TURING 图灵程序设计丛书

Apress

Practical Common Lisp

实用Common Lisp编程

[美] Peter Seibel 著
田春 译

- 第16届Jolt生产效率奖图书
- 了解黑客青睐的编程语言
- 带你进入Common Lisp的精彩世界

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

Practical Common Lisp

实用Common Lisp编程

[美] Peter Seibel 著
田春 译

人民邮电出版社
北京

版 权 声 明

对本书的赞誉

"Peter Seibel 的 *Practical Common Lisp* 名副其实：对于那些想要学习并开始将 Common Lisp 用于实际工作的人来说，这是一本极佳的入门参考书。这本书写得非常好并且读起来很有趣，至少对于那些认为学习新语言很有趣的人来说是这样的。

"Seibel 在书中并未花大量时间来抽象地讨论 Lisp 在编程语言界的地位，而是选择了正确的切入点，通过一系列复杂度递增的编程示例来引导读者。这一思路把重点放在了那些有经验的程序员们最常使用的 Common Lisp 特性上，而没有纠缠于那些即便是专家也要通过查询手册才知道的 Common Lisp 语言特性的角落。Seibel 这一示例驱动思路所产生的结果就是，让读者准确地感受到了 Common Lisp 在以最小的代价构建复杂而具革命性的软件系统过程中所爆发的强大威力。

"在 Common Lisp 领域已经有许多采用了更具抽象和比较性的编写思路来创作的好书，但一本能够针对实际开发者讲解实现方式及相关原因的好书却是很有价值的，无论是对当前的 Common Lisp 用户，还是其他潜在用户来说都是这样。"

——Scott E. Fahlman，Carnegie Mellon 大学计算机科学研究教授

"这本书要是在我当初学 Lisp 的时候出版该多好。这并不是说当时没有其他关于 Common Lisp 的好书，但它们都不像本书这样务实和与时俱进。况且我们还不要忘了 Peter 讲述了一些在其他 Lisp 著作中被完全忽略的内容，诸如路径名、状况和再启动等主题。

"如果你初学 Lisp 并且想要选择一个正确的切入点，那么就不要犹豫，赶快买下本书吧。一旦你读过并学会了它，接下来就可以继续阅读 Graham、Norvig、Keene 和 Steele 等人的'经典'著作了。"

——Edi Weitz，Common Lisp Cookbook 的维护者和 CL-PPCRE 正则表达式库的编写者

"有经验的程序员可以从示例中学到极其有用的知识，而 Seibel 这本示例丰富的入门教材中用 Lisp 来讲解真是件令人高兴的事。尤其令人赞叹的是，这本书中包含的如此多的示例涵盖了当今程序员日常可能用到的众多问题领域，诸如 Web 开发和流媒体技术。"

——Philip Greenspun，*Software Engineering for Internet Applications* 的作者，MIT 电子工程和计算机科学学院

“这是一本涵盖了 Common Lisp 广泛内容的优秀书籍，其中通过演示一些读者可以运行和扩展的实际应用，阐述了 Common Lisp 许多独一无二的特性。这本书不仅说明了 Common Lisp 是什么，还说明了为什么每个程序员都应当熟悉 Lisp。”

——John Foderaro，Franz Inc.资深科学家

“Maxima 项目经常得到那些想要学习 Common Lisp 的潜在贡献者的垂询。我很高兴最终有一本书让我可以毫无保留地推荐给他们。Peter Seibel 那简明直接的风格使读者能够快速地领略 Common Lisp 的威力。他包含在书中的许多示例重点关注当代的编程问题，充分说明了 Lisp 绝不仅仅是一门学院派编程语言。这本书是该领域的一本极受欢迎的著作。”

——James Amundson，Maxima 项目负责人

“我喜欢那些分散在书中描述‘真实’和有用程序的实践性章节。我们需要这样一本书来告诉世界，使用 Lisp 可以轻松地将字符串和数字组装成树和图。”

