Broadview
WWW.BROADVIEW.COM.CN

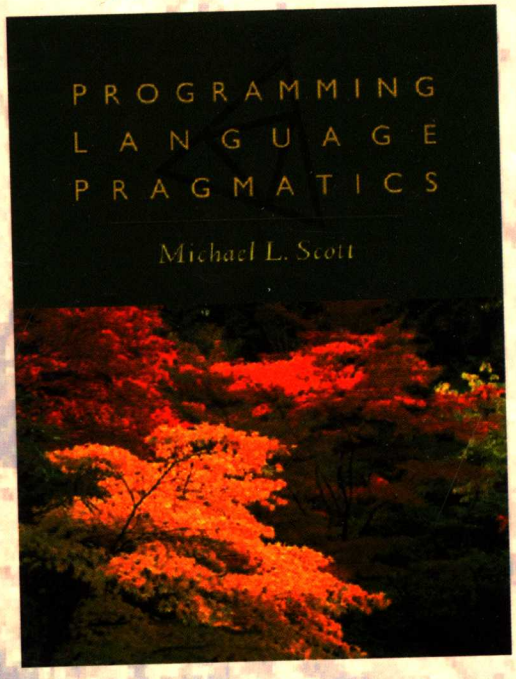
MK

# 程序设计语言

# PROGRAMMING LANGUAGE

## ——实践之路——

## PRAGMATICS




[美] Michael L. Scott 著

裘宗燕 译

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

# 程序设计语言——实践之路

PROGRAMMING LANGUAGE PRAGMATICS

[美] Michael L. Scott 著

裘宗燕 译

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

## 内 容 简 介

这是一本很有特色的教材，其核心是讨论程序设计语言的工作原理和技术。本书融合了传统的程序设计语言教科书和编译教科书的有关知识，并增加了一些有关汇编层体系结构的材料，以满足没学过计算机组织的学生们的需要。书中通过各种语言的例子，阐释了程序设计语言的重要基础概念，讨论了各种概念之间的关系，解释了语言中许多结构的形成和发展过程，以及它们演化为今天这种形式的根源。书中还详细讨论了编译器的工作方式和工作过程，说明它们对源程序做了什么，以及为什么要那样做。书的每章最后附有复习题和一些更具挑战性的练习。这些练习的特别价值在于引导学生进一步深入理解各种语言和技术。

本书在美国大学已使用了十余年，目前被欧美许多重要大学用于“程序设计语言”或者“软件系统”课程。本书适合高年级本科生或者一年级研究生使用，许多内容对专业程序员也很有价值。本书作者 Michael L.Scott 是计算机领域的著名学者，译者是北京大学的裘宗燕教授，他熟悉专业，译笔流畅，因此，这是一本难得的著、译双馨的佳作。



版权贸易合同登记号　图字：　01-2004-0342

**图书在版编目（CIP）数据**

程序设计语言：实践之路 /（美）斯科特（Scott,M.）著；裘宗燕译．—北京：电子工业出版社，2005.3
书名原文：Programming Language Pragmatics
ISBN 7-121-00900-5

I.程...　II.①斯...②裘...　III.程序语言－教材　IV.TP312

中国版本图书馆 CIP 数据核字（2005）第 006915 号

责任编辑：郑兆昭　方　舟
责任校对：陈元玉
印　　刷：北京智力达印刷有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
经　　销：各地新华书店
开　　本：787×980　1/16　印张：57　　字数：1000 千字
印　　次：2005 年 3 月第 1 次印刷
定　　价：88.00 元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系电话：（010）68279077。质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

