰好位于单词 **Top** 之后的某处。

在您刚刚创建的两个新窗口的滚动条下面的小垂直条上单击并拖动 鼠标左键，以对其进行调整，使得所有四个窗口具有大致相同的大小。如果您的 **Emacs** 框架太小，而不能显示所有四个窗口中的文本，那么您可以使用窗口管理器控件以调整 **X** 客户端框架的大小。

使用鼠标移动到其他窗口

通过在您希望移动到的窗口的模式行的空白区域上单击 鼠标左键（第一个鼠标按钮），您可以使用鼠标移动到任何 **Emacs** 窗口。（在 **Emacs** 的新版本中，单击缓冲区名可以将该窗口中的可见缓冲区更改为缓冲区列表中的下一个缓冲区。）您单击其模式行的窗口将成为活动窗口，并且将活动光标移动到了该窗口中的当前位置。

您还可以使用鼠标移动到另一个窗口中可见部分的任何位置：要完成这一任务，在该窗口中单击 鼠标左键。光标将移动到您单击的位置。

尝试在您刚刚创建四个窗口之间移动：

通过在其模式行的空白区域单击 鼠标左键，移动到右上方的窗口，然后键入 **C-u 349 C-n** 在该文件中向下移动。

通过在可见缓冲区窗口中的任何字符上单击 鼠标左键，移回到左上方的窗口，然后键入 **C-s Till C-s C-s C-a**，以便在缓冲区中的其他位置重新聚焦该显示。

暂停鼠标操作，并通过键入 **C-x o C-x o** 移动到左下方的窗口，正如您在移动到另一个窗口部分中所学习的。键入 **C-u 286 C-n**，以便在该缓冲区中向下移动。

在右下方窗口模式行中的缓冲区名称之后的某处，单击 鼠标左键，以使其成为活动窗口。键入 **C-s jour C-l**，以便将这个窗口中的视图移动到缓冲区中的其他位置。

使用鼠标调整窗口大小

您还可以在模式行上使用鼠标调整窗口大小。对于您先前在调整窗口大小部分中所学习的方法而言，这是一个替代方法。

要使窗口变得更高或者更矮，可以使用 鼠标左键 单击并拖动模式行。要使窗口变得更窄或者更宽，可以单击并拖动其滚动条下方的小垂直条。

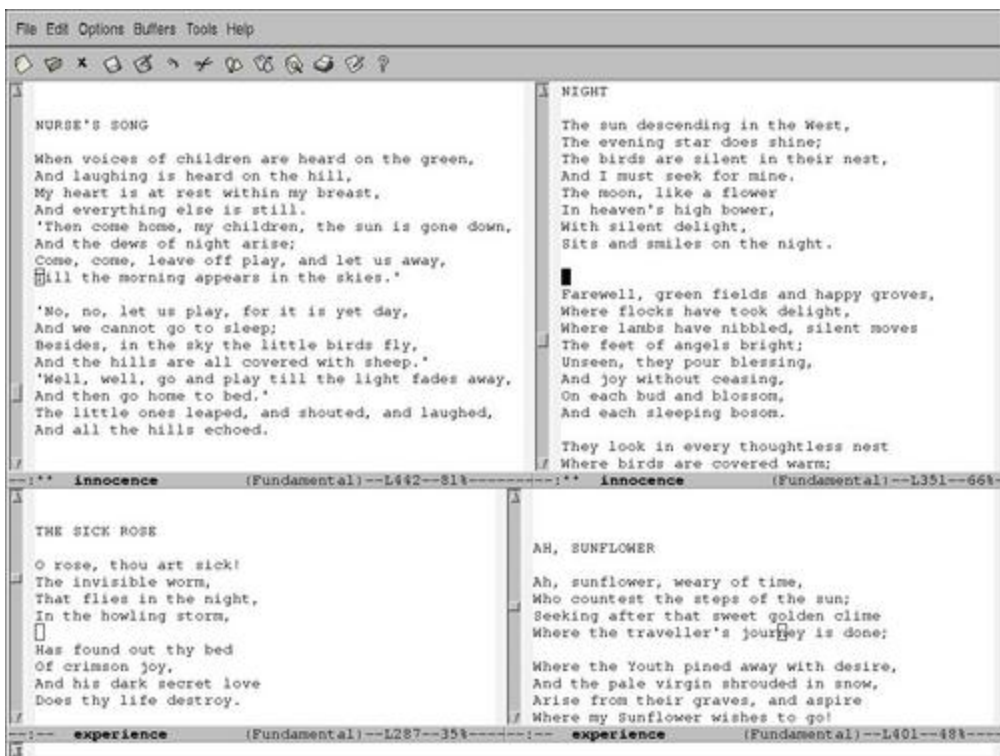
尝试使用鼠标调整窗口大小：

单击并稍微拖动 鼠标左键 以及右上方窗口滚动条下面的小垂直条，从而使得右上方窗口的宽度缩小几个字符，并使左上方窗口变得足够大，以便能够在这两个缓冲区的窗口中显示每行所有的字符。（您可能还需要使用窗口管理器的控件，以增大 Emacs 框架。）

通过在模式行上单击并拖动 鼠标左键 使底部的两个窗口变得更小，从而使底部的窗口收缩几行。

如果您按照这一部分中的所有步骤进行了操作，那么您的 Emacs 会话应该与图 7 所示类似。

图 7. 使用鼠标生成 Emacs 窗口并调整其大小



使用鼠标删除窗口

要删除一个窗口，可以在它的模式行上单击 鼠标右键。在模式行的空白区域上单击 鼠标中键，以关闭所有 其他窗口，并扩大该窗口，使其填满整个框架。

去掉一些窗口：

在左上方窗口的模式行上单击 鼠标右键，以删除该窗口。右上方的窗口向左边扩大以填满这个空间；现在这个窗口成为了活动窗口。

在它的模式行上单击 鼠标中键，以删除它下面的两个窗口，并使这个上面的窗口填满整个 Emacs 框架。

鼠标躲避模式

有时在 X 中，鼠标指针可能会妨碍您的 Emacs 会话，比如您选择了一个 Emacs 框架并开始进行编辑，并且鼠标指针悬停于某处文本的上方。

如果您觉得这种情况很讨厌，那么您可以使用 mouse-avoidance-mode 以更改鼠标指针的行为。这个模式提供了几种技术，如表 4 中所述。

表 4. Emacs 鼠标指针躲避的类型

模式	描述
<code>animate</code>	只要光标靠近鼠标指针，就使得鼠标指针迅速地移动到该框架中的一个随机位置。
<code>banish</code>	一旦您开始键入，就将鼠标指针驱逐到该窗口的右上角。
<code>cat-and-mouse</code>	用作 <code>animate</code> 的同义词。
<code>exile</code>	仅当鼠标指针离光标太近时，将鼠标指针移动到该窗口的右上角（如同 <code>banish</code> ）。一旦光标移开，将鼠标指针移回到它的原始位置。
<code>jump</code>	当光标靠近鼠标指针时，使鼠标指针立刻跳到该框架中的一个随机位置。
<code>none</code>	不提供鼠标躲避功能（缺省情况）。
<code>proteus</code>	与 <code>animate</code> 中一样移动鼠标指针，但是就像希腊神话中的海神普罗特斯，更改鼠标指针的形状（使用一个随机图像字符）。

最好使用一个实例加以说明。现在，请尝试下面的操作：

键入 `M-x mouse-avoidance-mode` `Enter` `cat-and-mouse` `Enter` 以打开鼠标躲避模式。

移动到临时缓冲区：键入 `C-x b *scratch*` `Enter` 。

移动鼠标指针，使其位于光标的左边，且与光标仅有几英寸的距离，并开始键入一行文本：“**When the cat wants to play the mouse runs away**”。一旦光标靠近鼠标指针，鼠标指针则应该迅速离开。

