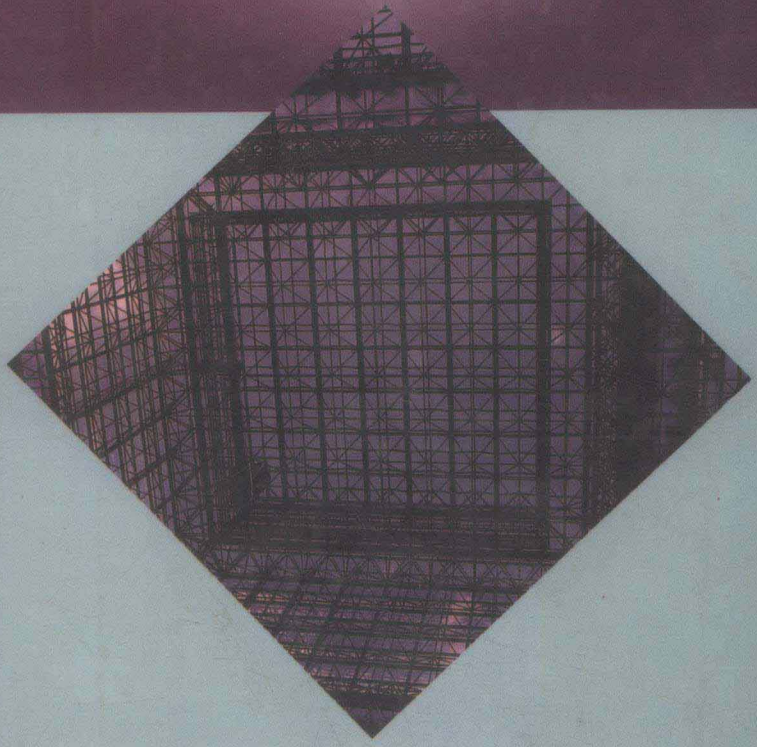
经 典 原 版 书 库

编译原理与实践

（英文版）

COMPILER CONSTRUCTION

Principles and Practice



Kenneth C. Louden

（美）Kenneth C. Louden 著

机械工业出版社
China Machine Press

中信出版社
CITIC PUBLISHING HOUSE

THOMSON

经 典 原 版 书 库

编译原理与实践

（英文版）

Compiler Construction Principles and Practice

（美） Kenneth C. Louden 著

机械工业出版社 China Machine Press

本书版权登记号：图字：01-2001-5320

图书在版编目（CIP）数据

编译原理与实践/（美）楼登[redacted]著.[redacted]北京：机械工业出版社，2002.8
（经典原版书库）
书名原文：Compiler Construction Principles and Practice
ISBN 7-111-10842-6

Ⅰ.编… Ⅱ.楼… Ⅲ.编译程序－程序设计－英文 Ⅳ.TP314

中国版本图书馆CIP数据核字（2002）第063184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：华章
北京忠信诚印刷厂印刷·新华书店北京发行所发行
2002年8月第1版第1次印刷
787mm×1092mm1/16·37印张
印数：0 001-3 000册
定价：58.00元

Preface

This book is an introduction to the field of compiler construction. It combines a detailed study of the theory underlying the modern approach to compiler design, together with many practical examples, and a complete description, with source code, of a compiler for a small language. It is specifically designed for use in an introductory course on compiler design or compiler construction at the advanced undergraduate level. However, it will also be of use to professionals joining or beginning a compiler writing project, as it aims to give the reader all the necessary tools and practical experience to design and program an actual compiler.

A great many texts already exist for this field. Why another one? Because virtually all current texts confine themselves to the study of only one of the two important aspects of compiler construction. The first variety of text confines itself to a study of the theory and principles of compiler design, with only brief examples of the application of the theory. The second variety of text concentrates on the practical goal of producing an actual compiler, either for a real programming language or a pared-down version of one, with only small forays into the theory underlying the code to explain its origin and behavior. I have found both approaches lacking. To really understand the practical aspects of compiler design, one needs to have a good understanding of the theory, and to really appreciate the theory, one needs to see it in action in a real or near-real practical setting.

