编 译 技 术

王力红　主　编
霍　林　副主编
吴晓红　姜瑛　丛松　编著

重庆大学出版社

内 容 简 介

本书全面、系统地介绍了编译程序的基本结构及编译技术的一般理论和常用方法。主要内容包括：文法和形式语言、有限自动机、语法分析、词法分析、语法制导翻译和中间代码生成、优化、目标代码生成、存储组织与分配、错误的诊察和处理、词法分析与语法分析程序的自动生成等。在语法分析方法中介绍了 LL(1)方法、递归子程序法、算符优先分析法和 LR(K)分析法等。本书大多数算法采用 C 语言描述，并采用 C 编写的 TINY 语言为样例语言。

本书在内容组织上循序渐进，叙述简洁明了、条理清楚，重点突出，算法详尽，例题和习题丰富，易于教学和自学。并有《编译技术上机指导》一书与之配套。

本书可作为各类高等学校计算机专业的教材，也可供从事计算机开发和研究的科技人员参考。

图书在版编目(CIP)数据

编译技术/王力红主编．—重庆：重庆大学出版社，2001.9

计算机科学与技术专业本科系列教材

ISBN 7-5624-2357-1

Ⅰ．编…　Ⅱ．王…　Ⅲ．编译技术—高等学校—教材　Ⅳ．TP314

中国版本图书馆 CIP 数据核字(2001)第 057761 号

编 译 技 术

王力红　主编

责任编辑　周立　彭宁

*

重庆大学出版社出版发行

新 华 书 店 经 销

重庆大学建大印刷厂印刷

*

开本：787×1092　1/16　印张：16.25　字数：406 千

2001 年 9 月第 1 版　　2001 年 9 月第 1 次印刷

印数：1—6000

ISBN 7-5624-2357-1/TP・313　定价：23.00 元

前言

编译技术是计算机专业的一门重要专业课。其中所涉及的编译程序构造的基本原理、实现技术以及抽象问题和解决问题的方法，不仅适用于编译程序的设计，也广泛应用于其他软件的开发。因此，本书除了可作为各类高等学校计算机专业的教材，还可供从事计算机开发和研究的科技人员参考。

本书介绍了将高级程序设计语言翻译成低级语言的一般原理、主要方法和实现技术。包括文法和形式语言、有限自动机、词法分析、语法分析、语法制导翻译和中间代码生成、优化、目标代码生成、存储组织与分配、错误的诊察和处理、词法分析与语法分析程序的自动生成等内容。本书的主要特点是条理清楚，重点突出，算法详尽，例题和习题丰富，易于教学和自学。同时结合当前计算机专业的教学实际和发展趋势，书中大多数算法采用 C 语言描述，并采用 C 语言编写的 TINY 语言为样例语言。

本教材的参考课堂教学时数为 50～72 学时。各章节既有联系，又相对独立，可以根据具体情况取舍。

编译技术是一门实践性较强的课程，建议另安排 10 学时左右的实验课。本书另有《编译技术课程设计指导》一书与之配套。

本书由王力红主编，霍林任副主编，吴晓红、姜瑛和丛松参加编写。第 7、10 章和附录由霍林编写，第 4、5、8 章由吴晓红编写，第 3、11 章由姜瑛编写，第 12、13 章由丛松编写，第 1、2、6、9 章由王力红编写。全书由王力红统稿。

本书是在作者对编译技术多年教学实践的基础上，参考各种参考资料完成的，在此谨对这些参考文献的作者致以诚挚的谢意。

由于水平有限，书中定有不妥之处，恳请读者批评指正。

编　者

2001 年 7 月

目录

此为试读,需要完整PDF请访问: www.ertongbook.com

第1章 概 论

计算机系统是由硬件和软件组成的。硬件是计算机赖以存在的物质基础，而软件则是计算机与使用者沟通的工具和桥梁。可以说软件的功能和质量在很大程度上左右着整个计算机系统的功能。软件的种类很多，按照其在人机交互中的地位和作用，可以分为系统软件和应用软件。其中系统软件是一类处于底层的支持各种具体应用的软件，如操作系统、编译程序和诊断程序等都属于系统软件。其中，编译程序与操作系统一样，已成为当今任何计算机系统最重要的系统程序之一。