总结

您已经了解了对 **Emacs** 会话形状进行操作的细节技术，即如何创建并控制多个框架、如何对任何框架进行划分和分区以得到多个窗口，以及如何遍历这些框架和窗口。您还了解了一些其他技巧，包括如何使用鼠标操作框架和窗口以及如何处理鼠标指针，并且通过这些优秀的、功能强大的技术，可以使得 **Emacs** 按照您所希望的方式进行工作，并具有您所希望的外观。

第 6 部分：自定义您的 Emacs 环境

设置 Emacs 变量

要对您的 **Emacs** 环境中的某些方面进行自定义，最快捷且最简单的方式就是设置 **Emacs** 变量。变量可以影响 **Emacs** 中各个方面的工作方式；它们是 **Emacs Lisp** 符号，通过名称对其进行引用，并且通常具有一个缺省值。有些变量可以包含数值或者字符串，而其他的变量则是“真”或者“假”：如果将一个变量的值设置为 `nil`，那么它的值是“假”，值 `t` 表示“真”。

在 **Emacs** 会话中，您可以在任何时间更改任何变量的值。进行这种操作，可以为特定的缓冲区设置局部变量，可以更改仅应用于当前会话的全局变量，或者测试不同值的行为。

例如，在本系列文章的第 2 部分教程（请参见[参考资料](#)）中，您学习了 `fill-paragraph` 函数（绑定于 `M-q`），该函数在光标处填充段落。`fill-column` 变量中包含了这个函数将要填充的宽度，和其他的 **Emacs** 变量一样，您可以显示其内容并对其进行更改。

首先，使用一个新文件的缓冲区启动 Emacs：

```
$ emacs bee
```

现在，在您的新缓冲区中插入一行文本：

```
The busy bee has no time for sorrow.
```

通过键入 M-q 运行 fill-paragraph 函数，以便填充当前段落（即您刚刚插入的行）。

当您键入这个内容的时候，看起来似乎没有出现什么变化，这是因为 Emacs 用于在缓冲区中填充文本的缺省宽度（以字符为单位），比您刚刚键入的段落要长。您可以通过观察 fill-width 变量的内容来进行证实。

显示变量的值

使用 describe-variable 函数以得到变量的值。当您运行这个函数的时候，Emacs 将打开一个新窗口，该窗口中包含该变量的给定当前值。如果有的话，还将提供关于该变量的描述。这个函数与 C-h v 按键进行了绑定。

通过键入 C-h v fill-width，可以尝试获得 fill-width 变量的值。

在完成了这个操作之后，您将在 Emacs 会话（包含[清单 1](#) 的文本）中看到一个新的窗口。

清单 1. 显示 fill-width 变量值的 Emacs describe-variable 函数的输出

```
fill-column's value is 70
```

```
Documentation:
```

```
*Column beyond which automatic line-wrapping should happen.  
Automatically becomes buffer-local when set in any fashion.
```

```
You can customize this variable.
```

从这个输出中您可以看到，直到到达第 70 个字符，fill-paragraph 函数才划分出一个新的行，所以您所键入的简短段落的长度并不足以进行自动换行。

设置变量

与任何 Emacs 变量一样，您可以更改 fill-column 的值。可以使用 set-variable 函数来完成这个任务，当您运行该函数的时候，它会请求输入变量的名称和您希望给它赋的值。

尝试将 fill-column 的值设置为 10，如[清单 2](#) 中所示。

清单 2. 使用 Emacs set-variable 函数更改 fill-column 变量的值

```
M-x set-variable Enter
```

```
Set variable: fill-column Enter
```

```
Set fill-column to value: 10 Enter
```


现在，再次填充该段落：确保光标位于您刚刚键入的行的末尾处，然后键入 `M-q` 以运行 `fill-paragraph` 函数。请注意，如何填充该段落以使得其中所有的行都不超过 10 个字符，如[清单 3](#) 中所示。

清单 3. 在将 `fill-width` 变量设置为 10 之后进行自动换行的示例段落

```
The busy
bee has no
time for
sorrow.
```

在将 `fill-column` 的值设置为 4 之后，再次尝试这个操作，使用[清单 2](#) 中给定的过程。在使用另一个 `M-q` 填充该段落之后，您将看到这些单词进行了不同的自动换行，如[清单 4](#) 中所示。

清单 4. 在将 `fill-width` 变量设置为 4 之后进行自动换行的示例段落

```
The
busy
bee
has
no
time
for
sorrow.
```

因为 `fill-column` 是一个局部变量，所以您对它做的任何更改都仅应用于当前缓冲区。在一个 Emacs 会话中，每个缓冲区都有它自己的 `fill-column` 变量；这些变量中的任何一个都可以设置为不同的值。

尝试选择您所插入的所有文本，并将其粘贴到一个新的缓冲区：

```
C-space M-< M-w C-x b cee Enter C-y
```

在键入了该内容之后，您应该拥有一个刚填充的段落的副本（从 `bee` 缓冲区复制到新的、名为 `cee` 的缓冲区，并且尚未进行保存）。

键入 `M-q` 以填充它。填充的内容仅占用一行，看起来与您在开始时键入的单行内容相同。

通过请求 Emacs 显示变量的内容，您还可以验证在两个缓冲区中 `fill-column` 的值是不同的。

键入 `C-h v fill-column Enter` 以显示这个缓冲区的 `fill-column` 变量的值，它应该为 70。

在完成了这个操作之后，请求 Emacs 显示 `bee` 缓冲区的 `fill-column` 变量的值：键入 `C-x b Enter C-h v fill-column Enter` 以验证它仍然设置为 4。如[图 1](#) 中所示，其中所显示的 `*Help*` 缓冲区同样可以告诉您这个变量的局部值与全局值是不同的。

图 1. 在缓冲区内对 Emacs 的 fill-column 变量进行了局部更改之后，显示它的值



使用并了解一些常见的 Emacs 变量

Emacs 提供了许多影响并改变其行为的变量。[表 1](#) 列出了一些经常进行修改的变量，并描述了它们的功能。

表 1. 常见 Emacs 变量汇总

变量	描述
auto-mode-alist	如果设置为 nil，那么将关闭根据文件名扩展自动选择主要模式的功能。它的缺省值是一些文件名扩展和相应模式的列表。
auto-save-default	如果没有将其设置为 nil，那么 Emacs 则根据预设的时间间隔，自动地将经过更改的缓冲区保存到相应的文件。它的缺省值是 t。

auto-save-interval	包含调用 Auto-save 模式（如果它被设置为“真”）之后经过更改的字符的数目，其缺省值是 300 。
calendar-latitude	包含用户工作站位置的纬度值，采用度数表示；其缺省值是 <code>nil</code> 。
calendar-longitude	包含用户工作站位置的经度值，采用度数表示；其缺省值是 <code>nil</code> 。
calendar-location-name	包含用户工作站所在位置的位置名（如城市、州或省、国家/地区）的值，其缺省值是 <code>nil</code> 。
colon-double-space	如果没有将其设置为 <code>nil</code> ，那么填充文本的命令将在冒号后面插入两个空格而不是一个。其缺省值是 <code>nil</code> 。
command-line-args	包含在当前 Emacs 会话中所执行的命令行中使用的参数列表。
command-line-default-directory	包含执行当前 Emacs 会话的目录的路径名。
compare-ignore-case	如果没有将其设置为 <code>nil</code> ，那么在运行 <code>compare-windows</code> 函数的时候， Emacs 将忽略大写字母和小写字母的区别，如在本系列文章的第 5 部分教程中所描述的（请参见 参考资料 ）。其缺省值是 <code>nil</code> 。
confirm-kill-emacs	如果设置为 <code>nil</code> ，那么 Emacs 在退出的时候不请求确认；否则，可能自定义 Emacs Lisp 函数以完成退出验证工作，如 <code>y-or-n-p</code> （请参见 使得简短的回答成为可能 部分）。其缺省值是 <code>nil</code> 。
default-justification	设置缺省的对齐风格。该值可能是 <code>left</code> 、 <code>right</code> 、 <code>center</code> 、 <code>full</code> 或者 <code>none</code> 中的一个。其缺省值是 <code>left</code> 。
default-major-mode	为新的文件或者缓冲区选择缺省主要模式。其缺省值是 <code>fundamental-mode</code> 。
display-time-24hr-format	如果设置为 <code>t</code> ，那么 Emacs 将采用 24 小时军用格式来显示时间，而不是采用带有 AM 或者 PM 后缀的标准 12 小时格式。其缺省值是 <code>nil</code> 。
display-time-day-and-date	如果没有将其设置为 <code>nil</code> ，那么 Emacs 以当前星期几、当前月份和日期的格式来显示时间，而不仅仅显示小时和分钟。其缺省值是 <code>nil</code> 。
fill-column	包含各行中的列数（从此处开始填充文本到下一行）。其缺省值是 70 。
initial-major-mode	指定启动时用于 *scratch* 缓冲区的主要模式。其缺省值是 <code>lisp-interaction-mode</code> 。
inverse-video	如果没有将其设置为 <code>nil</code> ，那么 Emacs 将对显示的颜色取反（如果可能的话）。其缺省值是 <code>nil</code> 。
kill-ring	包含 Emacs 剪切环的内容，如本系列文章的第 3 部分教程中所描述的（请参见 参考资料 ）。
kill-ring-max	设置剪切环中所允许的条目数。其缺省值是 60 。
kill-whole-line	如果没有将其设置为 <code>nil</code> ，那么 <code>kill-line</code> 函数（绑定于 <code>C-k</code> ）将剪切当前行以及其尾部的换行符（如果是在该行的开头处执行这个函数）。其缺省值是 <code>nil</code> 。
make-backup-files	如果没有将其设置为 <code>nil</code> ， Emacs 将进行任何更改之前保存缓冲区

