编译原理与技术

李文生 编著

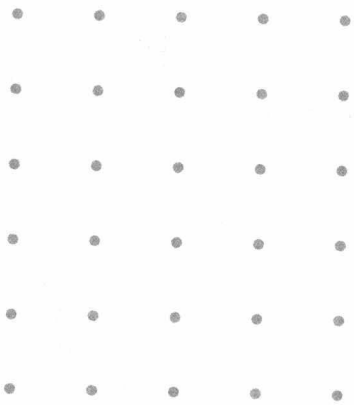
http://www.tup.com.cn

- 根据教育部“高等学校计算机科学与技术专业规范”组织编写
- 与美国 ACM 和 IEEE *Computing Curricula 2005* 同步

清华大学出版社

编译原理与技术

李文生 编著



清华大学出版社
北京

内 容 简 介

本书系统地介绍了编译程序的设计原理和基本实现技术，主要内容包括词法分析、语法分析、语义分析、中间代码生成、代码生成和代码优化等，还重点介绍了用于实现语义分析和中间代码生成的语法制导翻译技术，以及程序运行时存储空间的组织与管理。

在介绍基本理论和方法的同时，注重实际应用，介绍了 LEX 和 YACC 的使用方法及原理，剖析了 PL/0 语言的编译程序，讨论了 GCC 编译程序的基本结构。配合理论教学，给出了一些实践题目，旨在培养学生分析和解决问题的能力。

本书内容充实、图文并茂、各章节内容循序渐进，并注意理论与实践的结合。

本书可作为高等学校计算机科学与技术专业的本科生教材或参考书，也可供其他专业的学生或从事计算机工作的工程技术人员阅读参考。

图书在版编目（CIP）数据

编译原理与技术/李文生编著. —北京：清华大学出版社，2009.1
ISBN 978-7-302-19171-1

Ⅰ. 编…　Ⅱ. 李…　Ⅲ. 编译程序—程序设计　Ⅳ. TP314

中国版本图书馆 CIP 数据核字(2008)第 211342 号

责任编辑：张瑞庆　赵晓宁
责任校对：焦丽丽
责任印制：何　芊

出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**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：清华大学印刷厂
装 订 者：三河市新茂装订有限公司
经　　销：全国新华书店
开　　本：185×230　　**印　张**：25.75　　**字　数**：533 千字
版　　次：2009 年 1 月第 1 版　　**印　次**：2009 年 1 月第 1 次印刷
印　　数：1～4000
定　　价：34.80 元

前言

FOREWORD

“编译原理与技术”是计算机科学与技术专业的专业基础课程，通过本课程的学习，不仅可以提高编程技巧，掌握软件设计的一般方法，而且对计算机系统软件有一个比较清楚的认识和理解，为进一步的学习和研究打下良好的基础。

本书的前身是北京邮电大学出版社出版的《编译程序设计原理与技术》，主要介绍编译程序的设计原理和基本实现技术。根据多年的教学实践，对原书的内容进行了调整、补充和完善，并加强了实践环节。本书主要以 Pascal 和 C 语言为背景、就编译原理和技术有关的主要课题进行了系统和深入的讨论。

全书共分 11 章。第 1 章对编译程序的组成、功能及有关的前后处理器等进行了介绍，读者可以从中了解编译程序的概况，这是其余各章节的基础。第 2 章介绍了有关形式语言与自动机的基本概念，这是学习编译原理必备的基础理论知识。第 3 章引入了词汇、模式等概念，介绍了利用状态转换图手工编写词法分析程序的方法和步骤，并对词法分析程序自动生成工具 LEX 的使用和工作原理作了介绍。第 4 章详细讨论了常用的语法分析技术，如适于手工实现的递归调用预测分析方法、适合利用分析程序自动生成工具实现的完备的 LR 分析技术等，介绍了语法分析程序自动生成工具 YACC 的使用方法等。第 5 章讨论了语法制导翻译技术，介绍了语法制导定义和翻译方案的概念，以及根据语法制导定义和翻译方案设计相应的翻译程序的基本方法，后继的语义分析和中间代码生成就是基于这种技术实现的。第 6 章介绍了语义分析的基本概念和要求，讨论了编译程序所用的重要数据结构——符号表的组织和管理，详细介绍了借助符号表、利用语法制导翻译技术实现类型检查的方法。第 7 章讨论了程序运行时的存储组织与管理问题，介绍程序运行相关的问题及解决方案，有助于读者理解程序设计中的问题，如非局部名字的访问、参数传递机制等。第 8 章介绍了中间语言，讨论了如何利用语法制导翻译技术把一般的程序设计语言结构翻译成中间代码。第 9 章介绍了目标代码生成的思想和一个简单的实现算法。第 10 章简单讨论了常用的代码优化技术。最后一章介绍了编译程序实现的一般方法，剖析了 PL/0 语言的编译程序，介绍了 GCC 编译程序的基本结构，并提供了一个