本书将就编译程序的基本结构及编译技术的一般理论和常用方法作系统介绍。

1.1 程序设计语言及编译程序

程序设计语言是程序编写工具的总称，可分为两大类：一类为低级语言，包括机器语言、汇编语言等面向机器的语言，一类为高级语言，如 C、PASCAL、FORTRAN、BASIC 等语言。低级语言是计算机最早的语言，由于它对机器的依赖性强、直观性差、编写强度大、对编写人员的要求高，因而只适合于在计算机结构方面具有相当水平并经过专门训练的程序设计人员。于是，为充分发挥计算机的作用，便于使用计算机，高级语言应运而生。20 世纪 50 年代中叶，第一个高级语言 FORTRAN 问世。高级语言具有更接近于自然语言的语法结构，可移植性强，且易读易用易维护，一出现便得到广泛应用。

但是，原本只能读懂自己的指令系统，只能直接执行用相应机器语言编写的代码程序的计算机硬件，如何运行高级语言编写的程序呢？答案是将高级语言程序翻译成机器语言指令代码。完成这种翻译工作的程序就称为翻译程序。具体地说，所谓翻译程序，就是指能够把一种语言（源语言）编写的程序（源程序），翻译成与之等价的另一种语言（目标语言）的程序（目标程序）的程序。

通常，翻译程序可分为解释程序、汇编程序和编译程序。所谓解释程序是一种将源程序按动态顺序逐句进行分析解释翻译，边解释边执行、不产生目标程序的一种翻译程序。这种翻译程序结构简单、占用内存较少，易于在执行过程中对源程序进行修改，但工作效率低，只适合一些规模较小的语言，如解释 BASIC 等。汇编语言也是一种翻译程序，它的源语言和目标语言分别是汇编语言和机器语言。而编译程序（也称编译器）是源语言为某种高级语言，目标语言为相应于某一计算机的汇编语言或机器语言的一种翻译程序。这种编译程序将源程序翻译成执行时可完全独立于源程序的经优化的目标语言代码，因而运行效率高。更重要的是，它使工作于高级语言环境下的程序设计人员，不必考虑与机器有关的繁琐细节，却能完成几乎机器语

言所能完成的所有工作。编译程序的功能如图 1.1 所示。

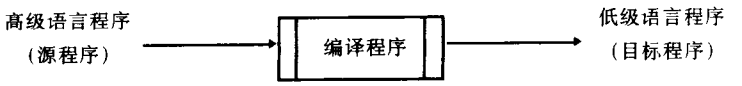
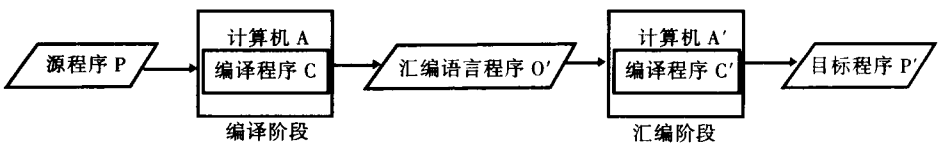
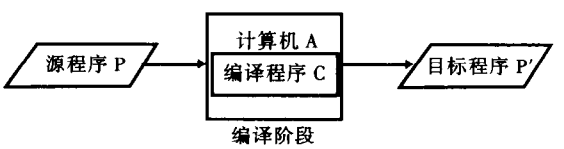


图 1.1　编译程序的功能

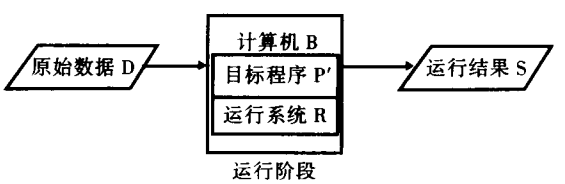
由定义可知,高级语言程序按编译方式的执行过程一般可分为两个阶段:编译阶段和运行阶段。其中,编译阶段完成由源程序到目标程序的翻译,若目标程序是汇编语言程序,还需再通过汇编程序进一步翻译成机器语言程序。而运行阶段的任务是在目标计算机上执行编译阶段所得到的目标程序。但目标程序往往不能由计算机直接执行,一般还应有运行系统进行配合,这个运行系统包括链接程序和由这样一些子程序组成的系统库,如标准函数计算子程序、数组动态存储子程序等。由链接程序将目标程序和系统库连接在一起,最终形成一个可执行程序,在计算机上直接执行。当然,系统库中的子程序越多,功能越强,编译程序本身就越简明紧凑。图 1.2 示出了编译程序的一般执行过程。



