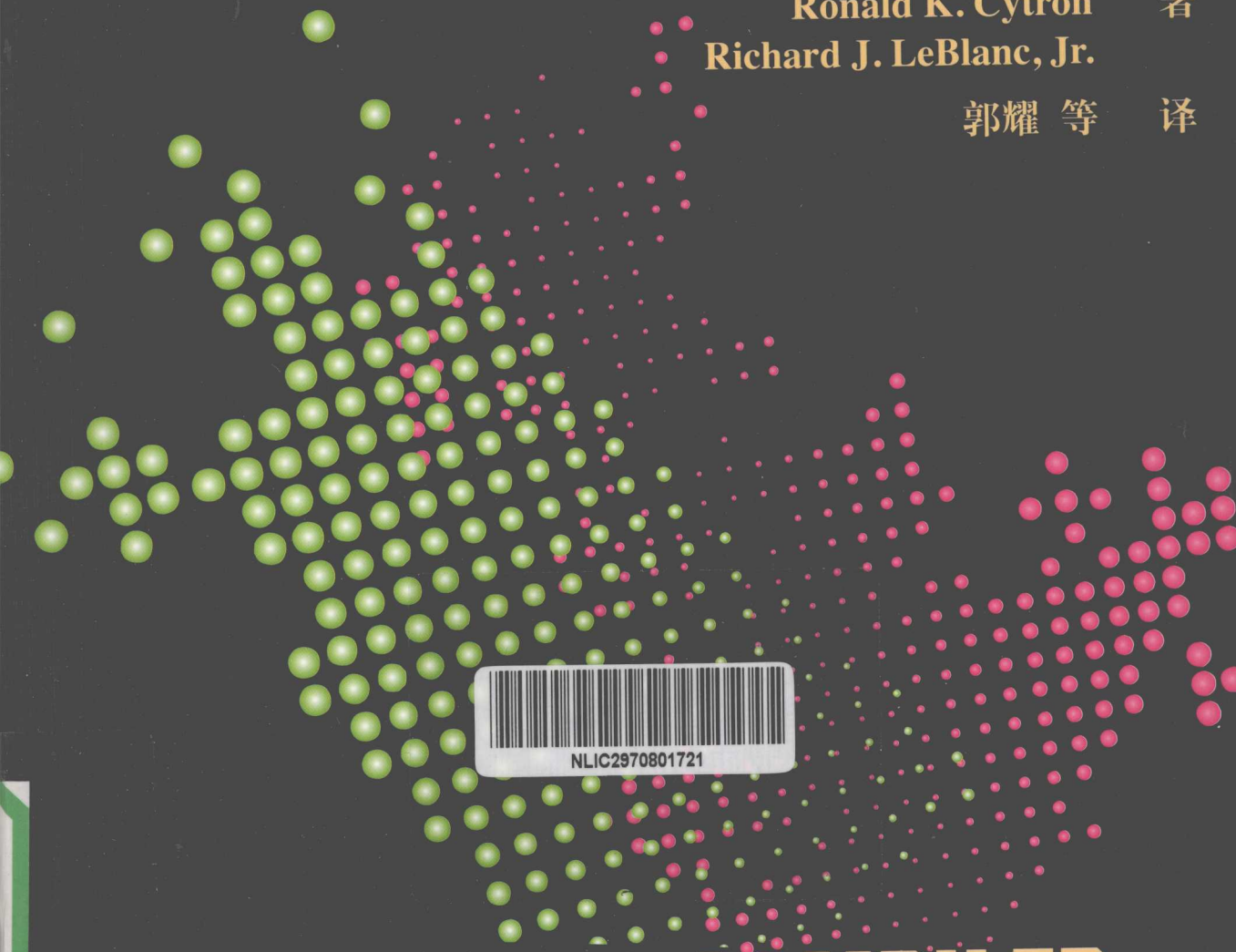
世界著名计算机教材精选

PEARSON

编译器构造

Charles N. Fischer
Ronald K. Cytron
Richard J. LeBlanc, Jr.
著

郭耀 等 译



CRAFTING A COMPILER

PEARSON

华大学出版社

世界著名计算机教材精选

编译器构造

Charles N. Fischer
Ronald K. Cytron
Richard J. LeBlanc, Jr. 著

郭 耀 等译

清华大学出版社
北 京

EISBN: 0136067050

图书在版编目(CIP)数据

编译器构造/(美)费希尔(Fischer, C. N.),(美)赛特朗(Cytron, R. K.),(美)莱比兰克(LeBlanc, R. J.)著;郭耀等译. --北京:清华大学出版社, 2012.5
(世界著名计算机教材精选)
书名原文:Crafting a Compiler
ISBN 978-7-302-28104-7

