普通高等教育“十一五”国家级规划教材

编译程序设计原理

（第二版）

金成植　金　英　编著

高等教育出版社

普通高等教育“十一五”国家级规划教材

编译程序设计原理

（第二版）

金成植　金　英　编著

高等教育出版社

内容提要

本书是普通高等教育“十一五”国家级规划教材。本书主要介绍了过程式程序设计语言的编译程序构造原理和实现技术。全书共分 11 章，主要包括词法分析和语法分析的理论与技术、语义分析原理与技术、运行时存储空间、动作文法与属性文法技术、中间代码生成、中间代码优化和目标代码生成的原理与技术以及对象式语言编译的基本技术等。

本书的特点是概念清晰，原理论述充分，例子丰富，整体性和实现性强，便于教学和自学，并反映了当前的实用技术。因此适合作为普通高等学校计算机科学与技术及相关专业的教材，亦可作为有关专业人员进一步学习编译程序构造原理和实现技术的参考书。

图书在版编目（CIP）数据

编译程序设计原理/金成植，金英编著. —2 版. —北京：高等教育出版社，2007.5
ISBN 978-7-04-020770-5

Ⅰ. 编… Ⅱ. ①金…②金… Ⅲ. 编译程序－程序设计－高等学校－教材 Ⅳ. TP314

中国版本图书馆 CIP 数据核字（2007）第 049007 号

策划编辑 倪文慧　**责任编辑** 张海波　**封面设计** 于文燕　**责任绘图** 黄建英
版式设计 马静如　**责任校对** 朱惠芳　**责任印制** 尤　静

出版发行	高等教育出版社	**购书热线**	010－58581118
社　　址	北京市西城区德外大街 4 号	**免费咨询**	800－810－0598
邮政编码	100011	**网　　址**	http://www.hep.edu.cn
总　　机	010－58581000		http://www.hep.com.cn
		网上订购	http://www.landraco.com
经　　销	蓝色畅想图书发行有限公司		http://www.landraco.com.cn
印　　刷	潮河印业有限公司	**畅想教育**	http://www.widedu.com
		版　　次	2000 年 7 月第 1 版
开　　本	787×1092　1/16		2007 年 5 月第 2 版
印　　张	30.5	**印　　次**	2007 年 5 月第 1 次印刷
字　　数	690 000	**定　　价**	33.20 元

物料号　20770－00

前　言

现代计算机系统都配有C++等各种语言的编译系统。如果只有高级语言而没有相应的编译系统，那么高级语言就无法在计算机上使用。因此，编译系统如同操作系统一样，是计算机系统中不可缺少的基本组成部分。

通过学习编译程序的构造原理和实现技术，读者不仅可以了解编译程序本身的设计和实现技术，还有助于提高对程序设计语言的理解及设计能力，提高元级程序的设计能力，提高大型软件的开发能力。

软件系统可分为目标级和元级两大类。目标级系统是指以通常的数据作为输入的系统，而元级系统则是以程序作为输入的系统。目标级系统中的数据均可用高级语言的类型机制定义其语法结构，元级系统则包含对程序的操作，而程序的语法结构是用所谓的文法机制来定义的，这种结构是通常语言（包括C++和Java）的类型机制所无法描述的，因此无法直接对程序进行操作。编译系统、解释系统、编辑系统、调试系统、程序分析系统、程序优化系统、程序转换系统、程序并行化系统、程序运行的演示系统等都是典型的元级软件系统。元级系统的共同特点是，都要用到词法分析、语法分析以及源程序到某种内部表示的转换技术，而这些技术在编译技术中表现得最为淋漓尽致，因此可以说编译技术是应用面很广泛的各种元级系统的基本技术。

程序设计语言是用文法系统定义的一套系统化的记法，用于描述计算或非计算过程。按其功能可分为科学计算用语言、商用语言、表处理语言、图形语言、公式处理语言、串处理语言、多用途语言等。按其抽象度可分为低级语言、高级语言和抽象语言。低级语言又分为机器语言和汇编语言，它们面向机器；高级语言和抽象语言面向人，主要用于描述算法，不妨称它们为算法语言。算法语言可分为如下几类：

[1] 过程式语言：FORTRAN，Pascal，Ada，C等

[2] 函数式语言：LISP，HASKELL，ML等

[3] 逻辑式语言：PROLOG等

[4] 对象式语言：Smalltalk，C++，Java等

不同类型语言的编译技术，往往差别比较大，因此无法统一起来介绍其编译技术。编译知识可分为编译理论知识和编译实现知识。编译原理可分为编译器的工作原理和编译器的设计原理。编译技术又可分为基本编译技术和高级编译技术，也可分为传统编译技术和现代编译技术。目前，通常说的现代编译技术往往是指函数式语言和逻辑式等语言的编译技术，已经可以看到这方面的书籍。