(a)高级语言程序到目标程序的翻译阶段(需进行汇编的情况)



(b)高级语言程序到目标程序的翻译阶段(无需进行汇编的情况)



(c)目标程序的运行阶段

图 1.2　高级语言程序按编译方式的执行过程

通常将编译程序、链接程序、系统库、源程序编辑程序等软件组成的系统称为编译系统。

1.2　编译过程和编译程序结构

1.2.1　编译程序的编译过程

由前面我们知道了编译程序是一种将高级语言程序翻译成机器语言程序或汇编语言程序的翻译器,那么编译程序是如何实现对高级语言的翻译的呢?下面对照自然语言翻译来讨论编译程序的编译过程。

考虑一篇英文文章。如果要将其翻译成汉语，必须进行两方面的工作：首先要对原文进行分析，理解其所要表达的思想，然后再根据这一思想进行综合，构造出恰当的汉语表达。

具体地说，自然语言的书面翻译应包括这样几步：第 1 步应从认识单词入手，通过查词典等手段逐行逐字识别文中所有的单词，了解各单词的词性、意义和用法；第 2 步根据英语的语法规则，分析各句子的语法成分，检查语法的正确性；第 3 步推敲出原文所表达的语义，根据该语义拟定汉语的初步译稿；第 4 步则对译稿进行修辞润色加工；最后写出译文。

因此，相应地，程序设计语言的编译过程一般也可以分为词法分析、语法分析、语义分析和中间代码生成、代码优化、目标代码生成 5 个阶段。

此外，由于语言本身的一些信息和编译过程中所收集到的源程序的各种信息需要被保留在各种表格里供随时调用，因而编译过程中需要有建表、查表、更新数据等有关表格的管理工作。在编译过程中发现源程序有错误时，需要准确报告错误种类和错误出现的位置，并应有一定的“校错”能力，因而在编译过程中还需要进行专门的错误处理。

(1)词法分析

词法分析是编译过程的第一个阶段。该阶段的主要任务是从构成源程序的字符串中识别出一个个具有独立意义的最小语法单位——单词，并指出其属性。因此词法分析阶段的输入是源程序的字符串序列，而输出则是源程序的一种等价程序——单词序列，供语法分析阶段进一步处理。

词法分析输出的单词常用形如

(类别码，值)

的二元组表示。

如 C 语言的赋值语句

x=a+b*2

经词法分析后被识别出有 7 个单词，组成如下的单词序列(设标识符、运算符、整常数的类别码分别为 1、2、5)：

(1,“x”)
(2,“=”)
(1,“a”)
(2,“+”)
(1,“b”)
(2,“*”)
(5,“2”)

此外，词法分析在将源程序由字符串序列转换成单词串序列的同时，还可以检查出源程序中的词法错误，如拼写错误等。

(2)语法分析

语法分析阶段的主要任务是在词法分析的基础上，根据相应程序设计语言的语法规定进行语法分析，在源程序的单词串中识别出各种语法成分，如“程序”、“语句”、“表达式”等，并确定整个输入串是否构成一个语法上正确的程序，检查出语法错误。

如上述赋值语句的单词串经语法分析后，表明其语法结构是正确的，即它是一个左部为简单变量、右部为算数表达式的赋值语句。

(3)语义分析和中间代码生成

与自然语言一样，任何程序设计语言都具有两方面的特征，即语法特征和语义特征。前者用来定义各语法成分的形式或结构，后者则用来规定其含义和功能。实际上所谓程序的语义就是它的“意思”。

因此顾名思义，语义分析和中间代码生成阶段的主要任务有两个，其一是进行相应的语义检查，以保证源程序在语义上的正确性，其二是确定各语法成分的含义、用途，收集类型信息，确定应进行的运算和操作，并将其转换为某种内部表示形式，即生成中间代码。

