GNU技术文档精粹

GNU Emacs Lisp编程入门

(美) Robert J. Chassell 著
毛文涛、吕芳 译
洪峰 审校

机械工业出版社
China Machine Press

本书的作者罗伯特·卡塞尔是自由软件基金会的合创人之一，也是理查德·斯托曼博士青年时期结交的挚友，他精通GNU Emacs Lisp的每一个方面。本书是一本GNU Emacs Lisp的编程入门，全书循序渐进地介绍了GNU Emacs Lisp编程的各种基础知识和方法，文笔流畅、讲解透彻，对GNU Emacs用户提高对它的理解和运用帮助极大。

图书在版编目（CIP）数据

GNU Emacs Lisp编程入门 /（美）卡塞尔（Chassell, R. J.）著；毛文涛等译. -北京：机械工业出版社，2001.5

（GNU技术文档精粹）

书名原文：Programming in Emacs Lisp: An Introduction

ISBN 7-111-08862-X

Ⅰ.G… Ⅱ.①卡… ②毛… Ⅲ.LISP语言-程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字（2001）第19381号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：姚蕾

北京昌平奔腾印刷厂印刷 ·新华书店北京发行所发行

2001年5月第1版第1次印刷

787mm×1092mm 1/16 · 13.5印张

印数：0 001-5 000册

定价：38.00元

致中国读者

Calendars, email, writing in general, programming, and debugging programs: GNU Emacs gives you tools for all these actions, and gives you even more. GNU Emacs is truly an integrated environment. Emacs Lisp is the language in which most of Emacs is written. It is a simple yet powerful language that is easily understood and learned.

FSF-CHINA led by Hong Feng is doing us all a great service by translating this document from English into Chinese. The work will bring the joy, the efficiency, and the power of GNU Emacs to many people. I hope you will gain as much from reading this book as I did from writing it.

My best wishes to you. Bob Chassell Robert J. Chassell

译者序

GNU Emacs 长期以来一直是自由软件基金会的旗舰产品。它是由理查德·斯托曼（Richard Stallman）博士为 GNU 工程开发的第一个自由软件。在所有目前已开发的 GNU 软件中，GNU Emacs 的作用和地位是非常突出的，因为几乎所有其他的自由软件基金会的工具都是用GNU Emacs 编写出来的。

从编程实践上看，GNU Emacs 有许多特点。其中最为突出的一个特点是斯托曼在创造 GNU Emacs 编辑器时非常巧妙地揉合了用Lisp 语言和 C 语言编写的代码。

斯托曼利用 Lisp 语言编写 GNU Emacs 的大部分代码不是偶然的。Lisp 语言发明于20世纪50年代，并广泛地应用于人工智能研究领域，而斯托曼早年曾经在麻省理工学院人工智能实验室工作过很长时间，所以他非常熟悉 Lisp 语言的优点。Lisp 是解释性的语言，用 Lisp 开发的程序具有良好的可读性，因此将它用于处理文本编辑这样的任务是非常合适的。当然解释性的语言与硬件直接作用时其效率与编译性的语言相比则显得不高，这样的任务还是由 C 语言代码来完成比较合适。

斯托曼的一个天才构想就是利用 C 语言编写与硬件直接作用的 GNU Emacs 模块(如显示模块)，而绝大多数文本编辑模块则统统利用 Lisp 语言来编写。Lisp 语言是一种功能全而的编程语言，其解释器被嵌入了 GNU Emacs 中后，用户便可以用它自行对 GNU Emacs 进行定制。这一几乎无限的灵活性是其他编辑器很难做到的。在 GNU Emacs中，Emacs的Lisp代码模块和 C 代码模块组织良好，它们相互取长补短，相得益彰。

为了保持源代码的可读性与一致性，斯托曼将 GNU Emacs 中的 C 语言代码模块的函数名写得很像 Lisp 函数名。如果不仔细研究模块底层的细节，那么实在很难将它们两者区分开。实际上，如果没有特殊的目的，作者希望你不要去辨认它们之间的区别。这样，在扩充和维护代码时，工作就会变得容易多了。经过这么一番匠心独运的安排，GNU Emacs 成为了一种“高级的、自带文档的、可定制的和可扩充的实时显示的编辑器”。难怪如此众多的自由软件开发人员整天都可以坐在计算机旁运行 GNU Emacs 而乐此不疲。

自从 GNU Emacs 问世以来，众多专家一致评价说：斯托曼的这一天才的泛对称设计思想极富艺术性，具有方法论研究的永久价值。

由于使用GNU Emacs 的开发人员数量众多，运行的平台又很广泛，因此各种使用GNU Emacs Lisp 编写的增强功能包也源源不断地产生了，有些功能包还成为标准GNU Emacs 发行版本的一部分。GNU Emacs 今天仍处在不断的演进和完善过程中，它代表着一种文化。这一切都是由于斯托曼当年的天才设计思想所引发的良好局而。