# 译者序

Michael Scott 的《Programming Language Pragmatics》是一本很有趣，也非常有价值的新教科书，它颠覆了传统意义上的“程序设计语言”课程的组织体系，其内容涵盖程序设计语言、编译技术、软件系统的许多方面，甚至延伸到硬件体系结构等许多领域。实际上，出现这一情况的根源也很明显：程序设计语言在计算机科学技术领域居于一种中心地位。程序是计算机科学技术中最核心的概念，而作为描述程序的语言，集中表现了人们由程序设计和软件开发实践中获得的最有价值、最具普遍性的认识和技术。程序设计语言下接硬件体系结构，上承丰富多彩的计算机应用需求，反映了开发者专业能力的发展和局限性，又受到实现的理论和技术的制约。这样，程序设计语言中，自然浓缩了许多相关领域中的知识和技术精华，要理解其发展和演化的现状和趋势，一定会涉及与之相关的各个领域。本书作者对与语言有关的众多领域有着深刻的认识，在其中纵横驰骋，为我们展现了一幅生动、全面，而又非常深刻的画卷。

本书系统地介绍了程序语言领域的各种基本概念，如语法和语义，数据、操作和控制，类型和抽象等等，介绍了许多有关语言处理的知识，不同的语言范型，以及相关的理论和实践情况。在学习各种语言特征时，我们还能看到今天人们对许多语言特征的评价和反思，了解为什么一些特征的设计在不断变化，看到理论和技术发展对语言形态和细节的影响。与此同时，本书还深入介绍了这个领域里的许多新发展、新问题和新技术。例如，作者用了很长的一章深入探讨了面向对象语言的问题，不仅介绍了这类语言的外在形式特征及其价值，还特别仔细地讨论了这类语言中各种新的重要机制的实现技术，例如动态方法约束，多重继承等等。书中还用很大篇幅讨论程序的静态连接和动态连接，帮助读者理解高级语言的加工和执行。作者在书中既强调了重要的概念和理论，也特别重视语言的各方面实现技术，还深入探讨了实现技术发展进步对语言的影响。应该看到，语言实现方面的许多技术都是最重要的程序技术，作者的这些想法也使本书成为一部很有价值的软件技术书籍（有趣的是，作者确实用这些材料讲授一门名为“软件系

统”的课程）。

总而言之，本书在许多方面有鲜明的特色，是最新的“程序设计语言”教科书的代表。正因为这样，它出版的时间虽然不长，就已经被世界各国的许多重要大学选为“程序设计语言”或相关课程的教科书或者最重要的参考书。本书不仅值得计算机专业的本科生或者研究生学习或阅读，也值得在计算机领域中的实践工作者们阅读。对本书的学习能帮助我们理解现有的各种程序设计语言，大大提高学习和掌握新语言的能力，还能帮助我们看清隐藏在高级语言的各种机制背后的秘密，明白各种重要语言特征的价值、缺陷和使用它们的代价。对程序设计语言的深入理解，对于计算机专业工作者深刻理解相关的理论和实践，灵活高效地运用程序语言和相关工具，都可能起到非常重要的作用。

我非常赞赏电子工业出版社引进这本很有价值的著作，也很高兴能在把本书介绍给国内读者方面做一些力所能及的工作。我要感谢出版社周筠女士的远见卓识和以她为首的编辑团队的辛勤工作，还要感谢夏萍和丛欣对我的一贯支持和帮助。本书涉及的领域广泛，其中许多都不是我的专长。虽然我在翻译过程中已经尽可能地查阅了一些相关材料，但书中难免还会留下许多缺陷，希望得到同行和读者的指教。

**裘宗燕**

**2005 年元月，于北大**

# About the Author
# 关于作者

Michael L. Scott 是罗切斯特大学计算机科学系的教授，1996 到 1999 年任系主任。1985 年他由麦迪逊的威斯康星大学获得博士学位，在那里他是 Crystal 和 Charlotte 研究组的成员。他的研究兴趣在并行和分布式计算，包括操作系统、语言、体系结构和工具。他是 Lynx 分布式程序设计语言的设计者，与他人合作设计了 Charlotte 和 Psyche 并行操作系统、Bridge 并行文件系统、Cashmere 分布式共享存储系统。他与 Rice 大学的 John Mellor-Crummey 合作设计的 MCS 互斥锁被用在许多研究性的和商业性的系统里。