Ⅰ. ①编… Ⅱ. ①费… ②赛… ③莱… ④郭… Ⅲ. ①编译器—教材 Ⅳ. ①TP314

中国版本图书馆 CIP 数据核字(2012)第 031351 号

责任编辑:龙啟铭
封面设计:傅瑞学
责任校对:白 蕾
责任印制:杨 艳

出版发行:清华大学出版社
网 址:http://www.tup.com.cn, http://www.wqbook.com
地 址:北京清华大学学研大厦 A 座 邮 编:100084
社 总 机:010-62770175 邮 购:010-62786544
投稿与读者服务:010-62776969, c-service@tup.tsinghua.edu.cn
质 量 反 馈:010-62772015, zhiliang@tup.tsinghua.edu.cn
印 刷 者:北京密云胶印厂
装 订 者:北京市密云县京文制本装订厂
经 销:全国新华书店
开 本:185mm×260mm **印 张**:29.5 **字 数**:694 千字
版 次:2012 年 5 月第 1 版 **印 次**:2012 年 5 月第 1 次印刷
印 数:1~3000
定 价:59.00 元

产品编号:026159-01

译 者 序

编译原理与编译器构造方法是计算机科学技术专业本科生的必修课程。一本优秀的编译教材会让学习计算机的大学生受益终生。

本书就是这样一本非常优秀、并且有鲜明特色的编译教材。本书的前身是1988年出版的同名经典教材《编译器构造》以及1991年出版的《编译器构造：C语言描述》和《编译器构造：Ada语言描述》。本书的之前3个版本均在美国许多大学被当做本科生教材而广泛使用。本书的几位原作者在三所美国大学拥有长达25年的编译器教学经验，根据他们的经验在本书中对编译器构造的基本知识与关键技术进行了全新的讲解。

在此前版本的基础之上，本书结合最新的程序语言和编译技术的发展，不仅引入了目前流行的Java语言作为编译器的分析对象，同时更创新性地采用了面向对象的设计模式来组织编译器中的数据结构，特别是对包括抽象语法树在内的各种不同中间表示形式的存储与访问都采用了visitor（访问者）设计模式来实现。设计模式的采用可以在很大程度上降低编译器构造的复杂程度，使初学编译器的读者能更加容易地上手实现自己的编译器。

本书的主要内容包括编译器历史和概述、词法分析（扫描）、语法分析（包括自顶向下和自底向上的分析）、语法制导翻译、符号表和声明处理、语义分析、中间表示形式、虚拟机上的代码生成、运行时支持、目标代码生成和程序优化等。与其他的编译器教材相比，本书具有如下主要特点：

- 本书全面使用了伪码的风格来描述算法，这种伪码方式不仅可以简洁地对算法进行描述，同时也支持更加清晰地对算法的目的和结构进行讨论。
- 在内容方面，本书不仅精简掉了一些过时的内容（比如属性文法），同时对于在实践中常用的技术（比如抽象语法树的构造和使用）提供了更加详尽的讲解。
- 本书在算法中使用了面向对象的设计模式，并且在算法的讲解中引入了现代软件实践中的常用方式。
- 在代码生成技术中，本书不仅讲解了传统的为目标机器生成代码的技术，同时还引入了为虚拟机生成目标代码的方法，给教师和读者提供了不同的选择。

本书提供了详尽清晰的算法，鼓励读者在实践中学习编译器构造的相关技术，同时提供了配合教材使用的教学网站、参考资料以及源代码下载。本书不仅可以作为计算机专业本科生或研究生的参考教材，同时也适合相关领域的软件工程师、系统分析师等作为参考资料。

译者感谢本书英文版的作者，特别是美国华盛顿大学（圣路易斯）的Ron Cytron教授，他多次为译者解答关于书中的疑点。译者同时要感谢本书中文版的编辑，清华大学出版社的龙启铭编辑为译者提供的帮助。

作为北京大学计算机系本科生《编译原理》课程的主讲教师，译者在翻译过程中一方

面尽量尊重原著，同时也力争采用目前国内更为常用的术语来解释教材中遇到的核心概念。另外，在翻译过程中，译者在书中发现近百处语言和逻辑上的小错误，大部分已经过原作者确认，并反映在了这个译本中。