This text undertakes to provide the proper balance between theory and practice, and to provide enough actual implementation detail to give a real flavor for the techniques without overwhelming the reader. In this text, I provide a complete compiler for a small language written in C and developed using the different techniques studied in each chapter. In addition, detailed descriptions of coding techniques for additional language examples are given as the associated topics are studied. Finally, each chapter concludes with an extensive set of exercises, which are divided into two sections. The first contains those of the more pencil-and-paper variety involving little programming. The second contains those involving a significant amount of programming.

In writing such a text one must also take into account the different places that a compiler course occupies in different computer science curricula. In some programs, a course on automata theory is a prerequisite; in others, a course on programming languages is a prerequisite; while in yet others no prerequisites (other than data structures) are assumed. This text makes no assumptions about prerequisites beyond the usual data

structures course and a familiarity with the C language, yet is arranged so that a prerequisite such as an automata theory course can be taken into account. Thus, it should be usable in a wide variety of programs.

A final problem in writing a compiler text is that instructors use many different classroom approaches to the practical application of theory. Some prefer to study the techniques using only a series of separate small examples, each targeting a specific concept. Some give an extensive compiler project, but make it more manageable with the use of Lex and Yacc as tools. Others ask their students to write all the code by hand (using, say, recursive descent for parsing) but may lighten the task by giving students the basic data structures and some sample code. This book should lend itself to all of these scenarios.

Overview and Organization

In most cases each chapter is largely independent of the others, without artificially restricting the material in each. Cross-references in the text allow the reader or instructor to fill in any gaps that might arise even if a particular chapter or section is skipped.

Chapter 1 is a survey of the basic structure of a compiler and the techniques studied in later chapters. It also includes a section on porting and bootstrapping.

Chapter 2 studies the theory of finite automata and regular expressions, and then applies this theory to the construction of a scanner both by hand coding and using the scanner generation tool Lex.

Chapter 3 studies the theory of context-free grammars as it pertains to parsing, with particular emphasis on resolving ambiguity. It gives a detailed description of three common notations for such grammars, BNF, EBNF, and syntax diagrams. It also discusses the Chomsky hierarchy and the limits of the power of context-free grammars, and mentions some of the important computation-theoretic results concerning such grammars. A grammar for the sample language of the text is also provided.

Chapter 4 studies top-down parsing algorithms, in particular the methods of recursive-descent and LL(1) parsing. A recursive-descent parser for the sample language is also presented.

Chapter 5 continues the study of parsing algorithms, studying bottom-up parsing in detail, culminating in LALR(1) parsing tables and the use of the Yacc parser generator tool. A Yacc specification for the sample language is provided.

Chapter 6 is a comprehensive account of static semantic analysis, focusing on attribute grammars and syntax tree traversals. It gives extensive coverage to the construction of symbol tables and static type checking, the two primary examples of semantic analysis. A hash table implementation for a symbol table is also given and is used to implement a semantic analyzer for the sample language.

Chapter 7 discusses the common forms of runtime environments, from the fully static environment of Fortran through the many varieties of stack-based environments to the fully dynamic environments of Lisp-like languages. It also provides an implementation for a heap of dynamically allocated storage.

Chapter 8 discusses code generation both for intermediate code such as three-address code and P-code and for executable object code for a simple von Neumann

architecture, for which a simulator is given. A complete code generator for the sample language is given. The chapter concludes with an introduction to code optimization techniques.

Three appendices augment the text. The first contains a detailed description of a language suitable for a class project, together with a list of partial projects that can be used as assignments. The remaining appendices give line-numbered listings of the source code for the sample compiler and the machine simulator, respectively.

Use as a Text

This text can be used in a one-semester or two-semester introductory compiler course, either with or without the use of Lex and Yacc compiler construction tools. If an automata theory course is a prerequisite, then Sections 2.2., 2.3, and 2.4 in Chapter 2 and Sections 3.2 and 3.6 in Chapter 3 can be skipped or quickly reviewed. In a one-semester course this still makes for an extremely fast-paced course, if scanning, parsing, semantic analysis, and code generation are all to be covered.