课程设计题目，按照软件工程的思想，对课程设计提出了基本要求，希望通过实际操作，有助于加深读者对编译原理的理解及对编译技术的掌握。

由于作者水平所限，书中难免存在缺点和不妥之处，真诚地希望得到广大读者和同行、专家的批评指正。

编　者

2008 年 7 月

目　录

CONTENTS

第 1 章 编译概述

众所周知，用高级语言书写的源程序是不能直接在机器上运行的（起码现有的机器不支持），要想运行它并得到预期的结果，首先必须把源程序转换成等价的目标程序，这个过程就是所谓的编译。编译程序是现代计算机系统的基本组成部分之一。

编译原理与技术是计算机工作者所必须具备的专业基础知识。编译程序的设计涉及到程序设计语言、形式语言与自动机理论、计算机体系结构、数据结构、算法分析与设计、操作系统，以及软件工程等各个方面。

本章通过描述程序设计语言、编译程序的组成，及编译程序的工作环境来介绍编译程序相关的基本概念，以便大家了解本课程的主要内容。

1.1 翻译和解释

1.1.1 程序设计语言

正如语言是人们进行交流的媒介和手段一样，在计算机应用领域，程序设计语言充当了人与问题，以及协助解决问题的计算机之间的通信工具。一种高效的程序设计语言应该能够提高计算机程序的开发效率，具有较好的问题表达能力，并在人们通常的非结构化的思维方式与计算机执行所要求的精确性之间架起桥梁。程序设计语言同时也是人与人之间的交流工具，在许多大型软件项目中，项目成功的关键在于程序员是否能读懂别人写的程序代码。

在计算机发展初期，人们直接用机器语言编写程序。机器语言（machine language）很不直观，难写、难读、易出错，查错就更难，错误也不易修改，并且对机器硬件的依赖性很强，移植性很差。程序设计人员必须受过一定的训练并且熟悉计算机硬件，这在很大程度上限制了计算机的推广应用。

之后，出现了符号语言（symbolic language），即用比较直观的符号来代替纯粹数字表

示的机器指令代码，这样使程序便于记忆、阅读和检查，在此基础上又进一步发展为汇编语言(assembler language)。在汇编语言中，除了用直观的助记符代替操作指令以对应一条条的机器指令外，还增加了若干宏指令，每条宏指令对应一组机器指令、完成特定的功能。这些宏指令构成指令码的扩展。汇编语言仍然是依赖于机器的，使用起来还是很不方便，并且程序设计的效率也很低。

为进一步解决这些问题，1954—1957 年 John Backus 等人参照数学语言设计了第一个描述算法的语言(即 FORTRAN 语言)，之后，又相继出现了得到广泛应用的过程性语言，如 1958—1960 年提出的通用程序设计语言 Algol 60，以及由 Algol 派生出来的 C 语言和 Pascal 语言等，这类语言完全摆脱了机器指令形式的约束，用它编写的程序更接近自然语言和习惯上对算法的描述，故称为面向用户的语言。随后，又相继出现了许多专门用于描述某个应用领域问题的专用语言(如用于数据库领域的语言 SQL 等)，这类语言称为面向问题的语言。随着面向对象程序设计技术的出现，面向对象程序设计语言也得到了广泛的推广应用，如C++ 、Java 等。

面向用户的、面向问题的以及面向对象的语言称为高级语言(high-level language)，机器语言和汇编语言称为低级语言。

相对于低级语言，高级语言具有以下优点。

(1) 高级语言更接近于自然语言、独立于机器。

程序设计人员不必了解机器的硬件，对计算机了解甚少的用户也可以学习和使用，一条高级语言的语句对应多条汇编指令或机器指令，编程效率高，所编程序具有良好的可读性，便于交流和维护，并且具有较好的移植性。

(2) 高级语言运行环境透明性。

程序员在编写程序时，不必对程序中出现的变量和常量分配具体的存储单元，不必了解如何将数据的外部形式转换成机器的内部形式等细节，也不必了解程序运行环境是如何建立和维护的，所有这些工作都由“编译程序”完成。

(3) 高级语言具有丰富的数据结构和控制结构，编程效率高。

如数据结构有数组、记录等；控制结构有循环结构、分支结构以及过程调用等。这些结构改善了程序的风格，便于程序设计人员采用科学的方法(如结构化的方法、面向对象的方法)来开发程序，从而提高了程序的规范性、可靠性，降低了开发费用。