编写本书的目的是，要写出适合于普通高等学校、便于教学和自学的编译原理教材。编译系统是非常庞大而复杂的程序系统，而不是数学系统，因此究竟如何介绍编译原理本身就是一种学问。作者曾设计实现过三个编译系统，从事了十多年的“编译原理”课程的教学工作，而且一直坚持编译理论与技术方面的研究工作，根据自己的教学和科研工作，并吸取相关参考文献中的有益部分，经过反复推敲写出了本书。本书的一大特点是讲究可读性、系统性和衔接性。为了醒目地突出每段的重点内容，写法上采用下面的形式，例如

[**静态语义**]……

它表示本段介绍静态语义内容。学完本书内容后，读者将对编译器的整个系统有一个比较完整的概念，而且具备开发中级编译器的实际能力，这也可作为本书的一个特点。编译系统中最重要的是编译原理，因为它具有普遍性，因此本书把重点放在原理的介绍上。

本书是在作者编写的普通高等教育“九五”国家级重点教材《编译程序构造原理和实现技术》的基础上，经过细致、全面的修改和补充完成的。本书仍保持原书的风格，而且概念的描述更加清晰，原理和理论的论述更加充分，例子更加丰富，直观性和理论性也更强。

本书介绍编译原理课程的基本原理和基本技术，而且为了与国外优秀编译原理书接轨，在记法上有所创新。本教材配有 100 多个练习题，读者最好尽可能多做练习，这也是一种最好的学习方法。

本书共分 11 章。第 1 章概述编译程序的功能结构、复杂性及其设计、开发、测试和维护等。第 2 章介绍一种小型语言 ToyL 的编译器，其目的是“解剖麻雀”，通过这一章的学习应掌握两个内容，一是单词概念和词法分析，二是处理简单表达式的语义栈技术。第 3 章介绍正则表达式和有穷自动机及词法分析器的构造，其中介绍正则表达式和自动机的目的完全是为了给出词法分析器的机械化构造方法。第 4 章介绍语法分析方法，其中包括上下文无关文法、LL 分析法、LR 分析法、二义性文法的处理、语法错误恢复、LL 分析法和 LR 分析法的比较等。第 5 章专门介绍语义分析，包括标识符属性的内部表示、类型的内部表示及其构造、符号表及其构造、声明中的语义检查、语句中的语义检查等。第 6 章介绍运行时的存储空间，包括运行时的存储结构、存储分配、活动记录、变量访问环境等。第 7 章介绍动作文法和属性文法，包括动作文法概念、动作文法的实现、属性文法概念、属性文法的实现、动作文法和属性文法的应用。第 8 章介绍中间代码生成，包括中间代码种类、表达式的中间代码、语句的中间代码等。第 9 章介绍中间代码优化，包括常量表达式的优化、公共表达式的优化、循环不变表达式的外提优化、循环归纳表达式的优化等。第 10 章介绍目标代码生成，包括变量的状态描述、寄存器的状态描述、寄存器状态追踪、寄存器分配、编址模式、间接编址模式的转换、目标代码生成的基本技术、基于树结构的代码生成、基于 DAG 的代码生成、代码自动生成系统等。第 11 章介绍对象式语言的编译原理基础。

本书的重要目的之一是提供一本将初级和中级编译原理合为一体的、适合于普通高等学校计算机专业的编译原理教材。在现行教学法的情况下，在一个学期的讲授时间内，讲授完本书的全部内容可能有些困难，本书目录中带*符号的章节为选学内容。即使不讲带*号的章节，也

能构成一本完整的编译原理教材。

关于编译的教学法，作者认为必须以原理为主，因为它具有普遍性。原理包括理论原理和算法原理。要把原理和实现细节全部讲清楚是不可能的，而且没有必要。具体实现算法应作为加深理解原理的辅助工具。教师上课时不需要完整地讲授一个算法程序，而只需讲算法要点，具体算法可让学生自己去读。这样，教师可以把精力主要投入到课程的讲授和原理的研究上，从而提高教学质量，也同时提高教师本身的水平。讲授本课程最好给学生推荐一些著名的国外教材，以便学生多了解情况。最好把编译中的问题转变成程序设计的问题，这样能增强学生对编译原理课的兴趣，也提高他们对编译技术的认识。

在本书的写作过程中，得到了吉林大学吕江花、张晶和东北师范大学郑晓娟的大力帮助，他们提出了许多有益的见解和建议，并反复、全面地检查了整个书稿，在此一并表示感谢。虽然经过多次的修改，但因为书中有些算法是为教学而设计的，因此难免有不足之处，敬请读者多提宝贵意见。