——Christian Queinnec 教授，Universite Paris 6（Pierre et Marie Curie）

“学习一门编程语言最重要的一部分就是学习它正确的编程风格。这很难教授，但通过阅读这本书，你可以轻松地获取到。仅是阅读那些实践性的示例就足以让我成为能胜任各种语言的高级程序员。”

——Peter Scott，Lisp 程序员

“它提供了这门语言的全新视角，并且后面章节里的那些例子在你作为程序员的日常工作中也是很有用的。”

——Frank Buss，Lisp 程序员和 Slashdot 贡献者（www.slashdot.org）

“如果你对 Lisp 感兴趣是因为它跟 Python 或 Perl 有关系，并且想要通过实际动手而不只是观看来学习它，那么这将是一本极佳的入门书。”

——Chris McAvoy，Chicago Python 用户组（www.chipy.org）

“终于有一本为我们写的 Lisp 书了。如果你想要学习如何编写一个阶乘函数，那么这本书并不适合你。Seibel 的书是为专业程序员所写的，它更关注工程师和艺术家，而非科学家，并在解决易于理解的现实问题过程中优雅而精细地体现出了这门语言的威力。

“阅读大多数章节的感受就好像正在编写一个程序，一开始只知道很少的内容，然后越来越多，就像在搭建一个最终可以站在上面的平台那样。当 Seibel 在构建一个测试框架的过程中顺势引入宏的时候，我对如此一个简单的例子就让我能真正‘领会’它们的含义而感到震惊。书中的叙事性内容极其有用，这样的技术书籍更出类拔萃。恭喜你!”

——Keith Irwin，Lisp 程序员

“在学习 Lisp 的过程中，人们不知道一个特定函数的用途时会去查询 CL HyperSpec，但我发现如果只是阅读 HyperSpec 通常很难‘领会’其含义。当遇到这类问题时，我每次都会翻阅这本书，它是目前在教授编程方式方面最具可读性的来源，而绝不仅仅是平铺直叙。”

——Philip Haddad，Lisp 程序员

“在急速发展的 IT 行业，专业人员需要最强大的工具。这就是 Common Lisp 这种最强大、最灵活和最稳定的编程语言正获得广泛关注的原因。这是一本令人期待已久的书，它将帮助你驾驭 Common Lisp，以应对当今各种复杂的现实问题。”

——Marc Battyani，CL-PDF、CL-TYPESETTING 和 mod_lisp 的开发者

“不要认为 Common Lisp 只能用于数据库、单元测试框架、垃圾过滤器、ID3 解析器、Web 编程、Shoutcast 服务器、HTML 生成解释器和 HTML 生成编译器等领域，这些只是碰巧在这本书中被实现的几件事。”

——Tobias C. Rittweiler，Lisp 程序员

“当我遇到 Peter 时，他正在写这本书。我自问：‘为什么已经有了许多很好的入门书籍，他还要写另一本关于 Common Lisp 的书？’一年以后，我看到了这本新书的初稿，也意识到我最初的想法错了。这本书不是‘另一本’书。作者更关注实践方面而非语言的技术细节。我最初是通过阅读一本入门书学习 Lisp 的，我觉得我理解了这门语言，但也有这样的感觉：‘那又怎样？’这就意味着我完全不知道如何使用它。相反，这本书在用最初的几章讲解了最基本的语言概念以后就转向了‘实践’章节。读者会在跟随开发这些‘实用’项目的过程中逐渐学会更多的语言知识，同时这些项目将会合并成具有相当规模的产品。读完本书，读者会感到他们已经是专业的 Common Lisp 程序员了，因为他们已经‘完成’了一个大项目。我认为，Lisp 是唯一一门允许采用这种实践方式来介绍的语言。Peter 充分利用了这个语言特性，勾勒出对 Common Lisp 的有趣介绍。”