面对这样一本大部头的编译教材，译者在翻译过程中尽力为读者提供一个尽量完美的译本，然而错误与疏漏在所难免，敬请读者批评指正。

郭 耀

2012 年 3 月于北京大学

前　言

自 Fischer 和 LeBlanc 合著的《编译器构造》（Crafting a Compiler）一书于 1988 年出版以来，这个世界已经发生了巨大的变化。虽然有些老师可能还会记得，该课本所附软件使用的 5.25 寸软盘，而今的大多数学生则可能根本没见过或者没有接触过这样的磁盘。学生们在课堂上和实践中所遇到的编程语言也发生了巨大的变化。在 1991 年的时候，该教材划分出了两个版本，分别使用了 C 和 Ada 两种语言来描述其中的算法。虽然 C 依然是一种很流行的语言，但是 Ada 并没有达到预期的广泛应用，已经逐渐淡出了人们的视线。从 C 语言派生出添加了面向对象特性的 C++语言。Java 则是一种更为简单的面向对象语言，由于其安全性以及可以在 Web 浏览器中运行而得以广泛流行。美国大学预修课程（College Board Advanced Placement）中使用的编程语言已经从 Pascal 转为 C++，现在又转向了 Java。

虽然大环境发生了很大的变化，但是众多学生和教师还是在继续学习和讲授编译器构造的相关课程。在编译器和程序设计语言翻译领域的研究工作也依然在持续增长，这是由于为了适应日益增加的体系结构和编程语言的种类，编译器被赋予了更加重要的使命。软件开发环境通用要依赖编译器来同各种不同的工具链构件进行交互，其中包括语法指示的编辑器、性能刻画工具（profiler）和调试器等。所有现代软件项目都要依赖于编译器对程序中的错误进行严格检查，并且做到对程序进行如实的翻译。

有些教材随着时间会做一些较小的改动，例如添加一些新的练习或者示例。与 1988 年和 1991 年的版本相比，本书则进行了非常显著的修订。虽然本教材的关注点依然是讲授编译器构造的基本知识，但是其中的算法和解决方案都采用了更为现代的方式：

- 对有些在实践中很少再使用的内容（例如，属性文法）采取了最小化处理，甚至全部去掉。
- 书中的算法采用了一种**伪码**（pseudocode）风格来展示，凡是学习过计算机学科基本算法的同学都应当会比较熟悉这种风格。伪码可以用来简洁地描述一个算法，并且支持对该算法的目的和结构进行合理的讨论。
  使用某种特定语言的实现细节则被放到了网上可以下载的补充材料中，网址是 http://www.pearsonhighered.com/fischer。
- 本书关于语法分析的理论和实践的组织可以支持各种不同的教学方法。
  有些读者可以从高层来学习这些内容以得到关于自顶向下和自底向上分析的高级视图。另外一些读者则可以选择更加详细地学习一种特定的方法。
- 编译器的前端和后端之间是通过**抽象语法树**（Abstract Syntax Tree, AST）连接起来的，抽象语法树是在分析过程中创建的主要制品。大多数编译器都会构造抽象语法树，但是很少有教材清楚地讲解它的构造和使用方法。

本教材还引入了**访问者模式**（visitor pattern），用来在语义分析和代码生成的过程中对抽象语法树进行遍历。

- 教师可以获取关于实验和上机的练习题。
  教师也可以把其中的某些部分布置为学生的练习，而其他内容则可以从课程支持网站中获取。

有些教材在修订的时候添加了更多研究生阶段的内容。虽然这些内容也会有助于开设一门高级课程，但是本教材的关注点依然是大学本科阶段要学习的编译器构造知识。研究生课程可以使用本书的第 13 章和第 14 章，而把教材的前面部分当做参考资料。

教材与参考资料

作为一本教材，本书是作者用过去 25 年时间精心建设的一门课程。本书的使用可以非常灵活，它曾经被用于在 10 周的短学期中讲授的三学分的高阶课程，也曾用于六学分的长学期中讲授的研究生课程。本教材适用于对于编程、算法和数据结构拥有基本背景的任何学生。本书的结构非常灵活，因此教师可以根据需要来构造一个短学期或长学期的教学大纲。对于那些没有详细讲解的内容，则可以为学生提供作者的解答。使用本教材，在一个学期内完成编译器中从分析到代码生成部分的编写工作是完全可行的。

本书也可以作为很好的专业参考书，因为它覆盖了对于编译器构造来说实际上最为重要的所有技术。我们的许多学生即使在他们毕业很多年之后，还会在他们工作中遇到本书中讲到的技术。

教 师 资 源

本书的网站地址是 http://www.pearsonhighered.com/fischer。网站上给符合资格的教师提供的内容包括示例的实验和项目作业布置，上机（主动学习）练习，可以被用作课程提供解答的代码库，以及部分练习的参考答案。

要想访问这些材料，有资格的教师应当访问 http://www.pearsonhighered.com，发邮件给 computing@aw.com，或者访问 http://www.pearsonhighered.com/irc/上的 Pearson 教师资源中心（Instructor Resource Center），来联系其当地的 Pearson 代表。