当然，要了解和运用这些新功能，或者自己动手开发所希望的特性来增强和扩充 GNU Emacs，使用 GNU Emacs Lisp 编程是必不可少的，而通过本书学习编程可以说是一个良好的起点。

本书的作者，罗伯特·卡塞尔（Robert Chassell）先生是自由软件基金会的合创人之一，也是斯托曼青年时期结交的挚友。他精通 GNU Emacs Lisp 的每一个方面。在这本编程入门著作中，他循序渐进地介绍了 GNU Emacs Lisp 编程的各种基础知识和方法，文笔流畅、讲解透彻，对GNU Emacs 用户提高对它的理解和运用帮助极大。

另外，由于各种 Lisp 解释器大同小异，因此，一旦通过这一教程理解了 GNU Emacs Lisp 的工作原理，那么你所掌握的知识和技巧对于学习其他版本的 Lisp 语言（如 AutoLisp）或者现在日趋流行的Python等解释性语言也会具有触类旁通的指导意义。

这部著作的中文版是自由软件基金会中国研究院组织人员翻译的第一本 GNU自由软件文档。我们出版这一著作以及《*GNU Emacs* 技术手册》、《*GNU Emacs Lisp* 技术手册》等中文版的主要目的就在于更好地引导读者体会编写程序、分析自由软件源代码的乐趣。这对于在中国催生新的黑客具有极大的促进作用，因为我相信黑客道的真正本质就是“热爱编写程序、并享受通过编写程序而变得更加聪明这一过程”。

自由软件基金会中国研究院
洪 峰
fred@mail.rons.net.cn

前　言

GNU Emacs 文本编辑器的绝大多数代码是用一种被称为 Emacs Lisp 的编程语言编写的。用这种语言编写的代码就是这个软件——指令集——用户通过它向计算机发布命令以告诉计算机如何工作。Emacs 就是为使你能用 Emacs Lisp 编写新的代码并能方便地作为编辑器的扩展部分来安装而设计的。这也是为什么 Emacs被称作“可扩展的编辑器”的原因。

因为 Emacs 的确提供了比编辑更多的功能，它或许应当被称为“可扩展的计算环境”，但是这个词显得口气太大。同样，在 Emacs 中做的任何事情——查找玛雅年代和月相、简化多项式、调试代码、管理文件、阅读信件以及撰写图书——所有这些活动都是“编辑”这个词所包含的。

虽然人们经常将 Emacs Lisp 与文本编辑器联系到一起，但它却是一种完整的计算机编程语言。可以像使用任何其他编程语言一样使用它。

也许你希望理解编程；也许你希望扩展 Emacs；或者也许你希望成为一名程序员。这本入门教程就是为你开始 Emacs Lisp 之旅而设计的：引导你学习编程基础，更重要的是告诉你如何自学提高。

在整本书中，你将看到为数不多的几个程序例子，你可以在 Emacs 中运行它们。如果用 GNU Emacs 的 Info 阅读本书文档，可以在例子程序出现时运行它们。（这很容易做到，我们将在例子出现时作进一步解释。）同时，你也可以将这本教程作为一本印刷的图书一样，当你坐在计算机旁运行 Emacs 时阅读。（这就是我所喜欢的方式，我喜欢印刷出来的纸版图书。）如果你身边没有一个运行的Emacs，你仍旧可以阅读这本书，但是在这种情况下，最好将其作为一本小说或者是一本你未到过的某个国家的导游手册来阅读：这样读起来会较有趣，但是你的收获会与亲身体验不同。

本教程用许多篇幅介绍 GNU Emacs 用到的代码。教程的这些设计安排有两个目的：一是使读者熟悉真实的正在运行的代码；二是使读者熟悉 Emacs 工作的方式。弄清一个编辑器如何工作是很有趣的。同样，我希望读者养成浏览源代码的习惯。读者可以从中学习并开阔思路。有了 GNU Emacs，就像拥有一个龙穴宝藏一样。

除了将 Emacs 当做一个编辑器、将 Emacs Lisp 当做一门编程语言学习之外，书中的例子和导引将使读者通晓将 Emacs 作为Lisp 编程环境的机会。GNU Emacs 支持编程，并提供了你将乐于使用的工具，如 “M-”（这是调用 `find-tag` 命令的键）。你还可以学习缓冲区和其他对象，这些都是编辑环境的组成部分。学习 Emacs 的这些功能就像熟悉家乡周围的新路一样。

最后，我希望传授一些使用 Emacs 来学习编程时读者不知道的技巧。你可以经常用Emacs 来解决那些困扰你的问题，并用它们做一些新奇的事情。这种自力更生不仅是一种乐趣，更是一种优点。