One reasonable alternative is, after an overview of scanning, to simply provide a scanner and move quickly to parsing. (Even with standard techniques and the use of C, input routines can be subtly different for different operating systems and platforms.) Another alternative is to use Lex and Yacc to automate the construction of a scanner and a parser (I do find, however, that in doing this there is a risk that, in a first course, students may fail to understand the actual algorithms being used). If an instructor wishes to use only Lex and Yacc, then further material may be skipped: all sections of Chapter 4 except 4.4, and Section 2.5 of Chapter 2.

If an instructor wishes to concentrate on hand coding, then the sections on Lex and Yacc may be skipped (2.6, 5.5, 5.6, and 5.7). Indeed, it would be possible to skip all of Chapter 5 if bottom-up parsing is ignored.

Similar shortcuts may be taken with the later chapters, if necessary, in either a tools-based course or a hand-coding style course. For instance, not all the different styles of attribute analysis need to be studied (Section 6.2). Also, it is not essential to study in detail all the different runtime environments cataloged in Chapter 7. If the students are to take a further course that will cover code generation in detail, then Chapter 8 may be skipped.

In a two-quarter or two-semester course it should be possible to cover the entire book.

Internet Availability of Resources

All the code in Appendices B and C is available on the Web at locations pointed to from my home page (http://www.mathcs.sjsu.edu/faculty/louden/). Additional resources, such as errata lists and solutions to some of the exercises, may also be available from me. Please check my home page or contact me by e-mail at louden@cs.sjsu.edu.

Acknowledgments

My interest in compilers began in 1984 with a summer course taught by Alan Demers. His insight and approach to the field have significantly influenced my own views.

Indeed, the basic organization of the sample compiler in this text was suggested by that course, and the machine simulator of Appendix C is a descendant of the one he provided.

More directly, I would like to thank my colleagues Bill Giles and Sam Khuri at San Jose State for encouraging me in this project, reading and commenting on most of the text, and for using preliminary drafts in their classes. I would also like to thank the students at San Jose State University in both my own and other classes who provided useful input. Further, I would like to thank Mary T. Stone of PWS for gathering a great deal of information on compiler tools and for coordinating the very useful review process.

The following reviewers contributed many excellent suggestions, for which I am grateful:

Jeff Jenness
Arkansas State University

Joe Lambert
Penn State University

Joan Lukas
University of Masschusetts, Boston

Jerry Potter
Kent State University

Samuel A. Rebelsky
Dartmouth College

Of course I alone am responsible for any shortcomings of the text. I have tried to make this book as error-free as possible. Undoubtedly errors remain, and I would be happy to hear from any readers willing to point them out to me.

Finally, I would like to thank my wife Margreth for her understanding, patience, and support, and our son Andrew for encouraging me to finish this book.

K.C.L.

国外经典教材

《程序设计实践》	Brian W. Kernighan, Rob Pike 著/裘宗燕 译/20元
《Linux操作系统内核实习》	Gary J. Nutt 著/陆丽娜 等译/29元
《程序设计语言概念和结构》(原书第2版)	Ravi Sethi 著/裘宗燕 等译/45元
《编译原理》	Alfred Aho 著/李建中 等译
《C++程序设计语言》(特别版)	Bjarne Stroustrup 著/裘宗燕 译/85元
《并行计算机体系结构》(原书第2版)	David E. Culler, Jaswinder Pal Singh 著/程旭 等译
《人工智能》	Nils J. Nilsson 著/郑扣根 等译/39元
《神经网络设计》	Hagan 著/戴葵 等译
《计算理论导引》	Sipser 著/张立昂 等译/30元
《数据库系统概念》(原书第4版)	Silberschatz 著/杨冬青 等译
《数据结构、算法与应用——C++语言描述》(中文版)	Sartaj Sahni 著/戴葵 等译/49元
《离散数学及其应用》(原书第4版)	Kenneth H. Rosen 著/袁崇义 等译/75元
《计算机图形学的算法基础》(第2版)	David F. Rogers 著/石教英 等译/55元
《可扩展并行计算:技术、结构与编程》	Kai Hwang, Zhiwei Xu 著/陆鑫达 等译/49元
《软件工程:实践者的研究方法》(原书第5版)	Roger S. Pressman 著/梅宏 译
《数据通信与网络》(第2版)	Behrouz Forouzan 著/吴时霖 等译/68元
《数据库原理、编程与性能》(原书第2版)	Patrick O' Neil, Elizabeth O' Neil 著/周傲英 等译/55元
《现代操作系统》(第2版)	Andrew S. Tanenbaum 著/陈向群 等译
《C程序设计语言》(第2版)	Kernighan, Ritchie 著/徐宝文 等译/28元
《组合数学》	Richard A. Brualdi 著/冯舜玺 等译/38元
《结构化计算机组成》	Andrew S. Tanenbaum 著/刘卫东 等译/46元