——Taiichi Yuasa 教授，京都大学计算机科学与通信学院

“本书展示了 Lisp 的威力，不仅是在传统领域，例如仅使用短短的 26 行代码就开发出一个完整的单元测试框架；而且还表现在一些全新领域上，诸如解析二进制 MP3 文件、构建浏览歌曲集的 Web 应用以及在 Web 上传输音频流。许多读者会很惊讶：Lisp 具有 Python 等脚本语言的简洁性、C++的高效性，以及设计自己语言扩展时无与伦比的灵活性。”

——Peter Norvig，Google 公司搜索质量组负责人，*Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp* 的作者

译 者 序

很荣幸，我被授权翻译 *Practical Common Lisp* 一书。本书是自 1994 年 Common Lisp 语言标准化以来，国内出版的第一本 Common Lisp 的中文教材。

Lisp 语言家族最早诞生于 1959 年，它是人类历史上第二个高级程序设计语言（第一个是 Fortran）。那一年，人工智能（AI）专家 John McCarthy 发表了具有重大历史意义的第一篇 LISP 论文“Recursive Functions of Symbolic Expressions and their Computation by Machine, Part I”，其中介绍了一种运行在古老的 IBM 704 计算机上的列表处理语言 LISP（LISt Processing，列表处理），借助它可以轻松描述当时人工智能领域用到的各种算法。从此，Lisp 语言在包括 AI 领域在内的所有主流计算机分支上，都获得了长足的发展。Lisp 平台不但在 IBM PC 出现之前的几乎所有计算机硬件体系上均有移植，甚至在 20 世纪 80 年代还出现过专门用来运行 Lisp 程序的硬件——Lisp 机。1994 年，ANSI 标准化的 Common Lisp 语言将之前历史上的所有现存 Lisp 厂商的各种语言和平台特性做了一次伟大的总结，从此语言核心不再变化，不但标准化以前的历史遗留代码只通过少量修改就可以兼容现代 Lisp 平台，而且标准化以后写出的所有新代码也都几乎不经任何调整就可能运行在任何一种 Common Lisp 平台上，无论是带有原生或是字节码编译器的，还是间接转译成 C 语言的，或是运行在 JVM 上的。目前至少有 13 种不同的 Common Lisp 语言平台可以运行在现代计算机上，其中 10 种还在广泛使用中，远超过它们所在的操作系统上 C 和其他语言编译器的数量。可以说，Lisp 语言家族长达 50 年的发展史就是整个计算机发展史的缩影。

我从 2003 年大学三年级时开始学习 Common Lisp 语言，至今已有八个年头。当时学习它的动机基本上是出于对人工智能（传统的逻辑和推理、知识表示等方向）的个人兴趣。不过随后很快就发现，Common Lisp 是一门通用的编程语言，如果不考虑其历史渊源而只从语言本身的特性来观察的话，可以说它跟人工智能毫无关系。在 *Practical Common Lisp* 一书中，作者 Peter Seibel 也谈到这个问题。当今有太多的人对 Lisp 语言存在类似的误解，包括相当多学过早期 Lisp 语言的人还停留在列表（List）是 Lisp 语言的唯一复合数据类型的认识上。如果读者从头到尾学完了这本书，就会发现 Common Lisp 是一门特性丰富的大型编程语言，不但提供了现代编程语言普遍支持的各种数据类型（包括各种数值类型、字符串、数组、结构体和哈希表在内），还支持几乎所有的编程范式（面向过程的、函数式的以及面向对象的），尤其带有一套特性丰富且思想独到的面向对象编程接口 CLOS（Common Lisp Object System）和 OO 扩展接口 MOP（Meta-Object Protocol）。如果要用一句话来描述 Common Lisp 中的 OO 与 C++/Java/Smalltalk 等语言的 OO 有

何不同，那就是 Common Lisp 对象系统完全不是基于消息传递的，而是基于广义函数的。有兴趣的读者应当仔细阅读本书的第 16 章和第 17 章，其中介绍了 CLOS 的一些入门内容。