学 生 资 源

本书的网站 http://www.pearsonhighered.com/fischer 上包含本书中使用的示例的所有可执行的代码，其中也包括了在第 2 章中讲到的玩具语言 ac 的代码。该网站还包含了一些讲义资料，另外还有一个网页上提供了不同编译构造工具的链接。

编程实习的方案

本书提供了编译器构造中相关理论知识的全面讲解。然而，通常来说在规划编译器构造课程时很重要的一个方面是设计一个与之结合的编程实现项目。因此，本书以及在线资料都更倾向于一系列探索性的练习，最终形成一个项目，来支持对这些内容的学习。

本书补充材料中会包含实验练习、上机讲解和课程项目等内容；我们也欢迎读者发给我们其他材料或者链接，这些可以粘贴到我们的网站上。这些练习与教材中讲解的章节和内容进度保持同步。例如，第 2 章引入了玩具语言 ac 来概要讲解编译过程。网站上也包含了该语言对应的扫描器、分析器、语义分析器和代码生成器的完整、可运行的版本。这些组件中都包含用不同源编程语言书写的版本。

网站上还提供了为一个类 Java 的简单语言开发可用的编译器所需的支持材料。这样教师就可以选择把其中一些组件作为练习来布置，而把其余组件直接提供给学生填补其中的缺口。有些教师也可以选择提供整个编译器，而要求学生对其进行扩展。对现有组件进行润色和细化也可以作为课程项目的基础。

伪码和指引表

与老版本的教材相比，本书的一个重要变化是：书中的算法不再使用任何具体的程序设计语言（例如 C 或 Ada）来展现。作为替代，算法使用了一种伪码（pseudocode）形式，这种形式对于所有学过最基本算法 [CLRS01] 的读者来说应当都会比较熟悉。伪码通过略去不必要的细节而简化对算法的阐释。然而，伪码中暗含了在真实程序设计语言中使用的结构，所以实现起来会相当容易。在本书最后会为书中出现的所有伪码函数提供一个指引表（guide）。

本教材中大量使用了缩写（包括首字母缩略词）来简化对问题的阐述，并帮助读者了解在编译器构造中经常使用的术语。在每章中首次出现每个缩写的时候，都会自动给出完整的定义。例如，AST 已经在本前言中用到过（作为抽象语法树的缩写），但是上下文无关文法（context-free grammar, CFG）还没有用到。作为进一步的帮助，在本书最后把所有缩写的索引也做了一个指引表。完整的索引列表中也包含了这些缩写，并会说明它们在书中哪些地方用到。**指引表**中的术语会用**粗体**形式来表示。在完整的索引表中也会包括所有这些术语的每一次引用。

如何使用本书

关于编译器构造的介绍性课程可以先讲述第 1～3 章。对于语法分析技术，可以选择讲解自顶向下（第 5 章）或者自底向上（第 6 章）的内容，但是有些教师会选择两章内容都讲解。第 4 章中的内容可以作为支持所要讲解的分析技术的基础。第 7 章介绍抽象语法树，

并引入用来对它进行遍历的**访问者模式**（visitor pattern）。有些教师可能会把管理抽象语法树的实用程序当做编程练习，而其他老师则可能会选择使用教材网站提供的实用程序。根据教师的选择，可以讲解第 8～9 章中关于语义分析的不同内容。如果是一个季度的课程，那么可能到这里就结束了；这样就需要用另外一个季度来讲解代码生成以后的内容（随后进行介绍）。

第 10 章提供了关于 **Java 虚拟机**（JVM）的简介，如果学生要在编程作业中生成 JVM 代码的话，那么就需要学习本章。第 11 章会讲解如何为这样的虚拟机进行代码生成。如果教师希望学生生成机器码的话，那么可以跳过第 10 章和第 11 章，而是讲解第 12 章和第 13 章。初级编译课程中可以包含第 14 章前面部分关于自动程序优化的部分内容。

更进一步的学习则可以包括在第 4～6 章中涉及的分析技术的更多细节。而第 8 章和第 9 章中的语义分析和类型检查也可以更加深入地进行讲解。还可以引入第 10 章和第 14 章中的更加高级的概念，例如**静态单赋值**（static single assignment，SSA）形式。教师还可以从第 14 章中找到关于程序分析和转换的高级内容（包括数据流框架在内）。第 13 章和第 14 章也可以作为研究生阶段的编译课程的基础，而把之前的章节作为有用的参考资料。