计算机科学丛书

《设计模式 可复用面向对象软件的基础》	Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides 著/吕建 等译/35元
《UNIX环境高级编程》	W. Richard Stevens 著/尤晋元 等译/55元
《TCP/IP详解 卷1:协议》	W. Richard Stevens 著/谢希仁 等译/45元
《TCP/IP详解 卷2:实现》	Gary R. Wright, W. Richard Stevens 著/谢希仁 等译/78元
《TCP/IP详解 卷3:TCP事务协议、HTTP、NNTP和UNIX域协议》	W. Richard Stevens 著/谢希仁 等译/35元
《数据库系统导论》	C. J. Date 著/王珊 等译/66元

《计算机图形学原理及实践——C语言描述》（原书第2版）	James D. Foley, Andries Van Dam, Steven K. Feiner, John F. Hughes 著/唐泽圣 等译
《C语言解析教程》（原书第4版）	Ira Pohl 著/麻志毅 等译/48元
《C++语言的设计和演化》	Bjarne Stroustrup 著/裘宗燕 译/48元
《人本界面——设计交互式系统的最新指示》	Jef Raskin 著/史元春 译
《分布式系统设计》	Jie Wu 著/高传善 等译/30元
《数据通信与网络教程》	William A. Shay 著/高传善 等译/40元
《专家系统原理与编程》	Giarratano, Riley 著/印鉴 等译/49元
《编译原理及实践》	Kenneth C. Louden 著/冯博琴 等译/39元
《最新网络技术基础》	Palmer 著/严伟 等译/20元
《数字逻辑：应用与设计》	John M. Yarbrough 著/朱海滨 等译/49元
《计算机文化》	June Jamrich Parsons, Dan Oja 著/朱海滨 等译/50元
《信息系统原理》	Ralph M. Stair, George W. Reynolds 著/张靖 等译/42元
《计算机信息处理》	Mandell 著/尤晓东 等译/38元
《数据库管理系统基础》	Philip J. Pratt, Joseph J. Adamski 著/戴葵 等译/20元
《数据仓库》	W. H. Inmon 著/王志海 等译/25元
《数据广播》	Lars Tvede 著/徐良贤 等译/28元
《数据库设计》	Ryan K. Stephens Ronald R. Plew 著/何玉洁 武欣 等译/35元
《软件工程——实践者的研究方法》（原书第4版）	Roger S. Pressman 著/梅宏 等译/48元
《软件工程——Java语言实现》	Stephen R. Schach 著/袁兆山 等译/38元
《数据库系统概念》（中文版）	Abraham Silberschatz, Henry F. Korth, S. Sudarshan 著/杨冬青 等译/49元
《数据通信与网络》	Behrouz Forouzan 著/吴时霖 等译/48元
《计算机网络》（第2版，中文版）	Larry L. Peterson, Bruce S.Davie 著/叶新铭 等译/49元
《高性能通信网络》（原书第2版）	Jean Walrand, Pravin Varaiya 著/史美林 等译/55元
《嵌入式计算系统设计原理》	Wayne Wolf 著/孙玉芳 等译/65元
《数据挖掘：概念与技术》	Jiawei Han, Micheline Kamber 著/范明 孟小峰 等译/39元
《编码的奥秘》	Charles Petzold 著/陆丽娜 等译/24元
《软件需求》	Karl E. Wiegers 著/陆丽娜 等译/19元
《面向对象程序设计——图形应用实例》	Micheal J. Laszlo 著/何玉洁 等译/35元
《现代数据库管理》（原书第6版）	Jeffrey A. Hoffer 著/施伯乐 杨卫东等译
《数据库系统基本原理》	Jeffrey D. Ullman 著/周傲英 等译
《数据库与事务处理》	Philip M. Lewis 著/施伯乐 杨卫东 等译
《数据库系统教程》	Hector Garcia-Molina 著/岳丽华 等译
《UNIX操作系统教程》	Sarwar 著/徐良贤 等译
《计算机网络与因特网》（第2版）	Douglas E. Comer 著/徐良贤 等译/40元
《数据库系统实现》	Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom 著/杨冬青 等译/45元