编　者

2007 年 3 月

目　录

第 1 章　编译器概述

1.1　为什么要学习编译技术

通过学习编译技术，可以了解以下一些知识：

[1] 编译系统和操作系统都是计算机系统中的常用基本软件，凡是使用高级程序设计语言编写的应用程序要想顺利运行都离不开编译系统，因此，从事计算机软件开发和高级计算机应用的工作者应了解和掌握编译器的构造原理和工作原理。

[2] 编译系统是涉及多种数据类型和复杂算法的大型软件系统，是一个经典的技术库，其中包含很多优秀的独立技术，因此学习编译技术将会大大提高学习者的编程能力和大型软件的开发能力。

[3] 编译系统是一种特殊软件，其特殊性就在于其操作对象是程序，而不是通常意义上的数据。从性质上看它具有自然语言翻译的性质。程序的特点是其结构不能用通常的类型机制来定义，而必须由语法公式（产生式）来定义。任何 C++等高级程序设计语言都无法描述程序结构，因此它们无法对程序直接进行所需的各种操作，而是必须首先将程序转换成相应语言的某种数据结构。编译器的最独特技术是程序的分解技术，即语法分析技术。我们知道，复杂数据的主要操作是取分量操作，例如下标变量 A[i]和域选择变量 R.x，而对程序来说取分量操作就是取子串的操作，如从一个条件语句的文本分解出其条件表达式的字符串部分、左分支语句的字符串部分以及右分支语句的字符串部分。显然，这种分解操作只能用语法分析等操作才能完成，而这些分解技术在编译技术中表现得最为淋漓尽致。因此，通过学习编译技术，可了解有关程序的各种处理技术。

[4] 软件系统可分为元级软件系统和非元级软件系统。对程序的操作称为元级操作，而将含元级操作的软件系统称为元级软件系统。例如，编译系统、解释系统、程序调试系统、编辑系统、程序分析系统、程序优化系统、程序转换系统、程序并行化系统、程序结构化系统等都是元级软件系统，因此这些系统的开发都需要编译系统的基本技术来支持。

1.2　编译器和解释器

[**语言分类**]：一种程序设计语言是一套系统化的记法，用于描述计算或非计算过程。按其功能可分为科学计算语言、商用语言、表处理语言、图形语言、公式处理语言、串处理语言、

多用途语言等。按其执行模式可分为顺序语言和并行（并发）语言。但一般地，将语言分为低级语言、高级语言和抽象语言三类。低级语言又可分为机器语言和汇编语言。

作为描述算法的语言可分为如下几类：

[1] 过程式语言：如 FORTRAN、Pascal、Ada、C 等。

[2] 函数式语言：如 LISP、HASKELL、ML 等。

[3] 逻辑式语言：如 PROLOG 等。

[4] 对象式语言：如 Smalltalk、C++、Java 等。

本书主要研究顺序过程式语言的编译原理和技术。函数式语言和逻辑式语言，特别是逻辑式语言，其编译技术与过程式语言具有较大的区别，因此不能照搬过程式语言的编译技术。而当前对象式语言的载体基本上是过程式的，因此掌握类似 Pascal 等过程式语言的编译技术，就不难掌握对象式语言的编译技术了。

[**编译器**]：计算机硬件系统只能执行自己的指令程序，而不能执行其他语言程序。因此，想用高级语言，则必须有这样一种程序，它能将用高级语言写成的程序转换成等价的机器语言程序，这种转换程序就是编译程序（Compiler）。通常把编译程序也简称为编译器。编译器的输入对象称为源程序（Source Program），输出对象则称为目标程序（Target Program），目标程序往往是汇编程序。编译器功能如图 1.2.1 所示。

图 1.2.1 编译器的功能

[**解释器**]：解释器（Interpreter）也是语言的一种实现系统，但在概念上与编译器有明显的区别。即解释器是源程序的一个执行系统，而编译器是源程序的一个转换系统。换句话说，解释器的工作结果是源程序的执行结果，而编译器的工作结果则是等价于源程序的某种目标机程序，因此编译器是高级语言程序到某种低级语言程序的转换器。解释器的功能如图 1.2.2 所示。

图 1.2.2 解释器的功能

[**编译器和解释器的比较**]：编译器和解释器在工作性质上具有某些共同点，但若从不同角度对比仍有一些差异，如图 1.2.3 所示。

[**使用解释器的优势**]：相对于编译器而言，使用解释器具有以下几点优势：

[1] 解释器通常是用高级语言写的，因而能够在大多数类型的计算机上运行，而编译器产生的目标代码只能在所选类型的计算机上运行。也就是说，解释器更加通用。

	编 译 器	解 释 器
程序规模	规模较大	规模中小
内部形式	机器代码（低级）	数据结构（高级）
运行机构	硬件 CPU	软件系统
运行速度	相对较快	相对较慢