章 节 简 介

第 1 章　概述

本教材首先会简要介绍编译的过程，其中会强调使用一系列构件来构造编译器的概念。本章还介绍了编译器的历史，以及如何使用工具来生成编译器的这些构件。

第 2 章　一个简单的编译器

本章介绍一个简单的语言 ac，以及用来把 ac 转换为另外一种语言 dc 的编译器构件。这些构件都是用伪码来介绍的，完整的代码可以从“《编译器构造》补充材料”中找到。

第 3 章　扫描——理论和实践

本章讲解用来构建词法分析构件的基本概念和技术，内容包括手工编码的扫描器，以及使用扫描器生成工具来实现表格驱动的词法分析器。

第 4 章　文法和分析

本章讲解形式语言的概念和基础，其中包括上下文无关文法、文法记号、推导和分析树。同时会引入在第 5 章和第 6 章中要用到的文法分析算法。

第 5 章　自顶向下分析

自顶向下分析是用来构造相对简单的分析器的常用技术。本章会讲解如何使用直接编码，或者通过给通用的自顶向下分析引擎构造一个表格来编写这样的分析器。本章还会讨论语法错误的诊断、恢复和改正。

第 6 章　自底向上分析

现代编程语言所使用的绝大多数编译器都会使用本章中讲解的某种自底向上的分析技术。读者可以很容易获得从一个上下文无关文法来自动生成这类分析器的工具。本章会讲

解这些工具构造的理论基础，其中包括用在某些文法会产生阻碍分析器构造的**冲突**（conflict）时，为消除冲突所采用的一系列越来越复杂的解决方案。本章还会详细讨论文法和语言的**二义性**（ambiguity），并讲解用来理解和消除二义性文法的启发式方法。

第 7 章　语法制导翻译

按照编译器的组成构件来划分，本章是本教材的一个分水岭。前面的章节中涉及的都是程序的词法和语法分析。这些章节的目标是构造一个抽象语法树。在本章中会介绍 AST，以及用来对 AST 进行构造、管理和遍历的一个接口。本章对于后续章节是至关重要的，因为它们会依赖于读者对于 AST 的理解，以及用来辅助对 AST 进行遍历和处理的**访问者模式**（visitor pattern）的理解。在“《编译器构造》补充材料”中包含关于访问者模式的一个教程，其中包括了从实践中抽取的一些实例。

第 8 章　符号表和声明处理

本章会着重介绍符号表的使用，它是在整个编译过程中都会用到的一个抽象构件。本章会为符号表定义一个准确的接口，并且讲解各种与实现相关的问题和想法，其中还会讨论如何实现嵌套的作用范围。

本章会讲解在处理符号声明的过程中必不可少的语义分析，包括类型、变量、数组、结构和枚举。另外，本章还会介绍类型检查，其中包括面向对象类、子类和超类。

第 9 章　语义分析

对于在分析过程中不能很容易检查的语言规范需要进行额外的语义分析。在这里会检查不同的控制结构，包括条件分支和循环。本章还讨论了异常（exception），以及在编译时它们要求进行的语义分析。

第 10 章　中间表示形式

本章会涉及编译器广泛使用的两种中间表示形式。第一种是 JVM 指令集和字节码形式，这已经成了用来表示经过编译的 Java 程序的标准格式。对于感兴趣在编译器项目中以 JVM 作为目标码的读者，第 10 章和第 11 章提供了必要的背景知识和技巧。另外一种表示是 SSA 形式，常用在很多优化编译器中。本章会定义 SSA 形式，但是它的构造会放到第 14 章中，等到我们把一些必要的定义和算法都讲解完之后介绍。

第 11 章　面向虚拟机的代码生成

本章讲解如何为虚拟机（virtual machine，VM）生成代码。考虑以虚拟机作为目标的优点是可以把许多运行时支持的细节都交给 VM 来承担。例如，大多数 VM 会提供不限数量的寄存器，从而就可以把**寄存器分配**（register allocation）推迟到掌握了代码生成的基础之后再进行。VM 的指令集通常比机器代码要高级。例如，函数调用通常可以用单个 VM 指令来支持，而相同的调用在机器代码中则需要很多条指令。

如果读者热切期望了解如何生成机器代码的话，那么很可能想跳过第 11 章，但我们建议在为机器代码级别尝试生成代码之前先要学习本章。本章中的思想可以很容易就应用到第 12 章和第 13 章，但是从 VM 的角度来看它们更加容易理解。

第 12 章　运行时支持

VM 内嵌的功能中大多都是它提供的运行时支持（例如，对存储管理的支持）。本章讨论为现代程序设计语言提供运行时支持所需的各种不同概念和实现策略。学习这些内容会提供对于 VM 构造的理解。对于那些希望为目标体系结构（第 13 章）生成机器代码的读者，运行时支持是必须要提供的，因此学习这些内容对于创建一个可用的编译器是至关重要的。