例如，对前面的赋值语句，如果收集到的类型信息为：x，a，b 为整型变量，且 a，b 已被赋值，则该语句的语义正确。同时该语句的含义是将表达式 a＋b＊2 的值赋给 x，设采用四元式为中间代码，则可翻译为如图 1.3 的四元式序列。

```
①(*,b,2,T1)          /* 相当于 T1=b*2   */
②(+,a,T1,T2)         /* 相当于 T2=a+T1  */
③(=,T2, ,x)          /* 相当于 x=T2     */
```

图 1.3　赋值语句 x＝a＋b＊2 的中间代码

语义正确性检查的项目十分繁杂，常见的项目有，说明句中是否有矛盾的类型说明，表达式的运算符是否有类型不匹配的运算对象，在过程调用中实在参数与形式参数在个数、次序、类型等方面是否匹配，等等。

由于到目前为止，对程序设计语言的语义还没有一个公认的形式化描述系统，因此常采用一种半机械化的方法解决语义分析问题，如本书将要介绍的“语法制导翻译方法”，该方法将语义分析与语法分析有机地结合起来，在语法分析的同时进行语义分析，产生中间代码。

(4)代码优化

代码优化阶段的任务是对上一阶段产生的中间代码进行变换和改造，以期获得高质量的目标代码。所谓高质量，是指程序的时空效率高，即程序所占存储空间小，运行时间少。

例如，如图 1.3 的四元式序列可优化为图 1.4 的四元式。

```
①(*,b,2,T1)          /* 相当于 T1=b*2   */
②(+,a,T1,x)          /* 相当于 x=a+T1   */
```

图 1.4　赋值语句 x＝a＋b＊2 优化后的中间代码

代码优化的方法很多，将在第 11 章作详细介绍。

应该指出，优化处理常常是以增加编译程序本身的时空复杂度和可靠性为代价的。对于某些优化项目，时空效率本身也是相互矛盾的。因此在设计一个编译程序时，究竟应考虑哪些优化项目，每个优化项目应进行到何种程度，均应权衡利弊，视具体情况而定。

(5)目标代码生成

目标代码生成阶段是编译过程的最后阶段。这一阶段的任务是把优化后的中间代码转换成特定机器上的绝对指令代码，或可重定位的指令代码，或汇编指令代码，由于这种转换工作与具体的目标计算机的系统结构及其指令系统有关，涉及到各种变量的存储空间分配、寄存器的调度、各种硬件功能部件的使用等，因此工作比较复杂。

例如，图 1.4 所示的四元式可以转换为图 1.5 所示的目标代码。

```
MOV   B,R0
MUL   #2,R0
ADD   A,R0
MOV   R0,X
```

图 1.5　赋值语句 x=a+b*2 的目标代码

在上述 5 个阶段中，前 3 个阶段属于对源程序进行分析的范畴，而后 2 个阶段的工作则属于综合的范畴。

应该说明的是，上面只是按一种典型处理模式对编译过程各阶段进行了划分。实际上并非所有的编译程序都分成上述 5 个阶段。如有的编译器，编译过程不产生中间代码，因而没有中间代码优化这一阶段。有的编译器，优化不仅在中间代码级进行而且还在目标代码级进行，因而多了目标代码优化阶段。而 PL/0 语言编译程序，语法分析后直接产生目标代码，因而其编译过程可分为：词法分析、语法分析、目标代码生成 3 个阶段。但对于大多数实用的编译程序，其编译过程通常包括上述 5 个阶段。

1.2.2　编译程序的结构

(1)编译程序的逻辑结构

前面讨论了编译过程的 5 个阶段及表格处理和错误处理的工作，这些工作通常由相应的程序或模块来完成。因此编译程序通常由词法分析程序、语法分析程序、语义分析和中间代码生成程序、代码优化程序、目标代码生成程序、表格处理程序和错误处理程序 7 个程序组成，其逻辑结构如图 1.6 所示。