不过 Lisp 语言最吸引人的地方还在于其与众不同的程序运行方式。从 C 语言一路学过来的人往往把一门语言的语法及其标准函数库视为语言的全部，因为一旦程序写好，编译器就会将整个代码编译成一个可执行程序或者被其他可执行程序使用的库。接下来语言本身是什么就不重要了，重要的是程序员写出了什么功能，甚至连编译器本身是什么都不重要，因为它只是一个黑箱，除了简单的优化开关之外几乎无法调整其行为。各种 Lisp 语言则采用完全不同的方式来运行 Lisp 程序：Lisp 平台本身是一个交互式的环境，它在很大程度上就是用其本身写成的。用户的 Lisp 代码以编译或解释的形式加载到 Lisp 环境中，然后跟 Lisp 语言或平台本身的代码直接融合在一起。换句话说，每一个 Lisp 程序都是对 Lisp 语言本身的某种形式的扩展。然后通过一个启动函数，整个程序得以运行。听到这里，读者似乎看到了 Python 或者 Ruby 的影子，但 Lisp 环境还有更绝的地方：几乎所有 Lisp 平台都允许用户将加载了用户代码的整个环境从内存中导出（dump）为一个磁盘文件。通过直接加载这个文件而不是默认的那个只含有 Lisp 本身的文件，可以迅速地重建导出前的 Lisp 环境，从而达到增量开发或者哪怕是快速加载已有 Lisp 程序的目的。最后，和其他语言很不同的一点是，Lisp 语言规范（至少 Common Lisp 是这样的）不但包括了如何定义某个程序组成部分（指的是变量、函数和类这些东西）的能力，还定义了从 Lisp 环境中清除任何程序组成部分以及就地修改它们的能力，并在语义和功能上确保了这些操作不会破坏运行中的 Lisp 代码。这导致了 Lisp 语言的另一个重要应用：通过加载补丁，Lisp 系统可以在运行中被任意修改，这对 24×7 的服务器端程序的平滑升级尤为有利。顺便说一句，Lisp 也是最早引入垃圾收集（GC）机制的编程语言，Lisp 环境中的任何对象，一旦失去了来自其他对象的引用，就会在某个时刻被 GC 系统从内存中清除掉。

读者可能已经注意到了我在不停地混用 Lisp 和 Common Lisp 两个概念。这有两层含义：首先，存在 Common Lisp 之外的 Lisp 语言，更准确地说是 Lisp 方言（dialect），至少包括了 Emacs Lisp、AutoLISP、Scheme、Racket（前身是 PLT Scheme）和 Clojure，其中最后一个是高速发展中的新兴 Lisp 方言；其次，所有 Lisp 家族的语言都有很多共性，除了上面描述中带有 Lisp 而非 Common Lisp 字样的部分以外，还有最大的也是初学者最容易看到的一点，那就是所有 Lisp 方言都使用前缀表达式和用小括号表示的列表，例如 1+1 在 Lisp 中将写成（+ 1 1）。很多初学者一开始都不适应前缀表达式，但我认为前缀表达式是有很多优点的：首先，它彻底消除了运算符结合性问题，令表达式毫无歧义可言；其次，它让语言处理器更加简单高效，避免了语法分析的困难。当然，一旦习惯了也就感觉没什么了。

学习本书对更好地使用其他 Lisp 方言无疑是大有帮助的。在翻阅书店里关于 AutoLISP（AutoCAD 计算机辅助设计软件的扩展语言）的各种书籍时，我经常痛心疾首地发现这些图书的作者虽然精通 AutoCAD 所提供的 Lisp 编程接口，但写出的 AutoLISP 代码要么极为难看，要么缺乏效率、滥用内存。AutoLISP 在语法上跟 Common Lisp 非常接近，本书的大部分内容都适用于 AutoLISP。因此我强烈推荐所有 AutoLISP 程序员阅读本书以加强自身的 Lisp 素养。同样的问题对于 Emacs Lisp（GNU Emacs 文本编辑器的扩展语言）来说也是一样的。Scheme 系的 Lisp 方