	的备份（使用相同的文件名，但在文件名后追加了波浪符（~））。
mark-ring	包含该缓冲区的当前标记环的内容，如本系列文章的第 3 部分教程中所描述的（请参见 参考资料 ）。
mark-ring-max	包含标记环中所允许的条目数。其缺省值是 16。
mouse-avoidance-mode	包含描述 mouse-avoidance 模式类型的值，如本系列文章第 5 部分教程中所描述的（请参见 参考资料 ）。其缺省值是 nil。
next-line-add-newline	如果没有将其设置为 nil，那么只要按下向下箭头键， Emacs 就会在该缓冲区的末尾添加一个新行。其缺省值是 nil（在更新的 Emac 版本中）。
scroll-bar-mode	包含 Emacs 框架侧边缘（放置滚动条的位置）的值：right 或 left。如果设置为 nil，则关闭滚动条。其缺省值是 left。
scroll-step	包含使用 scroll-down 和 scroll-up 函数（在缺省情况下，分别绑定于 PgDn 和 PgUp 键）在缓冲区中移动的行数。如果设置为 0，那么在滚动的时候， Emacs 使光标位于窗口的中心位置。
show-trailing-whitespace	如果没有将其设置为 nil，那么 Emacs 将显示当前缓冲区中的行尾处的任何空白字符。其缺省值是 nil。
visible-bell	如果没有将其设置为 nil，那么 Emacs 将使得该框架闪烁，而不是鸣响系统警铃。其缺省值是 nil。
x-cut-buffer-max	设置剪切环的字符的最大数目，该剪切环同样存储于 X Window System 的剪切缓冲区中。其缺省值是 20000。

自定义您的 Emacs 键绑定

正如您所知道的，**Emacs** 函数绑定于各种称为**键绑定**的按键组合；标准的 **Emacs** 绑定易于使用和记忆（绑定中所使用的键，通常存在与记忆相关的提示），而且使用起来速度很快（因为在设置它们的时候，考虑到您仅需要最少的手动工作，就可以迅速地键入任何给定的按键组合）。

但是如果您不喜欢这些绑定，您也可以对它们进行更改。您可以自定义 **Emacs** 的所有键绑定：如果您需要的话，您可以使 **A** 键入字符 **Z**，或者使得 **C-f** 向后移动光标，使得 **C-b** 向前移动光标。尽管这些更改对于大多数用户来说或许并没有多大用处，但是存在一些重新配置，您个人可能会认为是非常便捷的。而且，与提供了缺省绑定的函数相比，**Emacs** 中包含更多的函数，许多可能的键盘按键组合还没有绑定于任何函数。

如果您频繁地使用一个特定的函数或者函数组，并且它们没有绑定于任何按键（或者如果它们所绑定的按键使用起来没有其他的按键方便），那么您可以重新定义它们。或者如果有一个函数您并不使用，那么您可以删除它的绑定，释放该按键，使其可以为其他函数所用。

global-set-key 函数将给定的函数绑定于给定的按键，并使得它在所有的模式和缓冲区中都有效。接下来将使用它进行大量的自定义和修复。

修复 Home 和 End 键

在 **Emacs** 中，通常按两种方式对 **Home** 和 **End** 键进行配置：要么，它们将光标移动到当前行的开头和结尾；要么，它们将光标移动到当前缓冲区的开头和结尾。

查看您绑定到这两种方式中的哪一种：在您的 **bee** 缓冲区中，按 **Home** 和 **End** 各几次。

现在您知道了这两种键绑定的方式，请尝试更改它们的绑定：

- 如果您的 **Home** 和 **End** 绑定于移动光标到当前行的开头和结尾，那么键入：

```
M-x global-set-key Enter
```

```

Set key globally:

      Home

Set key <home> to command: beginning-of-buffer
                        M-x global-set-key

Set key globally:

      End

Set key <home> to command: end-of-buffer

```

-
- 如果您的 **Home** 和 **End** 绑定于移动光标到当前缓冲区 的开头和结尾，那么键入：

```

                        M-x global-set-key

Set key globally:

      Home

Set key <home> to command: beginning-of-line
                        M-x global-set-key

Set key globally:

      End

Set key <home> to command: end-of-line

```

在这个缓冲区中，测试 **Home** 和 **End** 几次，然后在 **cee** 缓冲区中尝试它们以证实您刚刚所做的更改是全局范围的：

```

      Home
      End
      Home
      End C-x b cee Enter
      Home
      End
      Home
      End

```

修复 Del 键

在 Emacs 的某些配置中，**Del** 键与 **Backspace** 键绑定于相同的函数，`delete-backward-char`。这并不是大多数人所想要的和期望的值；对于大多数软件，**Del** 键都用于删除光标 *后面的* 字符，而不是前面的字符。

测试您的 **Del** 键：

1. 键入 `C-x b *scratch* Enter`，以移动到临时缓冲区。
2. 键入一些文本：

```
Delete me
```

3. 键入 `M-b` 使得光标位于字符 `m` 上。
4. 按下 **Del**。

如果您的 **Del** 键绑定正确，那么将删除 `m`。如果您删除的是这两个单词之间的空格，那么可以使用下面的操作修复您的 **Del** 键：

```
M-x global-set-key
Set key globally:
      Del

Set key C-d to command: delete-char
```

创建自定义键

要为当前没有定义的按键定义一个新的绑定，可以运行 `global-set-key` 函数，给出新的按键，然后给出要运行的命令。

例如，在缺省情况下，**M-F1** 按键是未定义的。通过按住 **Alt**（或者您键盘中的任何 **Meta** 键），再按 **F1** 键，然后同时释放这两个键，就可以键入这个按键。尝试这个操作，当您在进行这个操作的时候，唯一应该发生的事情是 **Emacs** 发出警示声，指出这个按键还没有定义任何函数。

您可以定义这个按键，以运行 `phases-of-moon` 函数，该函数将打开一个显示当前月亮盈亏的新缓冲区窗口。尝试下面的操作：

```
M-x global-set-key
Set key globally: M-F1
Set key <M-F1> to command: phases-of-moon
```

尝试您新定义的键绑定：键入 `M-F1`。在完成这个操作之后，您应该看到打开了一个新的缓冲区窗口，并显示了当前月亮的盈亏。

自定义您的 Emacs 界面

可以通过大量的 **Emacs** 模式和函数来控制 **Emacs** 界面自身，并且您还可以对它们进行自定义。

删除菜单条

Emacs 菜单条对于初学者来说是很有帮助的，但是一旦您熟练掌握了 **Emacs**，就可能想要删除它。您将使用键盘而不是菜单，并且删除菜单条将在屏幕中为您的缓冲区增加额外的一行空间。