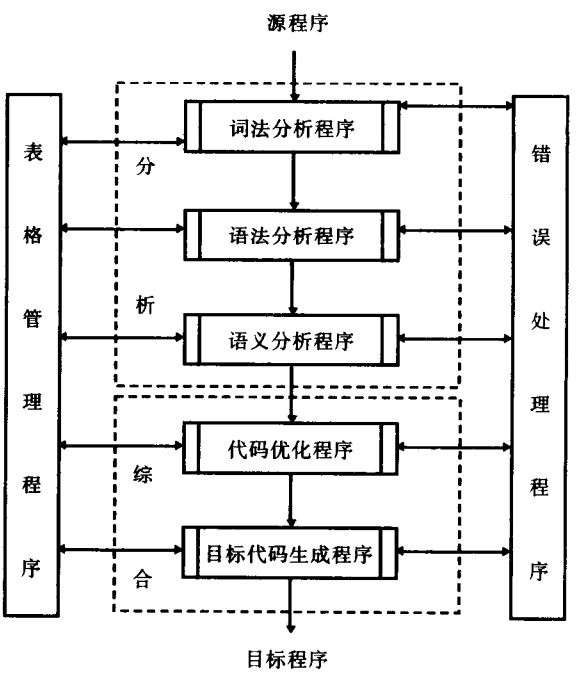


图 1.6　编译程序的逻辑结构

(2)编译程序的组织

图 1.6 给出了编译程序的逻辑结构，但这并不表明各部分之间实际执行的先后顺序。编

此为试读，需要完整PDF请访问：www.ertongbook.com

译程序的执行流程与编译过程中对源程序的扫描遍数有很大关系。

编译程序按其扫描次数可分为单遍(趟)扫描和多遍(趟)扫描。所谓一遍(pass)或一趟，是指在编译过程中对源程序或与之等价的中间程序从头到尾扫描一遍，并将其转换成下一个中间程序的整个过程。

PL/0 语言是一个典型的单遍(趟)扫描程序，其编译程序的结构是以语法分析程序作为主程序，词法分析程序和代码生成程序作为独立的子程序。每当语法分析程序需要一个单词时，便调用词法分析程序，每当经语法分析确认语法正确需要产生目标代码时，便调用代码生成程序。图 1.7 示出 PL/0 编译程序的结构。

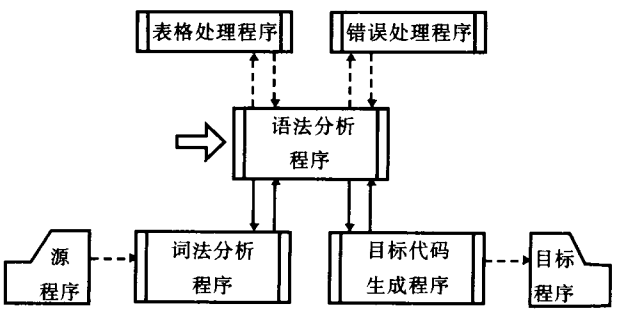


图 1.7　PL/0 语言编译程序的结构

大多数带有优化的编译程序都需要多遍扫描，如 IBM DOS PASCAL 就是一个多遍扫描的编译程序。典型的三遍扫描程序可以这样安排：第一遍专为词法和语法分析，可检测出绝大部分语法错误；第二遍用于语义分析和中间代码优化；第三遍生成目标代码(及目标代码的优化)。更深层的优化可能需要更多的遍。

采用多遍扫描方式，各遍的工作分工明确，便于多人合作开发，缩短开发周期，同时节省运行时的内存空间，易于提高目标程序的质量。但各遍之间的重复工作多，编译效率低。而单遍(趟)扫描的编译程序，编译速度快、效率高，但运行时占用空间大，目标程序质量低。

至于在设计一个编译程序时，究竟采用几遍扫描的方式，应根据语言的结构、目标计算机的规模、设计技术指标、设计人员多少等具体因素而定。

1.3　编译程序的实现途径

编译程序的开发是一个非常艰巨耗时的过程。最早人们使用机器语言或汇编语言用手工方式来开发编译程序，效率非常低。随着编译技术的发展和软件自动化水平的提高，编译程序的开发周期正在缩短。本节将分别讨论编译程序的开发技术和自动生成问题。

1.3.1　编译程序的开发技术

在本节之前只提到源语言和目标语言对编译器结构的影响。实际上任何一个编译程序都涉及到 3 个语言：源语言、目标语言和编译器的编写语言。由于编写第一个编译器时不存在任何其他编译器，因而该编译程序只能用机器语言编写，而现在各种编译器的存在，为用高级语言编写编译程序提供了可能，从而大大提高了编译程序的可读性、可维护性、可移植性和可靠