正如大家所知，用高级语言编写的程序要想在计算机上执行，必须经过加工处理，使其转换为等价的机器语言程序。这个转换过程就是“编译”。某种高级语言的编译程序加上一些相应的支持用户程序运行的子程序就构成该语言的编译系统。编译系统是计算机系统的重要组成部分。

1.1.2 翻译程序

人类社会存在多种语言，为了相互交流，需要建立各种语言之间的翻译。同样，人与计算机之间的信息交流也存在翻译的问题。每种计算机都有自己独特的指令系统(即机器语言)。虽然可以直接用机器语言编写程序，但很不方便(难写、难读、结构性差)。为解决此问题，设计了许多种高级语言，为了让一种语言投入使用，需要一个语言翻译器把用这种语言书写的程序翻译成计算机能够执行的表示形式或将源程序直接翻译成结果(即直接执行源程序)。直接执行程序的翻译器被称为解释器(即解释程序)，而把程序翻译成另外一种形式的翻译器称为编译器(即编译程序)。

编译程序扫描所输入的源程序，并将其转换为目标程序。通常，源程序是用高级语言或汇编语言编写的，如图 1-1 所示。

图 1-1　编译程序

如果源语言是汇编语言，目标语言是机器语言，则该编译程序称为“汇编程序”；如果源语言为高级语言，目标语言是某种机器的机器语言或汇编语言，则该编译程序称为“编译程序”。图 1-2 说明了高级语言程序的编译和执行阶段。

图 1-2　高级语言程序的编译和运行

实现源程序到目标程序的转换所占用的时间称为编译时间。目标程序是在运行时执行的。由图 1-2 可知，源程序和数据是在不同时间(即分别在编译阶段和运行阶段)进行处理的。

解释程序可以看做是一种模拟器，这种模拟器的“机器语言”就是要被翻译的语言。解释程序解释执行源程序但并不生成目标程序，即它同时处理源程序和数据，如图 1-3 所示。

图 1-3　源程序的解释过程

某些解释程序每次直接分析一条所要执行的源程序语句，这种方法很费时间，极少采用。一种更为有效的方法是将编译程序和解释程序组合起来工作，先由编译程序

将源程序转变为一种中间形式，然后再由解释程序来解释执行该中间代码形式程序。由于这样的解释程序直接执行程序而从不生成目标代码，所以被称为“伪解释器”。

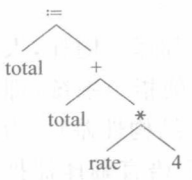


图 1-4　树结构表示

例如，对于 Pascal 语言的赋值语句 total：=total+rate * 4，解释程序先将源语句转换成一棵树（如图 1-4 所示），然后遍历该树，执行结点上所规定的动作。

例如，在根结点处，解释程序发现有一个要执行的赋值操作，它将调用一个程序来处理右边的表达式，当返回时，就可以将计算结果放到与标识符 *total* 相关的存储单元中。在根结点的右子结点上，被调用程序发现它必须计算两个表达式的和，因此，它将递归地调用自己，计算表达式 rate * 4，然后将该值与变量 *total* 的值相加。

目前，解释程序已经用得相当普遍，尤其是在微机环境中，因为解释执行方式便于人机交互。当程序员要根据前面执行的情况随时调整后面的工作，随时更改后面的程序时，采用解释程序是很合适的。

1.2　编译的阶段和任务

按照编译程序的执行过程和所完成的任务，可以把它分成前后两个阶段，即分析阶段和综合阶段。分析阶段根据源语言的定义分析源程序的结构，以检查是否符合语言的规定，确定源程序所表示的对象和规定的操作，并以某种中间形式表示出来。综合阶段根据分析阶段的分析结果构造出所要求的目标程序。

为了记录分析过程中识别出来的标识符及其有关信息，需要使用数据结构“符号表”。如果在编译过程中发现了源程序中的错误，则要向用户报告，为了使编译继续下去，还要对错误进行适当的恢复处理。

编译程序的典型结构如图 1-5 所示。该图给出了一个编译程序的基本组成及步骤。虽然对于不同的高级语言，其编译过程有所变化，但它仍不失为一种典型的编译过程的表示。

1.2.1　分析阶段

分析阶段的任务是根据源语言的定义对源程序进行结构分析和语义分析，从而把源程序正文转换为某种中间表示形式。分析阶段对源程序结构进行静态分析，包括词法分析、语法分析和语义分析。

1. 词法分析

词法分析也称为扫描，是一种线性分析。词法分析程序依次读入源程序中的每个字符，对构成源程序的字符串进行分解，识别出每个具有独立意义的字符串作为记号（token）并组织成记号流，形成记号的字符串叫做该记号的单词（lexeme），把需要存放的