言区别相对大一些，如果连基本的变量和函数定义都在形式上完全不同的话（当然，思想上是没什么本质区别的），我恐怕初学者从本书中学得 Scheme 编程思想的机会不大，这种情况下还是推荐《计算机程序的构造和解释》、*Lisp in Small Pieces* 和 *The Little Schemer* 等书籍比较好。

本书可以作为其他 Common Lisp 语言教材的学习基础。在本书的最后一章里，作者给出了很多后续的教材，在此就不一一重复了。需要特别指出的是，另一本著名的 Common Lisp 教材 *On Lisp*（作者 Paul Graham，也就是《黑客与画家》一书的作者）多年前已经被我和我的几位朋友共同翻译成中文版，细心的读者可以从网上轻易地找到它。*On Lisp* 主要介绍 Common Lisp 的宏编程，这是 Common Lisp 区别于其他语言甚至其他 Lisp 方言的最重要特性。我相信一旦读者掌握了本书中关于宏的章节以后就可以阅读 *On Lisp* 中的进阶内容，从而将自身对编程语言的认识上升到一个新的高度，不过更加符合实用原则的思路还是先把本书读完。

Common Lisp 绝不是一门过时的编程语言，整个 Common Lisp 社区一直都在高速的发展之中，近几年的发展尤为迅速。在我学习 Common Lisp 的这些年里，我亲眼目睹了几个 Common Lisp 平台从无到有（ECL、ABCL）或者发展壮大（SBCL、Clozure CL）的过程。经典平台（CMUCL、MCL）也得到了良好的维护并始终跟进操作系统的自然发展。随着计算机硬件的高速发展，即便相对保守的 Common Lisp 商业平台也开始或即将开始支持对称多处理器（SMP），其中 LispWorks 和 Scieneer CL 都以 SMP 支持作为主要卖点。第三方软件包长足发展，虽然尚未达到 Perl 社区 CPAN 的水平，但常用的工具包一应俱全，其中不乏高质量的大型项目。近年来最新的成果 Quicklisp 包管理平台，更是将 Common Lisp 第三方软件包的安装过程提升到了前所未有的便捷程度。免费平台越来越好，商业平台依然昂贵，开源工具蓬勃发展，所有这些都暗示着 Common Lisp 语言还保持着旺盛的生命力，唯一的问题是如何让更多的国内计算机领域爱好者了解它。这就是我翻译本书的目的所在。

过去 8 年里，我一直活跃在国内和国际 Common Lisp 社区的前沿。我在大学本科的最后两年学完了 Common Lisp 语言语法的主要部分，读完了包括本书在内的几本最经典的 Lisp 书籍，并已经能够在当时最常见的 CMUCL 平台（CMU Common Lisp）上编写一些简单的程序。后来在网易工作的 5 年里，我在工作之余从头研究了一遍 Lisp 语言的发展史，亲身体会了包括 Lisp 机在内的十几种不同的 Common Lisp 平台或实现，并自费购买了价值数千美元的商业开发环境 LispWorks，拥有三种主流操作系统上的 License。在网易从事 Linux 系统管理工作期间，我用 Common Lisp 从头实现了一万行源代码规模的 SNMP 简单网络管理协议工具包，它可以为任何服务器端 Common Lisp 程序添加通过 SNMP 协议进行远程监控的能力，也可以作为基于 Common Lisp 的网络监控系统的基础。我还在过去 3 年里参与维护了 Common Lisp 社区两个最重要的可移植网络库之一 usocket，并由于 SNMP 库的需要将其从原本只支持 TCP 扩展到了同时支持 UDP，其中对于 LispWorks 的 UDP 支持代码是完全从头写的，因为官方并不支持。2009 年，我向国际 Lisp 会议的投稿被接受，并作为会议论文集的一部分出版。我是长期担任水木社区函数型编程版块的板主之一，专门负责 Lisp 方向的讨论和技术分享。2011 年 7 月，我离开网易以后开始全职从事商业 Lisp 软件相关的开发工作。可能我还不是一个很好的译者，但作为一个经验丰富的 Common Lisp 程序员，我相信自己翻译这本书是合适的。