性，提高了开发效率。

假定一个编译程序的源语言为 A，目标语言为 B，编写语言为 C，则用如图 1.8 的 T 型图可以方便地表示一个编译程序所涉及到的三个语言的关系：

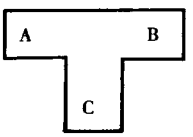


图 1.8　T 型图

下面利用 T 型图，讨论自展、交叉编译、自编译和移植等编译器的主要开发技术。

(1)自展

自展(也称自举)的基本思想是：若要编写语言 L 的编译程序，首先用目标机的机器语言或汇编语言编写一个 L 的核心子集 L_1 的编译程序，然后再以 L_1 为编写语言，编写 L 的一个更大子集 L_2 的编译程序，$L_1 \subset L_2$，…，依此类推，像滚雪球一样，最后生成语言 L 的编译程序。即经过 n 级编译程序后，有

$$L_1 \subset L_2 \subset \cdots \subset L_n = L$$

编译程序的自展过程如图 1.9 所示。

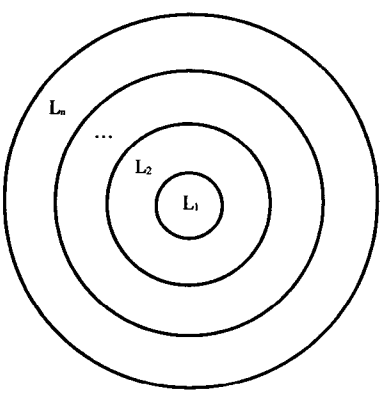
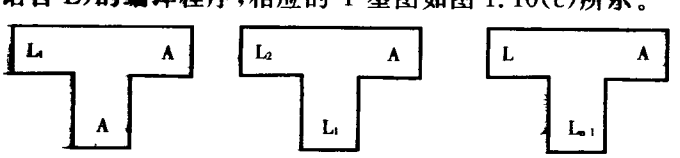


图 1.9　编译程序的自展

现在用 T 型图表示自展。设语言 L 的目标语言为某计算机上的机器语言 A，编写 L_1 的编译程序的机器语言也为 A，则自展的第一步可以用图 1.10(a)的 T 型图表示。自展的第二步是用语言 L_1 编写 L_2 的编译程序，相应的 T 型图如图 1.10(b)所示。自展的第 n 步是用语言 L_{n-1} 编写 L_n(即语言 L)的编译程序，相应的 T 型图如图 1.10(c)所示。



(a)自展的第 1 步　　(b)自展的第 2 步　　(c)自展的第 n 步(最后一步)

图 1.10　编译程序自展过程的 T 型图

如果把自展过程的各 T 型图组合起来，可以得到图 1.11 的组合 T 型图。图中“⇒”后的 T 型图为前面两级编译程序的等价 T 型图。

(2)交叉编译与自编译

前面利用自展技术在目标机器上实现了语言 L 的编译程序，语言 L 的编译程序可以运行于目标机并产生该机器的目标代码。现在的问题是，希望能够利用该目标机上的语言 L 的编译程序产生另一机器的目标代码。这种由 A 机器上的编译程序产生 B 机器上的目标代码的技术称为交叉编译。