本章会讨论静态分配、栈式分配和堆式分配的存储形式。另外还会涉及对**非本地存储**（nonlocal storage）的引用方法，以及为了支持这种引用所需的实现结构，例如，活动记录（frame）和 display 表。

第 13 章　目标代码生成

本章和第 11 章很相似，不同之处是代码生成的目标是与 VM 相比更为低级的指令集。本章对于在代码生成过程中会遇到的问题进行了详细讨论，包括寄存器分配、临时变量管理、代码调度、指令选择和一些基本的窥孔优化技术。

第 14 章　程序优化

绝大多数编译器包括一些对它们生成的代码进行优化的能力。本章介绍在编译器中常用的程序优化的实用技术。本章会讲解高级的控制流分析结构和算法，另外还简要介绍了数据流分析（data flow analysis），并讨论了相对容易实现的一些基本优化技术。本章讲解这些优化技术的理论基础，并介绍了 SSA 形式的构造和使用。

致　　谢

作者感谢如下人士在本书撰写过程中所提供的支持。我们感谢 Pearson 出版社的 Matt Goldstein 在本书修订过程中的耐心和支持。由于在本书撰写过程中造成的拖延，我们对 Matt 的前任深表歉意。我们也很感激 Jeff Holcomb 为我们提供了关于 Pearson 出版过程的技术指导。本教材在我们的审校编辑手中得到了很大的改进。Stephanie Moscola 及时而专业地审校并修改了本书中的每一章。她进行了异常细致的工作，书中所有剩余的错误都是作者的错误。我们非常感激她锐利的眼睛和洞察力，以及提出的建议。我们感谢 Will Benton 对第 12 章和第 13 章的编辑，以及在 12.5 节中的部分撰写工作。我们感谢 Aimee Beal 被留在了 Pearson 来负责对本书的风格和一致性进行编辑修改。

我们非常感谢如下同事们付出了他们宝贵的时间审阅本书，并提出了非常宝贵的反馈：Ras Bodik（加州伯克利大学）、Scott Cannon（犹他州立大学）、Stephen Edwards（哥伦比亚大学）、Stephen Freund（威廉姆斯学院）、Jerzy Jaromczyk（肯塔基大学）、Hikyoo Koh（拉马尔大学）、Sam Midkiff（普渡大学）、Tim O'Neil（阿肯大学）、Kurt Stirewalt（密歇根州立大学）、Michelle Strout（科罗拉多州立大学）、Douglas Thain（圣母大学）、V. N. Venkatakrishnan（伊利诺伊大学芝加哥分校）、Elizabeth White（乔治梅森大学）、Sherry Yang（俄勒冈技术学院）和 Qing Yi（得克萨斯大学圣安东尼奥分校）。

Charles Fischer：我对编译器着迷是从 1965 年在 Robert Eddy 先生的计算机实验课上开始的。虽然我们当时的计算机总共只有 20KB 的主存，而我们的编译器要使用打孔卡来作为它的中间媒介，但是编译的种子就是在那个时候种下的。

我的教育经历是从康奈尔大学开始的，在那里我学到了计算技术的深奥和严格。David Gries 富有创意的编译器教材教给我很多知识，并引领我走上了我的职业道路。

威斯康辛大学的教员们，特别是 Larry Landweber 和 Tad Pinkerton，允许我放开手脚建设编译课程和研究团队。教学计算中心的 Tad、Larry Travis 和 Manley Draper 给了我很多的时间和资源来学习编译的实践。威斯康辛大学的 UW-Pascal 编译器项目把我介绍给了许多优秀的学生，其中包括我的共同作者 Richard LeBlanc。我们在实践中进行学习，而这也成为了我的教学理念。

在过去的许多年间，我的同事们，特别是 Tom Reps、Susan Horwitz 和 Jim Larus，都很无私地让我分享了他们的智慧和经验，使我从中获益良多。在计算机体系结构方面，Jim Goodman、Guri Sohi、Mark Hill 和 David Wood 向我传授了现代体系结构的点点滴滴。作为编译器的开发人员必须认真仔细地理解处理器，才能释放它的全部能力。

我最应该感激的是我的学生们，他们给我的课程带来了巨大的能力和热情。他们热切地接受了我所展示给他们的挑战。从扫描器到代码生成器开发一个完整的编译器，这是在一个学期内看起来肯定无法完成的任务，但是他们做到了，而且做得很好。此中的许多经验都渗入到了这本教材中。我确信教会新的一代人如何构造编译器是非常有意义的一项工作。