计算机领域每天都在高速发展，新语言和新技术的产生速度早已超过了一般人的学习速度。对于一个计算机领域的从业人员或爱好者来说，学习通常是为了更好地应用，把所有时间都用来学习而无暇具体应用也是本末倒置。在这种情况下，有选择地学习最有用、最不易变质的知识，以及甄别各种计算机知识的重要程度和相互关系的能力就显得非常重要了。从计算机语言的发展历史来说，如果一门语言可以存活 50 年，那么它的内在生命力很可能保证其继续长期存活下去，一个人用这门语言写下的代码也将比其他语言的代码更有可能长久地造福后人。

总之，希望这本书能将读者顺利带入 Lisp 领域。学习一门新的语言总是要花些成本的，但我想说，和其他任何语言相比，花在理解 Lisp 上的时间和精力将绝对是物超所值的，即便相当多的读者可能没有机会在短期内将 Lisp 用于他们的日常工作。之所以这样说是有原因的：C 和 Lisp 是编程语言的两个极端，大多数人已经熟悉了 C 的那一端，但如果他们还熟悉另一端的话，那么迅速理解几乎所有其他的编程语言将不再是问题。

致 读 者

亲爱的读者：

Practical Common Lisp，这难道不矛盾吗？你可能和大多数程序员一样，知道一点儿 Lisp：要么是来自大学里的计算机科学课程，要么是学习了足够多的 Elisp 以定制 Emacs。或者可能你只是因为有人正喋喋不休地谈论 Lisp，而其认为这是一门历史上最伟大的语言。但是，你可能从未想过会在同一本书的书名里看到 Practical 和 Lisp。

但你确实已经在读这样一本书了，你一定还想知道更多。也许你相信学习 Lisp 将使你成为一名能够胜任所有语言的高级程序员。或者可能你只是想要知道那些 Lisp 狂热者们总是在喊些什么。或者可能你已经学了一些 Lisp，但还没有足够的能力去用它编写感兴趣的软件。

如果确实出现上述情形中的任何一种，那么本书就很适合你。Common Lisp 是一种 ANSI 标准化的工业级别的 Lisp 方言，通过使用它，我将向你展示如何编写那些远远超越了愚蠢的学术训练或简单编辑器定制的实际的软件。并且我将说明即便在其许多特性已被其他语言所采纳以后，Lisp 仍然有其独到之处。

与许多 Lisp 书籍所不同的是，本书不会只触及一些 Lisp 的伟大特性而让你自己思考如何使用它们。我讨论了你在编写实际程序中用到的所有语言特性，并用了多于三分之一的篇幅来开发具有一定复杂度的软件——基于统计的垃圾过滤器、用来解析二进制文件的库，以及一个带有完整在线 MP3 数据库和 Web 接口的通过网络流式传输 MP3 的服务器。

现在就翻开它，看看这门人类发明的最伟大的语言是多么的实用吧。

此致

Peter Seibel

致　谢

如果不是因为一些愉快的巧合，本书不会写出来，至少作者不是我。因此，我必须首先感谢Franz的Steven Haflich。我们在Bay Area Lispniks见面会上相遇时，他邀请我和Franz的一些销售人员共进午餐，席间我们都认为需要有本Lisp新书。其次我必须感谢Steve Sears，午餐中的一名销售人员，他随后将我引荐给Franz的总裁Fritz Kunze。Fritz曾经提到他正想请人来写一本Lisp书。当然，也非常感谢Fritz说服Apress出版社出版一本由我来写的新Lisp书。Apress出版社在整个过程中给予了我鼓励和帮助。也要感谢Franz的Sheng-Chuang Wu，提供了诸多帮助。