《ISDN、B-ISDN与帧中继和ATM》(原书第4版)	William Stallings 著/程时端 等译/48元
《UNIX操作系统设计》	Maurice J. Bach 著/陈葆珏 等译/35元
《Java 程序设计教程》(原书第3版)(上册)	Harvey Deitel 著/袁兆山 等译/55元
《Java 程序设计教程》(原书第3版)(下册)	Harvey Deitel 著/袁兆山 等译/69元
《C++精髓：软件工程方法》	Victor Shtern 著/李师贤 等译/85元
《Java编程思想》(第2版)	Bruce Eckel 著/侯捷 译/99元
《C++编程思想》(第2版)	Bruce Eckel 著/刘宗田 等译/59元
《并行程序设计》	Barry Wilkinson 著/陆鑫达 等译/43元
《神经网络原理》	Simon Haykin 著/史忠植 等译
《分布式操作系统：原理与实践》	Doreen L. Galli 著/徐良贤 等译
《现代操作系统》	Andrew S. Tanenbaum 著/陈向群 等译/40元
《Java编程思想》	Bruce Eckel 著/尤晓东 等译/60元
《Unix编程环境》	Brian W. Kernighan, Rob Pike 著/陈向群 等译/24元
《Internet技术基础》	Douglas E. Comer 著/袁兆山 等译/18元
《C++编程思想》	Bruce Eckel 著/刘宗田 等译/39元
《电子商务》	Gary P. Schneider, James T. Perry 著/成栋 等译/28元
《计算机网络实用教程实验手册》	Tamara Dean 著/陶华敏 等译/15元
《计算机网络实用教程》	Todd Meadors 著/陶华敏 等译/65元
《C程序设计教程》	Deitel 著/张祖荫 等译/33元
《C++程序设计教程》	Deitel 著/马鸣远 等译/22元

经典原版书库

《并行计算机体系结构》(英文版)	David E. Culler, Jaswinder Pal Singh 著/88元
《人工智能》(英文版)	Nils J. Nilsson 著/45元
《高性能通信网络》(英文版)	Jean Walrand, Pravin Varaiya 著/64元
《计算机网络》(英文版)	Larry L. Peterson, Bruce S. Davie 著/65元
《计算机体系结构：量化研究方法》第2版(英文版)	David A. Patterson, John L. Hennessy 著/88元
《计算机组织与设计：硬件/软件接口》第2版(英文版)	John L. Hennessy, David A. Patterson 著/80元
《计算机图形学的算法基础》(英文版·第2版)	David F. Rogers 著/45元
《计算机组成》(英文版·第5版)	Carl Hamacher 著/48元
《可扩展并行计算:技术、结构与编程》(英文版)	Kai Hwang, Zhiwei Xu 著/69元
《数据库系统概念》(英文版)	Abraham Silberschatz, Henry F. Korth, S. Sudarshan 著/65元
《高级计算机体系结构》(英文版)	Kai Hwang 著/59元
《数据结构、算法与应用——C++语言描述(英文版)	Sartaj Sahni 著/66元
《软件工程——实践者的研究方法》(英文版)	Roger S. Pressman 著/68元
《软件工程——Java语言实现》(英文版)	Stephen R. Schach 著/51元