Ron K. Cytron：我最初对程序设计语言和其编译器的兴趣以及之后的研究主要来自于在我的职业生涯中产生过重要作用的良师们。我们深切怀念的 Ken Kennedy 在 Rice 大学教过我的编译课。我现在所教的课程都是按照他的方式来设计的，特别是强调编程作业在帮助学生理解课程内容中的作用。Ken Kennedy 是一位杰出的教育家，我只奢望能像他一样与学生进行交流。他还在位于纽约州 Yorktown Heights 的 IBM T.J. Watson 研究实验室雇我做了一个暑期的实习，主要工作是应用自动并行化的软件。在那个夏天，我的研究很自然地把我引到了 Dave Kuck 和他的学生在伊利诺伊大学的研究工作。

我依然认为 Dave 能让我成为他的研究生对我来说是非常幸运的。Dave Kuck 在并行计算机体系结构，以及如何利用编译器来使如此高深的系统更加容易编程方面是一名先驱。我努力以他为榜样，学习他的努力工作、正直和坚持不懈，并把这些经验都传授给我的学生。我同样也经历了在一个团队中进行研究而带来的活跃和乐趣，并且试图在我的学生中间营造类似的氛围。

我作为本科生和研究生的经验把我引向了 IBM Research 的 Fran Allen，我一直非常感激她允许我加入了她刚刚成立的 PTRAN 研究组。Fran 在数据流分析、程序优化和自动并行化方面激励了几代人的研究工作。她对于许多重要问题及可能的解决方案拥有令人惊讶的直觉。在与同事们交谈中，我们一些最好的想法都来自于 Fran，以及她为我们提供的建议、指导或批评。

我职业生涯中最好的几年是在向 Fran 和 PTRAN 同事们学习和与他们共事中度过的，他们是 Michael Burke、Philippe Charles、Jong-Deok Choi、Jeanne Ferrante、Vivek Sarkar 和

David Shields。在 IBM，我还非常高兴能从 Barry Rosen、Mark Wegman 和 Kenny Zadeck 那里学习并与他们共事。虽然我的朋友和同事们的印记在本书中随处可见，但所有的错误都是属于我的。

如果读者注意到在本书中频繁出现 431 这个数字的话，它是为了纪念所有在华盛顿大学跟我学习过编译的同学。从我的学生那里学到的和我教给他们的一样多，而我对本书所做的贡献主要都来自于我在教室和实验室里得到的经验。

最后，我要感谢我的妻子和孩子给我时间来准备这本书，他们在整个过程中展现了很好的耐心和理解。最后我要感谢 Carole 阿姨总是在询问这本书写得怎么样了。

Richard LeBlanc：在我获得物理学学士学位，并发现自己对计算机比物理问题更加感兴趣之后，我在 1972 年来到了麦迪逊，加入了威斯康辛大学成了一名计算机科学专业的博士生。两年后，一位刚刚从康奈尔拿到博士学位的年轻助理教授，Charles Fischer，加入了计算机科学系。他讲授的第一门课程是研究生的编译课（CS701）。我选了那门课，并且至今还记得当时令人难以忘怀的学习经历，而正是由于他是第一次讲授该课程才让人觉得更加难能可贵。我们显然彼此都很投缘，由此引向了一系列相当长时间的合作。

在 Larry Travis 的支持下，我从 1974 年夏天开始在教学计算中心工作，并因此在 UW-Pascal 项目在一年后开始的时候，我早已成了该组织的一员。这个项目不仅给我提供了机会来应用我在刚学过的两门课中所学到的知识，还交给我关于好的设计和设计评价的影响。在与我的两位研究生同学 Steve Zeigler 和 Marty Honda 的共事中我也受益良多，从他们身上我了解到作为一个有效的软件开发团队的一部分是多么有趣。我们都发现了使用 Pascal 工作的价值：Pascal 是一个设计良好的语言，要求在编程中训练有素，并且我们使用的是自己开发的工具，因为我们在项目早期从 Pascal 的 P-Compiler 自举，得到了可以为 Univac 1108 生成本地代码的我们自己的编译器。

在完成了研究生学业之后，我接受了佐治亚理工学院的教职，因为我喜欢那里温暖的气候，以及由 Phil Enslow 领导的分布式计算研究项目，Phil 在我职业生涯的早期提供了非常宝贵的指导。我立即得到了讲授一门编译器课程的机会，并且试图模仿在威斯康辛学习的 CS 701 课程，因为我强烈相信 Charles 使用过的以项目为基础的教学方式。我迅速地意识到要学生在 10 周的小学期内开发一个完整的编译器是太大挑战。因此我开始使用另外的方法：给他们一个非常小的语言的可用编译器，然后课程项目是围绕对这个编译器的所有模块进行扩展来编译更加复杂的语言。在我的 10 周课程中使用的基础编译器就成了 Fischer-LeBlanc 教材中所带的辅助支持材料之一。