交叉编译的 T 型图如图 1.12 所示，设 A 和 B 分别为机器 A 和机器 B 上的机器语言或汇编语言。

与交叉编译对应，如果用某语言书写其自身的编译程序，这种技术称为自编译。机器 A 上实现的语言 L 的自编译，其 T 型图如图 1.13 所示。

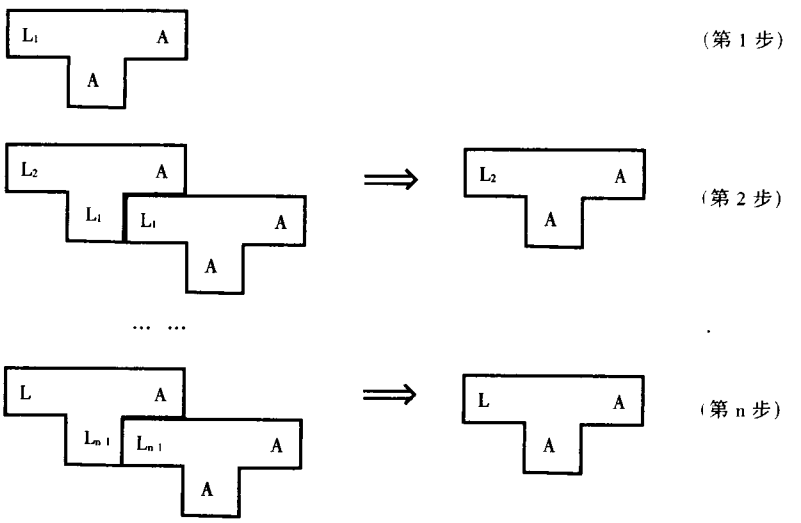


图 1.11　自展过程的组合 T 型图

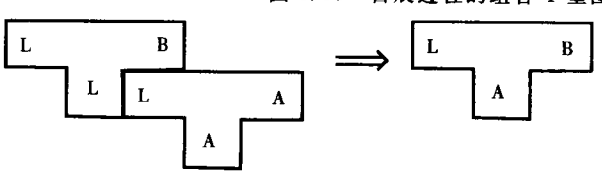


图 1.12　交叉编译的 T 型图

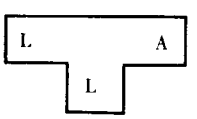


图 1.13　自编译的 T 型图

(3)移植

前面的交叉编译是利用 A 机器上的编译程序为 B 机器产生目标代码。但是在更多的情况下，更希望将 A 机器上的某语言的编译程序搬到 B 机器上运行，这种技术称为移植。

将 A 机器上的语言 L 移植到 B 机器上，可以通过以下两步实现：

①实现如图 1.12 所示的 A 机器上的语言 L 到 B 机器上的交叉编译，这种交叉编译器的源语言 L 将作为下一步的编写语言。

②利用①中的交叉编译器的源语言 L 编写 B 机器上的自编译程序，从而使语言 L 可在机器 B 上运行，即实现语言 L 的移植。如图 1.14 所示。

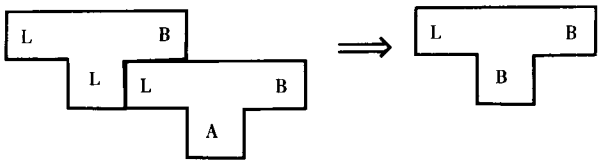


图 1.14　移植过程②的 T 型图

1.3.2　编译程序的自动生成问题

前面介绍了编译程序的实现技术。借助于这些技术，可以大大提高编译程序的开发效率和编译程序的质量。但是尽管如此，目前研制一个编译程序仍然是十分艰巨和耗时的，因此实现开发过程的自动化一直是人们追求的目标，并首先在一些领域取得了突破。由于形式语言的文法解决了程序设计语言的词法和语法的形式化描述的问题，因而与编译程序的后端相比，编译程序的前端，词法分析程序和语法分析程序的自动生成更容易实现。

最有代表性的自动生成工具是 20 世纪 70 年代由 Bell 实验室推出的基于 UNIX 操作系统的 LEX 和 YACC，目前已得到广泛应用。其中 LEX 是词法分析程序的自动构造工具，而 YACC 是语法分析程序的自动构造工具。本书将在第 5 章和第 7 章分别介绍它们的使用。

习　题　1

1-1　说明解释程序和编译程序的区别。

1-2　简述高级语言程序按编译方式的执行过程。

1-3　什么是编译系统？

1-4　编译过程通常有哪几个阶段？简述各阶段的主要任务。

1-5　什么是编译程序的遍？对编译程序来说，遍多和遍少各有什么优缺点？

1-6　一个最简单的编译程序至少要包含哪几个组成部分(程序)？

1-7　构造一个编译程序有哪些途径？

1-8　如果在 A 机器上已有 PASCAL 语言的编译程序，希望利用它实现

①在 A 机器上的 C 语言的编译程序；

②在 B 机器上的 C 语言的编译程序。

试分别设计实现方案，并给出各步骤的 T 型图。

1-9　LEX 和 YACC 分别是什么软件工具？

第2章 文法和形式语言

与操作系统等其他系统软件的开发类似，在构造程序设计语言的编译程序之前，首先必须制定该程序设计语言的标准规范，然后根据该规范，进一步确定分析方法、实现编译程序。这就需要对程序设计语言本身有一个确切的描述。因此编译理论首先就要解决程序设计语言的描述问题。