《通信网络基础》(英文版)	Jean Walrand 著/32元
《数据通信与网络》(英文版)	Behrouz Forouzan 著/59元
《离散数学及其应用》(英文版)	Kenneth H. Rosen 著/59元
《计算机体系结构：量化研究方法》(英文版·第3版)	John L. Hennessy, David A. Patterson 著/元
《现代操作系统》(英文版·第2版)	Andrew S. Tanenbaum 著/48元
《组合数学》(英文版·第3版)	Richard A. Brualdi 著/35元
《ISDN、B-ISDN与帧中继和ATM》(英文版·第4版)	William Stallings 著/35元
《数据库系统导论》(英文版·第7版)	C. J. Date 著/65元
《C++编程思想》(英文版·第2版)	Bruce Eckel 著/58元
《程序设计实践》(英文版)	Brian W. Kernighan, Rob Pike 著/22元
《数据库系统实现》(英文版)	Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom 著/42元
《Internet技术基础》(英文版·第3版)	Douglas E. Comer 著/23元
《结构化计算机组成》(英文版·第4版)	Andrew S. Tanenbaum 著/38元
《UNIX环境高级编程》(英文版)	W. Richard Stevens 著/49元
《网络互连：网桥.路由器.交换机和互连协议》(英文版·第2版)	Radia Perlman 著/36元
《TCP/IP详解 卷1：协议》(英文版)	W. Richard Stevens 著/39元
《Java编程思想》(英文版·第2版)	Bruce Eckel 著/69元
《设计模式：可复用面向对象软件的基础》(英文版)	Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides 著/38元
《TCP/IP详解 卷2：实现》(英文版)	Gary R. Wright,W. Richard Stevens 著/69元
《TCP/IP详解 卷3：TCP事务协议、HTTP、NNTP和UNIX域协议》(英文版)	W. Richard Stevens 著/28元
《人本界面——设计交互式系统的最新指示》(英文版)	Jef Raskin 著/28元
《程序设计语言：概念和结构》(英文版·第2版)	Ravi Sethi 著/39元
《C++语言的设计和演化》(英文版)	Bjarne Stroustrup 著/29元
《UNIX操作系统教程》(英文版)	Sarwar 著/49元
《高速网络与因特网——性能与服务质量》(英文版·第2版)	William Stallings 著/45元
《Linux操作系统内核实习》(英文版)	Gary J. Nutt 著/32元
《具体数学：计算机科学基础》(英文版·第2版)	Ronald Graham 著/49元
《计算机图形学原理及实践——C语言描述》(英文版·第2版)	James D. Foley, Andries Van Dam, Steven K. Feiner, John F. Hughes 著/88元
《编写有效用例》(英文版)	Alistair Cockburn 著/25元
《系统分析与设计》(英文版)	John W. Satzinger, Robert B. Jackson, Stephen D. Burd 著/60元
《计算机文化》(英文版·第4版)	Parsons Oja 著/55元
《电磁场与电磁波》(英文版)	Guru 著/68元
《面向对象与经典软件工程》(英文版)	Stephen R. Schach 著
《数据通信与网络教程》(英文版)	William A. Shay 著

《数字逻辑应用与设计》(英文版)	John M. Yarbrough 著
《神经网络设计》(英文版)	Hagan 著
《专家系统原理与编程》(英文版)	Giarratano, Riley 著
《编译原理与实践》(英文版)	Kenneth C. Louden 著
《计算理论导引》(英文版)	Sipser 著

全美经典学习指导系列

《关系数据库习题与解答》(英文版)	Ramon A.Mata-Toledo 著/26元
《计算机体系结构习题与解答》(英文版)	Nicholas Carter 著/30元
《计算机图形学习题与解答》(英文版·第2版)	Zhigang Xiang 著/35元
《数据结构习题与解答——Java 语言描述》(英文版)	John R. Hubbard 著/38元
《C++编程习题与解答》(英文版·第2版)	John R. Hubbard 著/39.5元
《操作系统习题与解答》(英文版)	J.Archer Harris 著/25元
《Java程序设计习题与解答》(英文版)	John R. Hubbaro 著/28元
《计算机科学导论习题与解答》(英文版)	Ramon A. Mata-Toledo 著/30元
《数据结构习题与解答——C++语言描述》(英文版)	John R. Hubbard 著/38元
《软件工程系统与解答》(英文版)	David A. Gustafson 著
《计算机网络习题与解答》(英文版)	Ed Tittel 著
《计算机导论习题与解答》	Pauline K. Cushman 著/薛静锋 等译/29元
《Visual Basic 编程习题与解答》	Byron S. Gottfried 著/周旭 等译/29元
《关系数据库习题与解答》	Ramon A. Mata-Toledo 著/周云晖 等译/19元
《Java 编程习题与解答》	Hubbard 著/王强 等译/29元
《C++编程习题与解答》	John R. Hubbard 著/徐漫江 等译/39元
《数据结构习题与解答——Java语言描述》	John R. Hubbard 著/阳国贵 等译/39元
《计算机图形学习题与解答》	Zhigang Xiang 著/陈泽琳 等译/29元
《SQL 编程习题与解答》	Mata Toledo 著/胡志军 等译/29元