我的职业规划使得我参与软件工程和教学工作相比编译器研究要更多。当我回顾在威斯康辛大学的早期编译器经验，我清晰地看到我对这两个领域都很感兴趣。Charles 和我最早决定开始编写第一版的《编译器构造》是基于我们相信可以帮助其他教师，通过基于项目的编译器课程，能为他们的学生提供更好的教学经历。在我们的编辑 Alan Apt，以及许多审阅者的宝贵帮助下，我相信我们已经取得了成功。许多同事都向我提到他们对于我们最初版本的教材以及《编译器构造：C 语言描述》的热情。他们的支持是一种巨大的奖励，鼓励我们最终完成了这本新版的教材。特别要感谢的是 Joe Bergin，他除了口头支持之外，还把我们早期的软件工具翻译成了新的编程语言，并且允许我们把他的版本提供给其他教

师使用。

在佐治亚理工学院的岁月给我提供了很好的机会来发展我对计算机教育的兴趣。在我职业生涯的早期，我很有幸成为先是由 Ray Miller，然后是 Pete Jensen 所领导的团队的一员。从 1990 年开始，我非常高兴能与 Peter Freeman 一起工作，创立并发展了计算学院。除了在佐治亚理工学院的时候 Peter 对我的多方指导之外，他还鼓励我参与了 ACM Education Board 与教育相关的工作，这些工作在过去 12 年中极大地丰富了我的职业生涯。

最后，我要感谢我的家人，包括刚出生的孙女，在写作本书的漫长过程中一直陪伴我。

献　　词

CNF：献给 Lisa
怀念属于伟大一代的 Stanley J. Winiasz

RKC：献给 Betsy、Jessica、Malanie 和 Jacob
怀念 Ken Kennedy

RJL：献给 Lanie、Aidan、Maria 和 Evolette

目　录

第 1 章　概述 1
1.1　编译的历史 1
1.2　编译器可以做什么 3
1.2.1　编译器生成的机器代码 3
1.2.2　目标代码格式 5
1.3　解释器 6
1.4　语法和语义 7
1.4.1　静态语义 8
1.4.2　运行时语义 8
1.5　编译器的组织结构 10
1.5.1　扫描器 11
1.5.2　分析器 12
1.5.3　类型检查器（语义分析） 12
1.5.4　翻译器（程序综合） 12
1.5.5　符号表 13
1.5.6　优化器 13
1.5.7　代码生成器 13
1.5.8　编译器开发工具 14
1.6　程序设计语言和编译器设计 14
1.7　计算机体系结构和编译器设计 15
1.8　编译器设计的考虑事项 16
1.8.1　调试（开发）编译器 16
1.8.2　优化编译器 17
1.8.3　可重定向编译器 17
1.9　集成开发环境 18
练习 18
第 2 章　一个简单的编译器 21
2.1　ac 语言的非形式化定义 21
2.2　ac 语言的形式化定义 22
2.2.1　语法规范 22
2.2.2　词法单元规范 24
2.3　一个简单编译器中的阶段 25
2.4　扫描 25

2.5 分析 27
2.5.1 分析过程的预测 27
2.5.2 产生式的实现 28
2.6 抽象语法树 29
2.7 语义分析 31
2.7.1 符号表 31
2.7.2 类型检查 32
2.8 代码生成 34
练习 36
第 3 章 扫描——理论和实践 38
3.1 扫描器概述 38
3.2 正则表达式 40
3.3 示例 42
3.4 有限自动机和扫描器 43
3.4.1 确定性的有限自动机 44
3.5 扫描器生成工具 Lex 46
3.5.1 定义 Lex 中的词法单元 47
3.5.2 字符类 48
3.5.3 使用正则表达式来定义词法单元 49
3.5.4 使用 Lex 进行字符处理 51
3.6 其他扫描器生成工具 53
3.7 构造扫描器的实际注意事项 53
3.7.1 处理标识符和字面常量 54
3.7.2 使用编译命令和列出源码行 57
3.7.3 扫描器的终止 58
3.7.4 向前看多个字符 58
3.7.5 性能上的考虑 60
3.7.6 词法错误恢复 61
3.8 正则表达式和有限自动机 63
3.8.1 把正则表达式转换为 NFA 64
3.8.2 创建 DFA 64
3.8.3 有限状态机的化简 66
3.8.4 把有限自动机转换为正则表达式 68
3.9 本章小结 71
练习 72
第 4 章 文法和分析 77
4.1 上下文无关文法 77
4.1.1 最左推导 79