读者对象

这本教程是作为入门读物为那些非编程人员编写的。如果你是一名程序员，可能并不满足

这本初级读物。原因在于你可能已经通过阅读参考手册成了专家。或许本书的组织方式已经使你失去兴趣。

一位评论过本书的编程专家曾这样对我说：

> 我更喜欢从参考手册中学习（编程）。我“潜入”每一个段落，并在段落之间跃出“水面”呼吸空气。
>
> 当到达一个段落结尾时，我假定这个主题已经结束。我知道了需要知道的所有东西（也可能存在这样的可能性，那就是下一个段落将对这个主题作更加详细的讲解）。我期望一份认真撰写的参考手册不要出现太多的冗余，并且它应指引我学习所希望的知识。

这本入门教程并不是为这类读者撰写的！

首先，我试图就每一件事情至少说上三遍：第一次介绍它；第二次在文中详细展现它的内容；第三次在不同的地方揭示它，或者复习一下。

其次，我几乎从不将一个主题的所有内容放在一个地方讲完，更不放在某一段中。以我思考的方式而言，那样做会给读者强加过重的负担。相反，我试图仅仅解释在那种情况下你需要知道的东西（有时会增加一点点附加信息，在后面读到这些附加信息的正式介绍时无需惊讶）。

阅读本书的时候，我并不指望你第一次就学会所有的东西。通常的情况是你仅需要对某些内容略微了解。我希望已经组织好本书，为你提供了足够的信息，并提醒你哪些是重要的线索，且着重讲述它们。

你应当“潜入”某些段落，除此以外没有其他方法。但是我已尽力减少这类段落。希望本书成为一座可以攀越的小山，而不是一座使人畏缩的高峰。

《*GNU Emacs Lisp* 编程入门》还有一个姊妹篇，那就是《*GNU Emacs Lisp* 技术手册》。那本手册比本书更详细。在那本手册中，关于任何一个话题的所有信息都集中在一个地方。如果你喜欢上面引用的那位程序员所欣赏的学习方法，那么应当掉头去阅读那本技术手册。当然，阅读完这本编程入门后，在编写自己的程序时，你会发现那本技术手册很有用。

Lisp 的历史

Lisp 是20世纪50年代晚期在麻省理工学院为研究人工智能而被首先发展起来的。Lisp语言的强大功能使之也能用于其他目的，比如编写编辑器命令。

GNU Emacs Lisp 在很大程度上得益于20世纪60年代在MIT编写的 MacLisp 。它同时也得益于在20世纪80年代成为标准的 Common Lisp。然而，Emacs Lisp 比 Common Lisp 简单得多（标准的 Emacs 发行版本中包含一个可选的扩展文件“`cl.el`”，它为 Emacs Lisp 增加了许多 Common Lisp 的特性）。

初学者注意

如果你不知道 GNU Emacs，阅读本书仍旧有益。但是，如果仅仅是想学习在计算机屏幕上如何操作，建议你学习 Emacs。可以通过在线教程自学如何使用 Emacs。为使用在线教程，按下组合键 `C-h t`（这意味着同时按下并释放 CTRL和 `h` 键，然后按下并释放 `t` 键）。

同样，我经常在提到 Emacs 的标准命令时列出激活该命令时应按下的键序列，然后在括号

中给出命令名，例如：M-C-\（indent-region）。这意味着indent-region命令通常是通过输入键序列：M-C-\ 来激活的（如果你愿意的话，可以改变激活这个命令的键序列，这称作“重新绑定”。参见16.11节，“键图”）。缩写 M-C-\ 意味着同时输入 META 键、CTRL 键和 \ 键。有时，像这样的一个组合键也叫做一个键和弦，因为它类似于在钢琴上演奏一个和弦。如果你的键盘没有 META 键，可以用前缀 ESC 键取代它。在这种情况下，M-C-\ 意味着按下并释放 ESC 键，然后同时按下并释放 CTRL 键和 \ 键。

如果用 GNU Emacs 的 Info 阅读这份文档，只用空格键就能翻阅整本书（可以输入 C-h i 然后选择 Info 来学习）。

关于术语的说明：当仅仅提到 Lisp 这个词时，常常是指各变种的Lisp；但是当提到 Emacs Lisp 时，就是特指 GNU Emacs Lisp 了。

致谢

感谢所有对这本书提供了帮助的人们。特别感谢 Jim Blandy、Noah Friedman、Jim Kingdon、Roland McGrath、Frank Ritter、Randy Smith、Richard M. Stallman 和 Melissa Weisshaus。同时要感谢 Philip Johnson 和 David Stampe耐心的鼓励。书中的所有错误都由我负责。

目 录