我在撰写本书时最不可或缺的资源是comp.lang.lisp新闻组。comp.lang.lisp的忠实成员们不厌其烦地回答了有关Lisp及其历史的各种问题。我也经常翻阅该新闻组的Google存档，它是技术资料的宝库。因此，感谢Google提供了这一服务，也感谢所有comp.lang.lisp参与者们一直以来提供各种内容。特别地，我要感谢两位长期的comp.lang.lisp贡献者——Barry Margolin和Kent Pitman。Margolin在我阅读该组资料时一直在提供各种琐碎的关于Lisp历史和他个人经验的资料；而Pitman除了作为一位语言标准的首席技术编辑和Common Lisp HyperSpec的开发者以外，还在comp.lang.lisp上撰写了成千上万字的帖子以阐述语言的多种方面及其来源。

在撰写本书时，其他不可或缺的资源包括用于PDF生成和排版的Common Lisp库以及由Marc Battyani编写的CL-PDF和CL-TYPESETTING。我使用CL-TYPESETTING生成用于我个人编辑工作所用的PDF文件，并将CL-PDF作为我用来生成本书中线条图的Common Lisp程序的基础。

我还想感谢许多在网上阅读了本书部分草稿，并通过电子邮件指出笔误、提出问题或简单给出建议的人们。尽管无法提及所有人的名字，但对于一些提供了有价值的反馈的人要在这里特别指出：J. P. Massar（Bay Area Lispnik成员，他在几次比萨午餐上极大地激发了我的灵感）、Gareth McCaughan、Chris Riesbeck、Bulent Murtezaoglu、Emre Sevinc、Chris Perkins以及Tayssir John Gabbour。我的几个非Lisp相关的朋友也参与审阅了某些章节，感谢Marc Hedlund、Jolly Chen、Steve Harris、Sam Pullara、Sriram Srinivasan和William Grosso提供反馈。另外要感谢Scott Whitten允许我使用图26-1中的那张照片。

我的技术审稿人Steven Haflich、Mikel Evins、Barry Margolin以及我的文字编辑Kim Wimpsett，他们以数不清的方式提升了本书品质。当然，如果书中还有任何错误，那都是我的责任。同时也感谢Apress出版社其他参与本书出版工作的人们。

最后，也是最重要的，我要感谢我的家庭。感谢父母所做的一切，以及总是坚定地相信我可以完成本书的我的妻子Lily。

排版约定

诸如 foo 这样的内嵌文本是代码，它们通常是函数、变量和类等元素的名字，这些名字要么是我已经引入的，要么是我将要引入的。由语言标准所定义的名字会写成这样：**DEFUN**。更大块的示例代码的样式将会如下所示：

```
(defun foo (x y z)
  (+ x y z))
```

由于 Common Lisp 的语法以其规范性和简洁性著称，因此我采用简单的模版来描述各种 Lisp 形式[①]的语法。例如，下面描述了 **DEFUN** 的语法，它是标准的函数定义宏：

```
(defun name (parameter*)
  [ documentation-string ]
  body-form*)
```

这些模版中以斜体书写的名字需要填入我将在正文中描述的具体名字或 Lisp 形式。斜体名字后跟的星号（*）则表示该名字出现了零次或更多次。而位于方括号（[]）内的名字代表可选的元素。偶尔也会出现以竖线（|）分隔的替代内容。模版中的所有其他内容（通常只是一些名字和括号）都是一些将会出现在 Lisp 形式中的字面文本。

最后，由于跟 Common Lisp 之间的许多交互都发生在交互性的“读-求值-打印”循环（即 REPL）中，所以将经常如下显示在 REPL 中对 Lisp 形式求值的结果：

```
CL-USER> (+ 1 2)
3
```

其中的 CL-USER>是 Lisp 提示符，其后总是跟着需要求值的表达式，在本例中是(+ 1 2)。求值的结果和生成的其他输出都会显示在接下来的几行里。有时，我也会在表达式后使用→来表示求值的结果，就像下面这样：

```
(+ 1 2) → 3
```

偶尔还会使用等价符号（≡）来说明两个表达式是等价的，就像这样：

```
(+ 1 2 3) ≡ (+ (+ 1 2) 3)
```

---

① Lisp 形式（Lisp form）是 Lisp 特有的一种语法构造，本书在不致引起歧义的上下文里，也会将其简称为“形式”。——译者注

目　录