使用 `menu-bar-mode` 函数打开和关闭菜单条，该函数可以进行切换。

尝试关闭它，然后再打开它：

```
M-x menu-bar-mode
M-x menu-bar-mode
```

有些类似的函数同样也值得我们深入地研究。`tool-bar-mode` 函数可以切换 **Emacs** 工具条的显示，工具条显示于 **Emacs** 框架（在 **X** 中）的顶部，并且包含大量的图形图标（各种常见 **Emacs** 命令的鼠标快捷方式）。

`scroll-bar-mode` 函数（请参见[表 1](#)）可以控制是否在 **Emacs** 框架的左侧或者右侧绘制滚动条，或者完全将其省略。

在模式行中放置一个时钟

另一个简单的 Emacs 自定义是 `display-time` 函数，该函数可以在模式行中显示当前时间（以及其他有价值的当前状态信息）。

现在，通过键入 `M-x display-time` 来打开它。

在缺省情况下，这个函数在该缓冲区名的右边采用小时和分钟的形式显示当前时间，其后紧跟当前系统的负载级别，以及单词 *Mail*（如果您有尚未阅读邮件的话）。每分钟都会对这些值进行更新。

您可以更进一步对其进行自定义，以便在所有状态信息之前显示当前星期几、以及当前月份（这两者都采用三字母的缩写），其后紧跟该日期。要完成这项操作，您必须设置 `display-time-day-and-date` 变量，在[表 1](#) 中对该变量进行了描述。现在尝试对其进行设置：

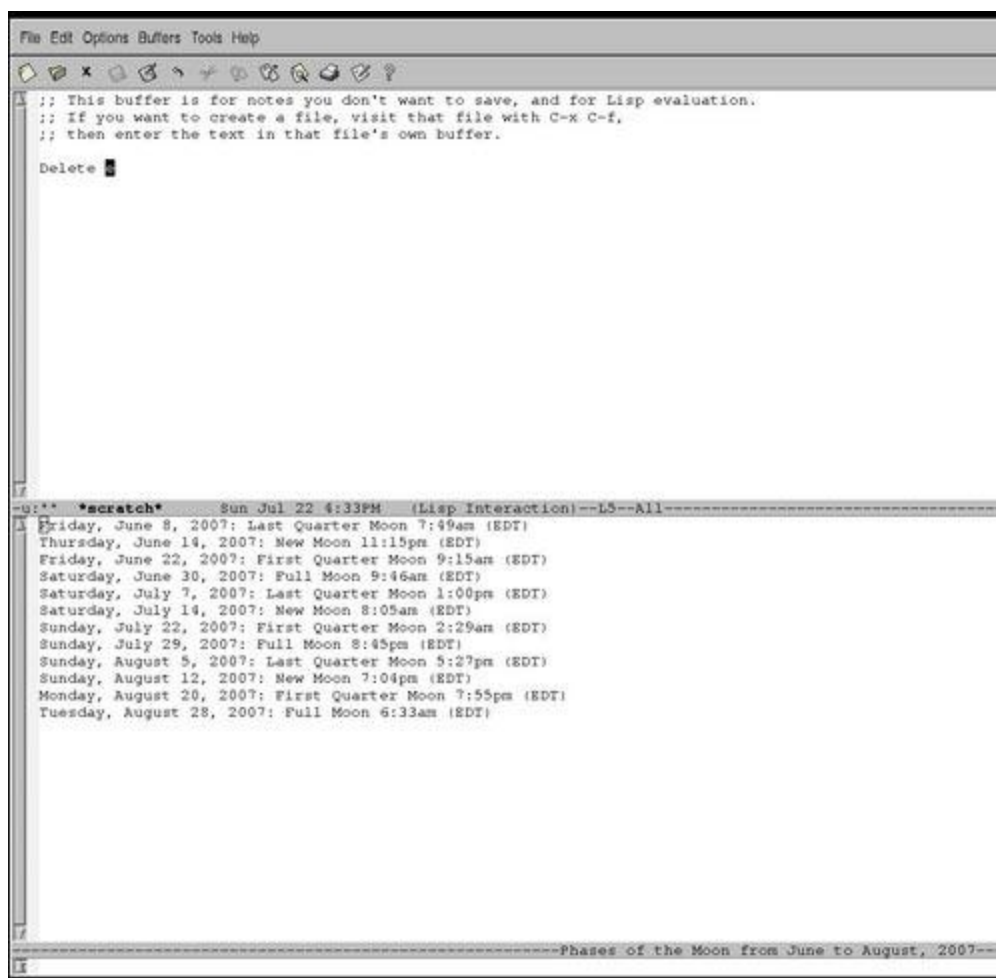
```
M-x set-variable Enter
```

```
set variable: display-time-day-and-date Enter
```

```
set display-time-day-and-date to value: 1 Enter
```

显示并不会立即更新，您必须等待几分钟才能看到更改，或者再次运行 `display-time` 函数（通过键入 `M-x display-time`）。此时，如果您已经遵循本文中的示例进行了相应的操作，那么您的 Emacs 会话应该与[图 2](#) 所示类似。当前缓冲区的模式行（在这个示例中，即 **scratch**）包含星期几、月份的名称、日期和时间（采用标准的 12 小时格式）。您也可以采用 24 小时军用计时格式来显示时间，要实现这一点，可以使用 `display-time-24hr-format` 变量（请参见[表 1](#)）。

图 2. 自定义模式行中当前时间的显示



要返回到原始风格，可以将 `display-time-day-and-date` 设置为 `nil`：

```
M-x set-variable Enter
```

```
Set variable: display-time-day-and-date Enter
```

```
Set display-time-day-and-date to value: nil
```

当分钟发生了更改、或者如果您再次运行 `display-time` 函数的时候，当前时间将出现在模式行中，但没有显示星期几、月份名称和日期。

自定义您的位置

Emacs 可以根据您所处的地理位置完成一些有价值的操作，比如给出日出和日落的时间。要启用这个功能，需要为您的位置设置 `calendar-latitude`、`calendar-longitude` 和 `calendar-location-name` 变量的值。

尝试将这些变量设置为一个示例位置，如[清单 5](#) 中所示。

清单 5. 在 Emacs 中配置地理位置数据

```
M-x set-variable Enter
```

```
set variable: calendar-latitude Enter
```

```
set calendar-latitude to value: 26.37 Enter
```

```
M-x set-variable Enter
```

```
set variable: calendar-longitude Enter
```

```
set calendar-longitude to value: -80.09 Enter
```

```
M-x set-variable Enter
```

```
set variable: calendar-location-name Enter
```

```
set calendar-location-name to value: "Boca Raton, FL (USA)" Enter
```

现在，运行 `sunrise-sunset` 函数：

```
M-x sunrise-sunset
```

在迷你缓冲区中，您应该可以得到今天的日出和日落时间（对于您所给定的位置）。

但是这个函数还可以给出任何一天的日出和日落时间，而不仅仅是今天的。要获得特定日期的日出和日落时间，可以在调用这个函数之前加上 `universal-argument` 和 `C-u`。尝试这个操作，如[清单 6](#) 中所示。

清单 6. 针对特定的日期运行 `sunrise-sunset` 函数

```
C-u M-x sunrise-sunset
```

```
Year (>0): 2007
```

```
Backspace 8 Enter
```

```
Month name: Mar Enter
```

```
Day (1-30): 4
```

在完成了这个操作之后，迷你缓冲区将向您报告下面的信息：

```
Tue, Mar 4, 2008: Sunrise 6:41am (EST), sunset 6:22pm (EST)  
at Boca Raton, FL (USA) (11:41 hours daylight)
```