Dr. Scott 是 IEEE、ACM、Concerned Scientists 联合会和社会责任计算机专业工作者协会的会员，他曾为很多程序委员会和研讨会服务，作为负责人或协作研究者参与了来自 NSF、ONR、DARPA、NASA、国防部、福特基金会和数字设备公司（现在 Compaq）资助的许多项目。他对有关计算机科学的 GRE 高级考试做了许多贡献，是超过 50 篇经过评审发表的论文的作者。他在 1983 年获得贝尔实验室的博士奖学金，1986 年获得 IBM 的开发奖。

# Preface
# 前言

关于计算机程序设计的课程给了普通学生有关计算机领域的第一个印象。大多数学生在这样的课程之前对于计算机已经有了一些接触，例如以计算机游戏或者其他个人应用的形式，在他们还没有写出自己的程序之前，就已经开始意识到这些应用的工作方式了。在获得了作为程序员的一定能力之后（假定已经学过很好的有关数据结构和算法的课程），很自然的下一步就是想知道程序设计语言是如何工作的。本书将对此提供一个解释。

在常规的有关“系统”的教学计划里，数据结构（或者再加上计算机组织）之后的内容被分门别类归入属于一些子领域的一批课程，如程序设计语言、编译、计算机体系结构、操作系统、数据库管理系统，可能还有软件工程、图形学或者用户界面系统。这种安排方式存在一个缺点：有关计算机科学的最有趣的东西，许多都处于这些子领域的边界上。例如，RISC 革命推动了计算机体系结构和编译器构造之间的结盟，微内核的出现使得操作系统和语言运行库之间的界限变得模糊起来，基于 Java 的系统以类似形式模糊了编译器和运行库之间的分野。超级计算机里功能强大的存储器系统正在重新定义操作系统、编译器和硬件之间的相对作用。而程序设计语言的设计也始终受到实现问题的重要影响。越来越多的教育工作者和研究者逐渐认识到关注这些相互关系的重要性。

分门别类的教学计划的另一个问题是提供的课程太多，本科学生没有足够时间去学完它们。如果一个学生希望在理论、人工智能、数值方法或者其他独立领域里打下坚实的基础，那么就可能没时间去上 5 门系统方面的高级课程。我的认识是，与给予学生对两个或者三个子领域的深入讨论相比，集中提供一些横跨这些子领域的最基本材料可能更有意义。

《程序设计语言——实践之路》一书的核心，就是讨论程序设计语言如何工作的问题。从某种角度看，它是程序设计语言和编译的传统教科书的混合，再加上一些有关汇编层体系结构的材料，以满足那些没有学过计算机组织的学生的需要。它不是综述性的语言教科书，其中没有列举许多不同语言的细节，而是集中关注学生们可能遇到的作为所有语言之基础的一批概念，并通过各种语言的例子阐释这些概念。它也不是一本有关编译器构造的教科书，其中没有去解释如何构造一个编译器（那只是很少数的程序员最终需要整个参与的一种工作，虽然许多其他工具也会使用其前端技术），而是解释编译器如何工作，它对源程序做了什么事情，以及为什么要那样做。语言的设计和实现在这里被放在一起考察，其中特别强调它们之间相互作用的各种方式。在讨论迭代的时候（6.5.1 节），我们可以看到语义问题（什么是指标变量的作用域？如果循环的体要修改循环指标或者界的时候会出现什么问题？）与实际的问题（循环的每次迭代中必须执行多少条分支指令？更新指标时如何避免算术溢出）相互作用，造就了循环结构的演化和发展。在讨论面向对象的程序设计时，我们可以看到在语义优雅与实现速度之间的紧张关系，看到它怎样影响着如 Smalltalk、Eiffel、C++和 Java 等语言的设计。

在典型的本科生教学计划里，我们希望将本书用于程序设计语言课程。与其他教科书相比，这本书中稍微少了一点综述风格的细节，但也覆盖了同样广泛的语言和概念，并包含了更多有关实现问题的信息。对于那些在语言设计方面有强烈兴趣的学生，可以鼓励他们去进一步学习形式语义、类型理论或者面向对象的设计方面的课程。对那些在语言实现方面有着强烈兴趣的学生，应该鼓励他们去学习编译器构造方面的进一步课程。有了这本书作基础，有关编译器的课程就可以把更多的时间用在代码生成和优化方面，这些问题也是今天的大部分有趣工作之所在。