Contents

Chapter 1

Introduction

Compilers are computer programs that translate one language to another. A compiler takes as its input a program written in its **source language** and produces an equivalent program written in its **target language**. Usually, the source language is a **high-level language**, such as C or C++, and the target language is **object code** (sometimes also called **machine code**) for the target machine, that is, code written in the machine instructions of the computer on which it is to be executed. We can view this process schematically as follows:

A compiler is a fairly complex program that can be anywhere from 10,000 to 1,000,000 lines of code. Writing such a program, or even understanding it, is not a simple task, and most computer scientists and professionals will never write a complete compiler. Nevertheless, compilers are used in almost all forms of computing, and anyone professionally involved with computers should know the basic organization and operation of a compiler. In addition, a frequent task in computer applications is the development of command interpreters and interface programs, which are smaller than compilers but which use the same techniques. A knowledge of these techniques is, therefore, of significant practical use.

It is the purpose of this text not only to provide such basic knowledge but also to give the reader all the necessary tools and practical experience to design and pro-

gram an actual compiler. To accomplish this, it is necessary to study the theoretical techniques, mainly from automata theory, that make compiler construction a manageable task. In covering this theory, we do not assume that the reader has previous knowledge of automata theory. Indeed, the viewpoint taken here is different from that in a standard automata theory text, in that it is aimed specifically at the compilation process. Nevertheless, a reader who has studied automata theory will find the theoretical material more familiar and will be able to proceed more quickly through those sections. In particular, Sections 2.2, 2.3, 2.4, and 3.2 may be skipped or skimmed by a reader with a good background in automata theory. In any case, the reader should be familiar with basic data structures and discrete mathematics. Some knowledge of machine architecture and assembly language is also essential, particularly for the chapter on code generation.

The study of the practical coding techniques themselves requires careful planning, since even with a good theoretical foundation the details of the code can be complex and overwhelming. This text contains a series of simple examples of programming language constructs that are used to elaborate the discussion of the techniques. The language we use for this discussion is called TINY. We also provide (in Appendix A) a more extensive example, consisting of a small but sufficiently complex subset of C, which we call C-Minus, which is suitable for a class project. In addition there are numerous exercises; these include simple paper-and-pencil exercises, extensions of code in the text, and more involved coding exercises.

In general, there is significant interaction between the structure of a compiler and the design of the programming language being compiled. In this text we will only incidentally study language design issues. Other texts are available that more fully treat programming language concepts and design issues. (See the Notes and References section at the end of this chapter.)

We begin with a brief look at the history and the raison d'être of compilers, together with a description of programs related to compilers. Then, we examine the structure of a compiler and the various translation processes and associated data structures and tour this structure using a simple concrete example. Finally, we give an overview of other issues of compiler structure, including bootstrapping and porting, concluding with a description of the principal language examples used in the remainder of the book.

1.1 WHY COMPILERS? A BRIEF HISTORY

With the advent of the stored-program computer pioneered by John von Neumann in the late 1940s, it became necessary to write sequences of codes, or programs, that would cause these computers to perform the desired computations. Initially, these programs were written in **machine language**—numeric codes that represented the actual machine operations to be performed. For example,

```
C7 06 0000 0002
```

represents the instruction to move the number 2 to the location 0000 (in hexadecimal) on the Intel 8x86 processors used in IBM PCs. Of course, writing such codes is extremely time consuming and tedious, and this form of coding was soon replaced by