打开语法突出显示

Emacs 的 Font Lock 模式可以打开字体和颜色，对于包括 *语法突出显示* 在内的许多内容来说，这是很有用的；在语法突出显示中，可以对特定类型文档的重要结构元素进行着色，或者在显示中采用某种方式对其进行格式化。例如，在 HTML 模式中，用某些颜色突出显示 HTML 文档中用括号括起来的各种标记。font-lock-mode 函数可以进行切换，以便为当前缓冲区打开或者关闭这个模式。

现在尝试设置它：

1. 通过键入 C-x C-f，然后指定 ~/.emacs 作为要查找的文件，来访问名为 .emacs 的新缓冲区。
2. 如果在您的 home 目录中有一个 .emacs 文件，那么前面的步骤将在一个新的缓冲区中打开它；如果没有 .emacs 文件，那么 Emacs 将生成一个使用该名称的新的空文件。无论在何种情况下，插入[清单 7](#)中给出的 Emacs Lisp 的内容行，以定义刚刚描述的三个位置变量（当然，除非先前存在的 .emacs 文件已经包括，并且已定义了这些变量）。

清单 7. 在 .emacs 文件中指定地理位置

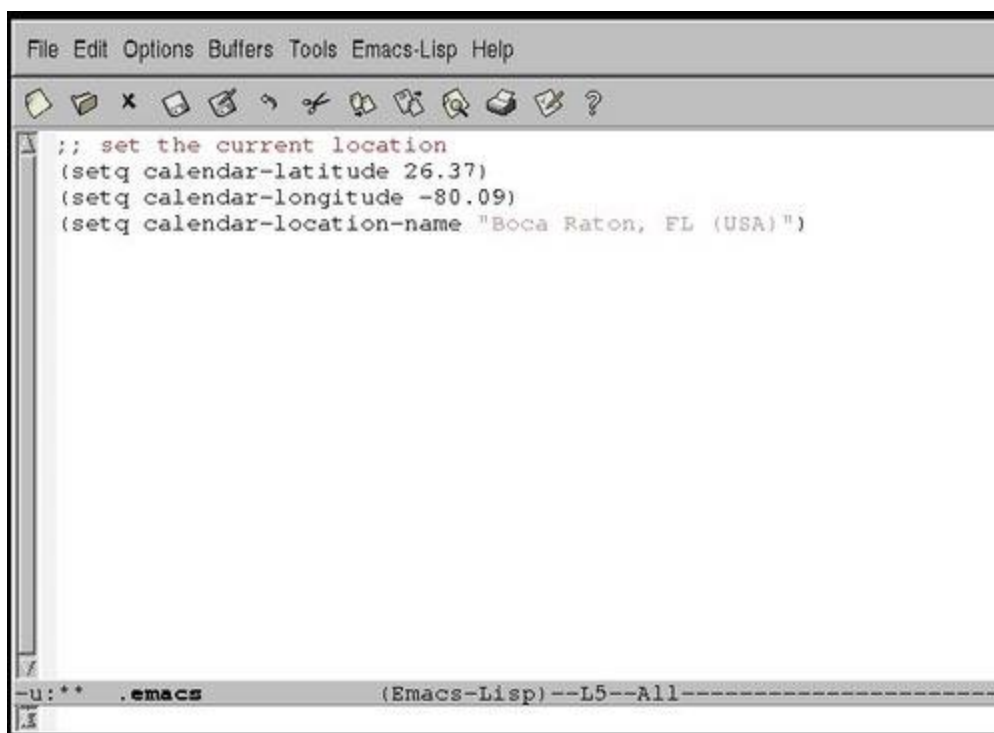
```
;; set the current location
(setq calendar-latitude 26.37)
(setq calendar-longitude -80.09)
(setq calendar-location-name "Boca Raton, FL (USA)")
```

3. 为这个缓冲区打开 Font Lock 模式：

```
M-x font-lock-mode
```

使用应用于各种元素的突出显示方式，以相应的颜色显示 Emacs Lisp 代码，如[图 3](#)中所示。

图 3. 在 Emacs Lisp 缓冲区中启用 Font Lock 模式



您可以通过键入 C-x C-s 将该文件保存到磁盘。

常见 Emacs 界面函数汇总

[表 2](#) 列出并描述了影响 Emacs 界面的各种函数。

表 2. 常见 Emacs 界面函数汇总

函数	描述
column-number-mode	在模式行中，为光标所在的当前列（前面有一个 C）切换显示。其缺省值是 nil。
display-time	在模式行中切换当前时间的显示。其缺省值是 nil。
font-lock-mode	如果没有将其设置为 nil，那么 Emacs 将为当前缓冲区自动地打开 Font Lock 模式。其缺省值是 nil。
global-font-lock-mode	如果没有将其设置为 nil，那么 Emacs 将为所有的缓冲区自动地打开 Font Lock 模式。其缺省值是 nil。
line-number-mode	在模式行中，为光标所在的当前行（前面有一个 L）切换显示，其缺省值是 t。
menu-bar-mode	切换 Emacs 菜单条的显示。其缺省值是 t。
sunrise-sunset	根据当前地理位置，显示今天日出和日落的时间。如果在它的前面加上 universal-argument，那么这个函数将提示输入一个特定的日期。
tool-bar-mode	切换 Emacs 工具条的显示。其缺省值是 t。

使用您的 Emacs 启动文件

每次 Emacs 启动的时候，它都会在您的 home 目录中查找一个隐藏文件 `.emacs`，并执行该文件中包含的所有 Emacs Lisp。这就是您的 Emacs 启动文件或者初始化文件；可以使用它来设置变量或者进行其他更改（在 Emacs 启动时自动发生）。

使用圆括号将 Emacs Lisp 调用括起来；注释的内容以分号 (;) 开头。随着 Emacs Lisp 的发展，产生了许多约定；例如，描述函数的注释通常以两个分号和一个空格字符开头。在 *GNU Emacs Lisp 参考手册*（请参见[参考资料](#)）中描述了这些约定。

在 `.emacs` 文件中，您可以进行各种各样的自定义工作，包括设置变量、定义键绑定并设置各种模式，接下来将对这些内容进行描述。

（正如在本系列文章的第 4 部分教程中所描述的（请参见[参考资料](#)），您可以启动 Emacs，而无需通过给定 `-q` 选项首先加载这个初始化文件，并且您可以通过给出用户名作为 `-u` 选项的参数，以使得 Emacs 加载另一个用户的启动文件。）

在您的启动文件中设置变量

使用 `.emacs` 初始化文件的最常见的目的可能是[设置某些变量](#)，以使得每次启动 Emacs 的时候，都能够自动地设置这些值。要在 `.emacs` 文件中设置变量，可以使用 `setq` 函数，如下所示：

```
(setq VARIABLE VALUE)
```

您在[清单 7](#) 中看到了一个示例，其中您创建了一个简单的 `.emacs` 文件并将该缓冲区保存到您的 home 目录中。现在您可以对其进行演示：

1. 键入 `C-x C-c`，以退出 Emacs；可能会丢失在任何未保存的缓冲区（本教程中您进行工作的缓冲区）中所做的更改，这没有什么关系。
2. 重新启动 Emacs：

```
$ emacs
```

3.