在 Rochster 大学里，本书的材料已经被使用了十多年，用于教授一门名为“软件系统”的课程。这一课程吸引着中高年级的本科生和一年级的研究生。本书对于专业程序员和其他实际工作者也同样很有价值，如果他们希望对自己所喜爱的程序设计语言“背后”的情况有一个更好的理解。通过把有关语法、语义和实际（实现）方面的问题集中到一起讨论，本书企图对语言的设计问题提供一个比其他教科书更完整和平衡的处理，希望能帮助学生理解为什么各种语言特征被设计成现在的样子，使程序员能针对具体应用选择合适的语言，更容易学好新的语言，更清晰高效地使用任何语言。

在多数章的最后作为总结的一节里，我们都回到了有关设计与实现的问题，其中特别强调两者在前面各节里所表现出的相互作用。此外，附录 B 列出了一个表，总结了这些相互作用，其中还包括了讨论它们的章节索引。这些相互作用被分为几类，包括今天的大多数语言设计师认为是错误的语言特征，至少部分原因是实现很困难；一些可能有用的特征在有些语言里没有，主要是考虑它们的实现可能很困难或者很慢；某些语言特征被引入，至少部分地是由于它们能高效而优美地实现等等。

有些章（第 2、4、5、9 和 13 章）比其他的章更着重强调实现问题。这些章与其他更多关注语言设计的章之间的顺序可以在一定限度内调整，但一定要保证第 5 章或者与之相当的内容出现在第 6、7、8 章之前。许多读者可能已经熟悉了第 5 章的大部分或者全部材料，多半是来自一个有关计算机组织的课程。在这种情况下就可以简单地跳过这一章。请注意，后面各章都假定了对于新型（即 RISC）微处理器的汇编层体系结构的理解。有些读者可能已经熟悉了第 2 章的一些材料，可能是来自一门有关自动机的课程。在这种情况下，该章的内容可以很快读过去，只是可能要在某些实际问题上（例如从语法错误中恢复）多停留一下。

作为自学，或者作为一个整整一年的课程，我建议从头到尾使用这本书。在 Rochster 作为一学期的课程时，我们也覆盖了本书的大部分内容。课程里集中讨论教师们从下面章节里选出的材料：第 1 章，1.2 节到 2.2.3 节，第 3、4、6、7、8 章，9.1 到 9.3 节，第 10 到第 12 章。同时要求学生阅读本书里所有材料，除了标星号的各节。还要求他们跳过第 5 章，主要是因为已经学过有关计算机组织的课程。

如果作为更传统的程序设计语言课程，那么可以排除 2.2 节和第 4、5、9 章，减少对其他章节里与实现有关的材料的重视程度，把更多时间用到仔细考察语义问题和不同的程序设计范型（例如，第 11 章里的一些理论基础材料）。对那些采用 4 学期制的学校，一种明显选择方案是开一学期的导引性课程和两个随后的选修课程。导引性课程可以覆盖下面章节：第 1 章，2.1 节到 2.2.3 节，第 3、6、7 章，8.1 节到 8.4 节。后一学期的面向语言的课程可以覆盖 8.5 节到 8.6 节，第 10 到 12 章，以及另外的有关形式语义学、类型系统或其他相关论题的补充材料。后一学期面向编译器的课程可以包含 2.2.4 节到 2.3 节，第 4、5（如果需要）、9 和 13 章，以及某些有关自动代码生成、更富进取性的代码优化、程序设计工具等等方面的材料。对这种安排的反对意见可能是认为它没有把有关面向对象、函数式和逻辑式程序设计的内容包括在导引部分。另一种选择就是从更广更具针对性的基于设计的观点出发，把 1.4 节到 1.6 节和 2.2.1 节到 2.2.3 节移到面向编译器的课程中，减少第 6 章到第 8 章里与实现有关的材料，加入 10.1 节到 10.4 节和 10.6 节，以及第 11 章里理论基础之外的部分。