图 1.2.3 编译器与解释器的比较

[2] 与编译器相比，设计解释器的后端工作量要少得多。

[3] 使用解释器比使用编译器更加安全。这一点对 Java 的流行起了重要作用。

1.3 编译器的功能分解和组织结构

[**编译器的功能分解**]：每种编译器都具有自己独特的组织和工作方式，这是根据源语言的具体特点和对目标语言的具体要求来确定和设计的。因此，不存在一种固定的编译器结构。

本节所讨论的编译器功能结构是指编译器内部都做哪些工作以及它们彼此之间的关系。图 1.3.1 刻画了编译器的功能结构。源程序以文本文件方式，即字符串形式存在。词法分析部分的主要任务是检查词法错误并把源程序中的单词转换成一种内部形式（数据形式）；语法分析部分的任务是检查源程序的语法错误，当发现错误时输出相关信息，并尽可能地继续检查；语义分析的主要任务是构造符号表，并检查语义错误；中间代码生成部分的任务是产生便于优化和便于生成目标代码的中间代码；中间代码优化部分的任务是进行不依赖于目标机的优化，以产生高质量的目标代码；目标代码生成部分的任务是根据目标机的特点由中间代码产生高质量的目标代码。

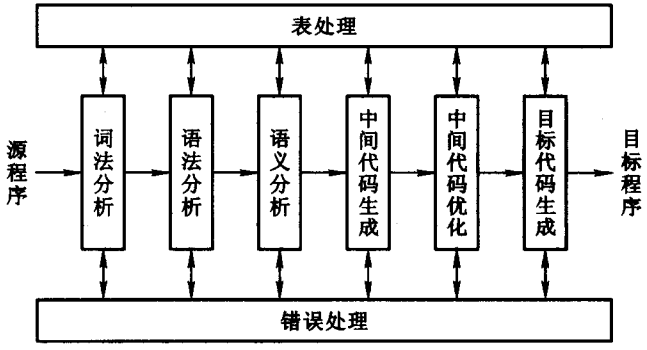

图 1.3.1 编译器的功能分解图

[**编译器前端**]：编译器通常分为前端和后端，其中前端一般包括词法分析、语法分析、

符号表构造、语义分析、中间代码生成、代码优化和错误处理等。此部分工作的特点是不依赖于具体机器。

[**编译器后端**]：后端主要是指中间代码到目标代码生成的阶段。此部分对中间代码和目标机有较高的依赖性，而一般不依赖于源程序代码。至于是否使用符号表，则要视具体的编译器而定。

[**编译器遍数**]：遍数是指编译器为了从源程序代码生成目标代码所进行的加工次数，可分为一遍扫描和多遍扫描。一遍扫描法是指直接从源程序代码产生目标代码；而多遍扫描法是指通过多遍扫描（加工）来产生目标代码。两遍扫描的编译器模式如图 1.3.2 所示。一般来说，大型语言的编译器宜采用多遍扫描法，前一遍的扫描结果是后一遍的输入。多遍编译器的优点是结构和算法清晰，层次分明，易于掌握，它便于优化，便于产生高效的目标代码，也便于移植和修改。一遍扫描法的优点是可避免重复性工作，因此编译速度快；其缺点是当发生语法或语义错误时，前面所做的各种工作全部作废，而且算法不清晰，也不便于分工及优化。如果要产生的是不需做优化处理而且是某种虚拟机上的目标代码，则这种方法是完全可行的。实际上，Pascal 的 P-编译器就是这样一种编译器，它产生的是栈式虚拟机上的目标代码。据估计，50%～75%的实际 Pascal 编译器都源自 P-编译器。

对于编译技术的学习，建议读者学好多遍扫描的编译技术，因为只要掌握了多遍扫描的原理，也就不难实现单遍扫描了。

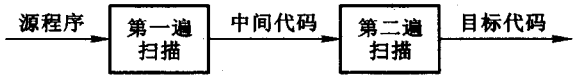

图 1.3.2 两遍扫描的编译器模式

1.4 编译器的伙伴

[**可执行代码**]：可执行代码是指不需要做任何处理就可在计算机上运行的绝对地址代码，它分为两大类，一是汇编代码，二是抽象机代码。实际编译器一般不产生可执行代码。从源程序代码到可执行代码的转换过程如图 1.4.1 所示。

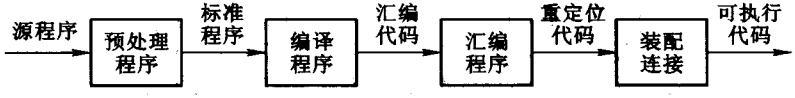

图 1.4.1 从源程序到可执行代码的过程

[**预处理器**]：对于一个编译器而言，如果要处理的输入是对原编译器输入的扩充，包括常见的增加语句、增加宏定义、增加文件包含等扩充方法，那么原编译器就无法处理了。这时