4. 获得今天的日出和日落时间：键入 `M-x sunrise-sunset`。

您无需设置 Emacs 的纬度、经度、或者位置名称变量，因为当 Emacs 启动的时候，它会从您的 `.emacs` 文件中读取它们。但是在 Emacs 会话中的任何位置，您都可以重新定义在您的 `.emacs` 文件中设置的任何变量。

在您的启动文件中定义键绑定

当您使用在[创建自定义键](#)部分中所描述的方法来定义新的键绑定时，它仅仅在当前 Emacs 会话中是可用的（一旦退出，这些更改都将丢失）。但是您可以将自定义的绑定放在您的 `.emacs` 文件中，以使得在您每次启动 Emacs 的时候它们都是可用的。

例如，如果您的 `Del` 键有问题，那么您可以在 `.emacs` 文件中包括下面的内容：

```
;; Make Del delete the character at point.
(global-set-key [delete] 'delete-char)
```

并且您可以包括[清单 8](#)中所描述的任何一对代码，以便自动地为每个 Emacs 会话重新定义您的 `Home` 和 `End` 键。

清单 8. 在 `.emacs` 文件中重新定义 `Home` 和 `End` 键

```
;; Make Home and End move to the beginning and ending of the buffer
(global-set-key [home] 'beginning-of-buffer)
(global-set-key [end] 'end-of-buffer)

;; Make Home and End move to the beginning and ending of the line
(global-set-key [home] 'beginning-of-line)
(global-set-key [end] 'end-of-line)

;; Make Home and End move to the beginning and ending of the sentence
(global-set-key [home] 'backward-sentence)
(global-set-key [end] 'forward-sentence)
```

在您的启动文件中设置模式

还可以在您的 `.emacs` 文件中打开或者关闭模式。

因为大多数模式都可以作为切换进行调用，所以在其后紧跟一个正参数可以打开它们，而紧跟一个负参数可以关闭它们。您还可以使用下面这种格式：

```
(MODE-NAME VALUE)
```

例如，您可以使用下面的操作自动地关闭菜单条：

```
;; turn off the Emacs menu bar
(menu-bar-mode -1)
```

通常，当给定一个正整数值或者 `t`（表示“真”）的时候，将打开模式；当给定一个负整数、零或者 `nil` 值的时候，将关闭模式。

下面所有的行都可以完成相同的任务：它们设置 `global-font-lock-mode`，该模式将自动地打开所有缓冲区的 `Font Lock` 模式：

```
(global-font-lock-mode)
(global-font-lock-mode 1)
(global-font-lock-mode t)
```

使得简短的回答成为可能

有时，当 Emacs 需要从您这里得到一个回答的时候，您不得不键入单词 `yes` 或者 `no`，并在其后使用 **Enter** 键。当您很清楚自己所执行的操作时，这个操作就可能会和某些窗口应用程序要求您在弹出窗口中单击 **OK** 以验证您刚刚指定的命令一样令人讨厌。

为了简化这个过程，以便只需要按 **Y** 或者 **N** 键即可，可以将下面的内容添加到您的 `.emacs` 文件：

```
;; eliminate long "yes" or "no" prompts
(fset 'yes-or-no-p 'y-or-n-p)
```

这种技术使用了 `fset` 函数，以重新定义给定的函数。要了解更多关于这个函数和类似的 Emacs Lisp 函数的信息，请参阅 *GNU Emacs Lisp 手册*（请参见[参考资料](#)）。

实际运用

[清单 9](#) 中包含一个示例 `.emacs` 初始化文件，以及在这个部分中描述的许多自定义内容。它还包含使用[表 1](#)和[表 2](#)中所描述函数和变量进行附加自定义的内容。您可以编辑新的 `.emacs` 文件，以包含这些条目，然后重新启动 Emacs 以观察这些更改如何产生作用。

清单 9. 示例 `.emacs` 初始化文件

```
;; set the current location
(setq calendar-latitude 26.37)
(setq calendar-longitude -80.09)
(setq calendar-location-name "Boca Raton, FL (USA)")

;; eliminate long "yes" or "no" prompts
(fset 'yes-or-no-p 'y-or-n-p)

;; Custom mode configurations
;;
;; turn off the menu bar
(menu-bar-mode -1)
;;
;; turn off the tool bar
(tool-bar-mode -1)
;;
;; turn on Font Lock mode globally
(global-font-lock-mode t)

;; Custom key bindings
;;
```

```
;; Make Del delete the character at point.
(global-set-key [delete] 'delete-char)
;;
;; Make Home and End move to the beginning and ending of the sentence
(global-set-key [home] 'backward-sentence)
(global-set-key [end] 'forward-sentence)
```

记住 Emacs 窗口和框架的自定义内容

在本系列文章的第 5 部分教程（请参见[参考资料](#)）中，您了解了如何操作 Emacs 框架，并将它们水平或者垂直地划分为多个窗口。您还可以保存对窗口所做的自定义工作，并稍后在您的会话中重新调用它们，甚至在对屏幕的布局做了许多更改之后重新调用它们；您可以使用 *Winner* 模式来完成这个任务。

Winner 模式是一个次要模式，它可以记录并记住您对窗口和框架所做的更改。您对窗口配置的每次更改，它都会记住；您可以撤消您的更改，以还原为最近的配置。您还可以重做您的更改，以返回到包含这些更改的情况。如果您正在尝试不同寻常的或者复杂的配置，或者如果您希望完成不同类型的任务（涉及到不同的窗口设置），那么使用它是非常方便的。

在 *Winner* 模式中工作的两个函数：C-x <-（通过按住 **Ctrl**，再按 **X**，同时释放这两个键，然后按**向左箭头**键，可以键入这个函数）将运行 `winner-undo` 函数，并撤销对窗口所做的最近一次更改。它的对等项是 C-x ->，它将运行 `winner-redo`，恢复对窗口所做的最近一次（您先前撤销的）更改。

启用 Winner 模式

要开始使用 *Winner* 模式，可以设置一个自定义 Emacs 框架：

1. 退出 Emacs（如果当前它在运行的话），然后重新启动它（不使用任何参数）：

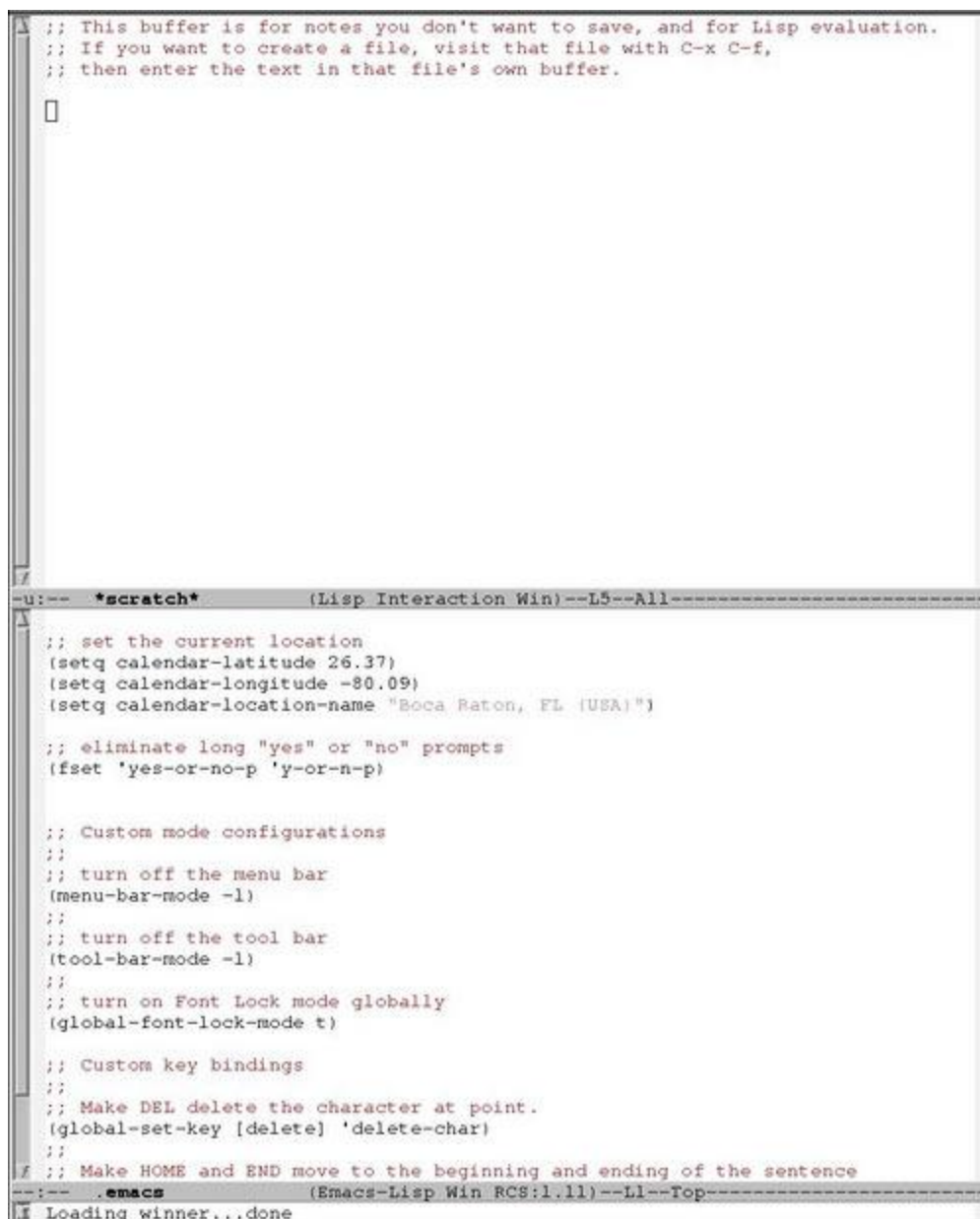
```
$ emacs
```

- 2.
3. 键入 C-x 2 以垂直地划分 Emacs 窗口，然后键入 C-x o C-x C-f .emacs Enter 以便在下面的窗口中打开您的 .emacs 文件。
4. 键入 M-x winner-mode 以打开 Winner 模式。