我假定典型的读者已经对至少一种高级命令式程序设计语言有了相当的经验。具体是哪种语言并没有关系。书中例子取自各种不同的语言，但总是带有足够的说明和其他讨论，使不熟悉该语言的读者也能比较容易地理解它们。在需要时，算法都是采用自明的非形式的伪代码给出。真正的程序设计语言代码用的字体是 this font（Computer Modern），伪代码用的字体是 this font（UniversLight）。

每章最后都有一些复习题和一些更具挑战性的练习。这些练习的特别价值在于引导学生理解各种语言或者技术，其中许多都是他们不大会在其他地方遇到的，或者不会很快遇到的。我建议程序设计作业用 C++或者 Java；Scheme、ML 或者 Haskell；以及 Prolog。布置一个有关异常处理的题目也是非常好的想法，它可以用 Ada、C++、Java、ML 或者 Modula-3 写。如果课程里包含了并行性，作业应该在 SR、Java、Ada 或者 Modula-3 里给，可以根据本地的条件选择。在附录 A 里给出了各种语言实现的资源信息。

除了这种小型课题之外（或者在那些希望的地方），教师还可能希望学生做一些语言实现方面的工作。由于从空白开始做一个小编译器也是一个学期的工作，Rochester 采用的方式是给学生提供能工作的编译器的代码，要求他们做些修改。对于其中的许多人，这是他们第一次阅读、理解和修改一个大的实在的程序——就其本身而言也是非常有价值的练习。Rochester 的 PL/0 编译器把一个归功于 Wirth [Wir76，307-347 页] 的小语言翻译到 MIPS I 汇编语言，这一汇编语言被广泛认为是商品的 RISC 指令集中“最友好的”。威斯康星大学计算机系提供了一个非常好的 MIPS 解释器（“SPIM”，www.cs.wisc.edu/~larus.spim.html）。编译器本身可以从 Rochester 得到（ftp://ftp.cs.rochester.edu/pub/packages/plzero/）。它是用 C++写的，仔细划分了各个编译阶段，并有很详尽的文档。

## 有关封面

封面上的三角形图标是想表示程序设计语言设计中语法、语义和语用三个方面，其中的每一个都需要在另外两者的上下文中找到自己的意义。本书的目的是想对所有这三个方面给以同等的关注，它的取名来自其他有关语言的教科书关心得最少的那个方面。

## 致谢

许多人对于本书有很大贡献。我要感谢许多提出了评语和建议的审阅人，包括 Greg Andrews、John Boyland、Preston Briggs、Steve Muchnick、David Notkin、Ron Olsson、Constantine Polychronopoulos，以及另一些匿名评阅人。我已经为回应他们的关注而做了所有可能做的事情，剩下的错误都完全由我本人负责。我还要感谢 CSC 2/254（原来是 2/254）的学生们，他们甘愿作为本书早期版本的试验品，其反馈意见使得本书大为改观。这里需要特别感谢 Donna Byron 和 Brandon Sanders 的广泛评注，Scott Ventura 做的 12.2.3 节的互联网浏览器实例，以及 Angkul Kongmunvattane 帮我理解了有用的和使人分心的交叉索引之间的差异。

我还要衷心感谢 Morgan Kaufman 出版社里那些尽心工作的编辑人员，特别是 Denise Penrose、Mike Morgan、Edward Wade 和 Meghan Keeffe。Rochester 大学及其计算机系在本书写作期间为我提供了鼓舞和支持的学术环境。Sarada George 和 Dianne Reiman Cass 花了很多天时间去寻觅各种参考文献。最后还要感谢我的家庭 5 年的耐心等待。

本书最初的手稿是在一部 Apple Powerbook Duo 计算机上用 Marc Parmet 移植的 `emacs` 文本编辑器和 Andrew Trevorrow 的 OzTEX 版本的 TEX 撰写出来的。图形采用 Adobe 的 Illustrator。

# Contents

# 目 录