当您在进行这个操作时，请注意在两个可见缓冲区的模式行中，已将 win 添加到括号中的模式清单。这意味着，已经打开了 *Winner* 模式，并且它在所有的缓冲区中都可以工作。（它甚至还可以在其他 Emacs 框架中工作，只要您创建了它们。）

您的 Emacs 会话应如 [图 4](#) 所示。请注意，Emacs 框架不再包含菜单条或者工具条，因为在 .emacs 文件中关闭了这些内容。

图 4.在 Emacs 会话中启用 Winner 模式



```
;; This buffer is for notes you don't want to save, and for Lisp evaluation.
;; If you want to create a file, visit that file with C-x C-f,
;; then enter the text in that file's own buffer.

□

--u:-- *scratch* (Lisp Interaction Win)--L5--All--
;; set the current location
(setq calendar-latitude 26.37)
(setq calendar-longitude -80.09)
(setq calendar-location-name "Boca Raton, FL (USA)")

;; eliminate long "yes" or "no" prompts
(fset 'yes-or-no-p 'y-or-n-p)

;; Custom mode configurations
;;
;; turn off the menu bar
(menu-bar-mode -1)
;;
;; turn off the tool bar
(tool-bar-mode -1)
;;
;; turn on Font Lock mode globally
(global-font-lock-mode t)

;; Custom key bindings
;;
;; Make DEL delete the character at point.
(global-set-key [delete] 'delete-char)
;;
;; Make HOME and END move to the beginning and ending of the sentence
--:-- .emacs (Emacs-Lisp Win RCS:1.11)--L1--Top--
Loading winner...done
```

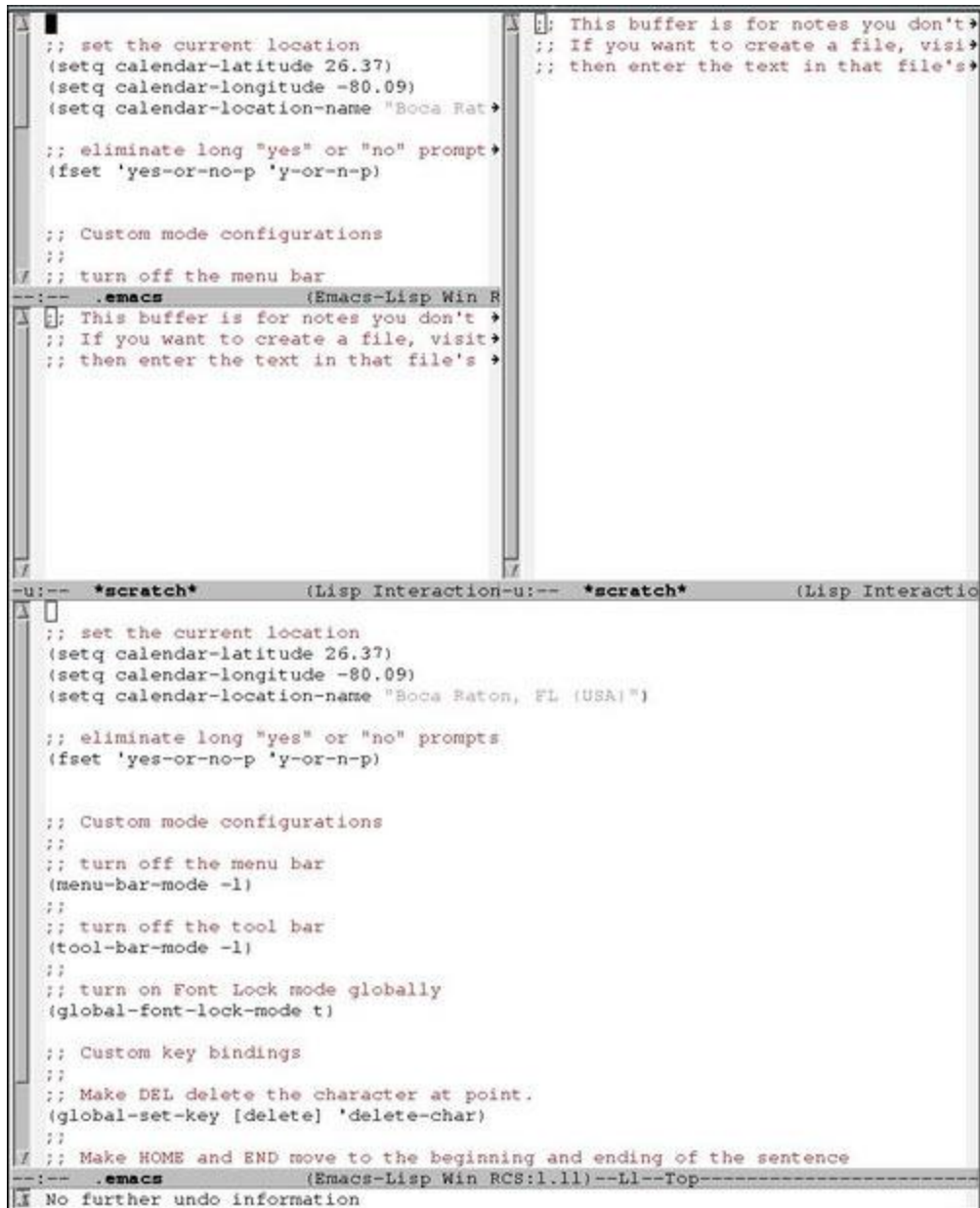
遍历您的窗口配置

现在，进行一些新的窗口配置：

1. 使用鼠标单击顶部窗口中的 **B1**，使得其变为活动窗口，然后键入 `C-x 3` 以水平地划分该窗口一次。
2. 通过键入 `C-x 2` 垂直地划分左上方的窗口，然后通过键入 `C-x` 再次水平地划分新的、左上方的窗口。键入 `C-x b .emacs Enter` 以切换到新窗口中的 `.emacs` 缓冲区。

您的 Emacs 框架应该与图 5 所示类似。请注意，在该框架的每个窗口的模式行中都出现了 *Win*。

图 5. 启用 Winner 模式重新配置 Emacs 框架



现在，您可以返回到过去的配置，并遍历它们，键入 `C-x <-` 以撤消您最近对窗口所做的更改。

再次键入 `C-x <-`，以撤消另一个更改。继续进行撤消操作，直到迷你缓冲区报告无法进行更进一步的撤销操作为止。此时，您的 Emacs 框架应该与图 4 所示类似，即您第一次启用 Winner 模式时的样子。

键入 `C-x ->` 以运行 `winner-redo` 函数，并撤消您最近的更改。您可以在 `C-x <-` 和 `C-x ->` 之间循环，以遍历所做的所有窗口更改。

`winner-mode` 函数是一个切换：再次键入 `M-x winner-mode`，以将其关闭。当您在进行这个操作时，`winner-redo` 和 `winner-undo` 函数将不再起作用，并且不再记住窗口的更改，但是一旦再次启用它，您可以撤消和重做您对窗口所做过的任何更改。

使用 Emacs 的 Customize 函数

`Customize` 是一个特殊的 Emacs 函数，该函数可以使得对编辑器行为各个方面的自定义变得更加容易。它允许您进行交互地选择、设置和保存相关的值，以及 Emacs 许多方面的设置，包括用于显示文本的各种字体、外观和颜色。其目的是在详细的 Emacs 自定义过程中进行简化和控制，使得对于非 Emacs Lisp 程序员的 Emacs 用户可以更加容易地自定义他们的设置。

对于设置文本颜色和外观，Customize 函数特别有价值，因为您可以观察并且交互地进行更改，您可以在您的当前会话中测试它们，而无需将它们应用到所有将来的会话中。当您为以后的会话保存您的更改时，可以自动地将它们写入您的 .emacs 文件，因此它们将应用于当前以及所有将来的 Emacs 会话。

启动自定义缓冲区

要开始对您的 Emacs 进行自定义工作，可以通过键入 M-x customize 运行 customize 函数。

当您在进行这项操作时，在您的 Emacs 框架中将出现一个新的自定义缓冲区，如图 6 所示。请注意该缓冲区的名称：*Customize Group: Emacs*。

图 6. Emacs 自定义缓冲区



选择自定义组

将 Emacs 中可进行自定义的各个方面划分为不同的组，每个组中包含一些类似的元素和编辑器的方面。所显示的组取决于您的系统设置。

要打开一个组并观察其中不同的成员，可以使用鼠标左键单击 **Go to Group** 文本按钮，或者使用键盘将光标移动到该按钮上并按 **Enter**。有时候，打开一个组可以得到新的子组的列表，在这种情况下，进行选择的过程是相同的。

现在, 尝试选择 **Editing** 组。当您在进行这个操作时, Emacs 将打开一个名为 ***Customize Group** 的新缓冲区: **Editing*** 包含一个新的组清单, 其中所有的内容都与 Emacs 的编辑函数相关。

在这个菜单中, 选择 **Fill** 组, 以便打开 ***Customize Group: Fill*** 缓冲区。这个缓冲区包含许多不同的选项和附加的子组, 您必须向下滚动以查看所有的内容。您的 Emacs 会话应该与图 7 中所示类似。

图 7. Emacs Fill 自定义组缓冲区



更改自定义选项

这个组清单中的每个选项都有一个值, 您可以对其进行更改。对于每个选项, 包括 **More** 选项 (提供关于该选项的更多信息), 您都可以看到许多按钮。当您单击这个按钮的时候, 它的标签变为 **Hide**; 当您再次单击它的时候, 则将扩展内容隐藏起来。

State 按钮描述该选项的当前状态: 未对其标准设置进行过更改、隐藏、经过更改但未保存、或者经过更改且已保存。

尝试更改一个值: 向下滚动到 **Fill Column** 条目, 请注意该值设置为 **70**, 您应该记得这恰好是该变量的缺省值。将光标移动到该值的框, 并将其更改为 **10**。这个值的 **State** 按钮可以报告下面的信息:

you have edited the value as text, but you have not set the option.

向上移动到该缓冲区的顶部，并单击 **Set for Current Session** 按钮；这个操作将使您的更改生效，但是一旦您退出 Emacs 它们都将重新设置。（如果您希望进行一项影响本次会话和所有将来会话的更改，可以改为单击 **Save for Future Sessions** 按钮）。

单击 **Finish** 按钮，这个操作将删除该缓冲区，并单击所有其他 **Customize** 缓冲区中的 **Finish** 按钮。

现在，您可以测试您的自定义内容：

1. 键入 `C-x C-f .emacs Enter` 以便在新的缓冲区中打开您的 `.emacs` 初始化文件的一个副本。
2. 将光标移动到每个命令行（它们以 `;;` 开头，其后紧跟一行文本），并键入 `M-q`，对它们运行 `fill-paragraph` 函数。

这些段落将在每行的第十个字符处进行填充，因此您的 `.emacs` 初始化文件应该与[图 8](#) 中所示类似。

图 8. 自定义填充列之后，在 .emacs 文件中对注释文本进行自动换行

```
;; set the
;; current
;; location
(setq calendar-latitude 26.37)
(setq calendar-longitude -80.09)
(setq calendar-location-name "Boca Raton, FL (USA)")

;; eliminate
;; long
;; "yes"
;; or "no"
;; prompts
(fset 'yes-or-no-p 'y-or-n-p)

;; Custom
;; mode
;; configurations
;;
;; turn
;; off the
;; menu
;; bar
(menu-bar-mode -1)
;; turn
;; off the
;; tool
;; bar
(tool-bar-mode -1)
;; turn
;; on Font
;; Lock
;; mode
;; globally
(global-font-lock-mode t)

;; Custom
;; key
;; bindings
;;
;; Make
;; DEL
;; delete
;; the
;; character
;; at
;; point.
(global-set-key [delete] 'delete-char)
;; Make
;; HOME
;; and END
;; move to
;; the
;; beginning
--:** .emacs (Emacs-Lisp RCS:1.11)--Ll--Top-----
| Mark set
```

您可以验证，对缺省填充所做的更改仅作用于本次会话：保存经过重新格式化的 .emacs 文件，使用 C-x C-c 退出 Emacs，然后使用您的 .emacs 文件重新启动它：

```
$ emacs .emacs
```

现在，移动到包含文本的所有注释行，并在每行中键入 M-q，以观察它们变为之前的情况（在第 70 列处进行填充）。

自定义特定的属性

[↑ 回页首](#)

每次当您希望创建一项特殊的自定义内容时，并不需要遍历自定义缓冲区的整个系列。有一些命令可以直接将您带到特定的组，并且某些 Emacs 自定义菜单具有它们自己的特殊 Emacs 函数。

例如，customize-group 函数可以为给定的组打开自定义缓冲区。

尝试打开 Fill 组的一个新的 Emacs 自定义缓冲区：

```
M-x customize-group
Customize group: (default emacs) fill
```

与前面一样，您得到一个 *Customize Group:Fill* 缓冲区，如[图 7](#) 中所示。

了解 Emacs 自定义函数

[表 3](#) 列出并描述了各种 Emacs Customize 函数。

表 3. Emacs Customize 函数汇总

函数	描述
customize-changed-options <i>Enter version</i>	为所有的外观、选项、或者自从通过 <i>version</i> 给定 Emacs 的版本之后经过更改的组，打开一个新的自定义缓冲区。
customize-customized	为所有经过自定义但尚未保存到磁盘的选项和外观，打开一个新的自定义缓冲区。
customize-face <i>Enter regexp</i>	为所有的外观、选项、或者与通过 <i>regexp</i> 给定的正则表达式相关的组，打开一个新的自定义缓冲区。
customize-face <i>Enter face</i>	为通过 <i>face</i> 给定的外观名称，打开一个新的自定义缓冲区。
customize-group <i>Enter group</i>	为通过 <i>group</i> 给定的组名称，打开一个新的自定义缓冲区。
customize-option <i>Enter option</i>	为通过 <i>option</i> 给定的选项名称，打开一个新的自定义缓冲区。
customize-saved	为所有使用 Customize 函数更改的外观和选项，打开一个新的自定义缓冲区。

总结

您已经了解了几种通过设置变量、更改缺省键绑定和以交互地方式操作模式来配置和自定义 Emacs 编辑环境的重要方式。您还了解了如何使用 Emacs 初始化文件为将来的会话保存您的自定义内容，如何使用 Winner 模式以保存并重新调用自定义 Emacs 窗口和框架，以及如何使用内置的 Emacs Customize 函数来自定义各种内容（从基本的字体和颜色，到执行特定的文本编辑函数的方式）。

Emacs 的各个方面都可以采用功能强大的方式进行更改或者扩展，并且您已经学习了如何完成这项任务。现在，您可以让 Emacs 按照您所希望的方式去工作了。