1956 年著名语言学家 N. Chomsky 提出了一种用来描述语言的数学系统——形式语言，为程序设计语言甚至自然语言语法结构的形式化描述(形式语法)提供了数学工具。尽管到目前为止，语义的形式化描述(形式语义)还未能完全解决，但形式语言已成为计算机科学的重要组成部分，对程序设计语言的设计与编译实现产生了极为深远的影响。

所谓形式化描述是指采用一整套带有严格规定的符号体系来描述问题的理论和方法，与非形式化描述相比，其优点是描述准确严密且便于符号间的推演。由于形式语义学尚不成熟，本书将不涉及语义的形式化描述。以后凡提到定义语言均是指对语言语法结构的描述。按 Chomsky 理论，描述语言语法结构的主要工具是文法，这是形式语言和编译理论的基础。本章将围绕文法和语言的概念介绍形式语法。

2.1 符号和符号串

符号和符号串是构成语言的基本单位。在自然语言中，英语的基本符号是 26 个字母、数字、标点符号，由若干基本符号按一定要求组成的符号串构成了英语单词，而由若干单词按一定语法结构组成的符号串序列则构成英文句子，所有英文句子的集合便是英语。汉语的组成也类似，只不过中文的基本符号是汉字、数字和标点符号而已。在程序设计语言中，C 语言的基本符号是字母、数字、运算符、标点符号等各类符号，保留字、标识符、函数名、程序名等是“单词”，语句、函数、程序是“句子”，所有句子的集合构成 C 语言。所以，与自然语言一样，程序设计语言是具有一定语法结构的符号串序列的集合。因此，符号和符号串是形式语言和编译理论中最基本的概念。为此首先讨论符号、符号串的有关概念。

2.1.1 字母表和符号串

1)字母表　元素的非空有穷集合。习惯上用大写字母表示。

例如，字母表 $\Sigma=\{a,b,c\}$

字母表 $V=\{0,1\}$

2)符号或字符　字母表中的元素。

例如，字母表 Σ={a,b,c}中的元素 a,b,c 是字母表 Σ 上的符号；

字母表 V={0,1}中的元素 0,1 是字母表 V 上的符号。

3)符号串　符号的有穷序列。

例如，字母表 Σ={a,b,c}上的符号串有

a,b,c,aa,ab,ac,ba,bb,…,aaa,…

注意，符号串与符号的组成顺序有关，如，ab 与 ba 是两个不同的符号串。

4)空符号串　不含任何符号的符号串，记为 ε。

5)符号串集合　字母表 Σ 上的符号串组成的集合。

例如，若字母表 Σ={a,b,c}，则

A={a,ab}，B={aa,ab,ac}

均为该字母表上的符号串集合。

2.1.2　符号串的运算

1)符号串的长度　符号串中所包含的符号个数。设符号串为 x，则 x 的长度记为|x|。

例如，|a|=1，|abc|=3，

|ε|=0。

2)符号串的连接　设有符号串 x 和 y，把 y 的所有符号相继写在 x 的符号之后所得到的符号串称为 x 与 y 的连接，记为 xy。

例如，若 x=We，y=are，则

xy=Weare，yx=areWe。

且有，|xy|=|yx|=|x|+|y|。

此外，显然有　εx=xε=x。

但一般来说，xy≠yx。

若 x 和 y 都是字母表∑上的符号串，则 xy 和 yx 也是∑上的符号串。

3)符号串的方幂　设 x 是符号串，则 x 的 n 次连接称为 x 的 n 次方幂，记为 x^n。即

$x^0=\varepsilon$，$x^1=x$，$x^2=xx$

……

一般地，若 n>0，有　$x^n=x^{n-1}x=xx^{n-1}$。

例如，若 x=abc，则

$x^0=\varepsilon$，$x^1=abc$，$x^2=abcabc$，…，$x^n=abcabc\cdots abc$(n 个 abc 相连接)。

4)符号串的前缀、后缀和子串　设 x 是一个符号串，则从 x 尾部截去若干个(含 0 个)符号之后所余下的部分称为 x 的前缀，从 x 头部截去若干个(含 0 个)符号之后所余下的部分称为 x 的后缀，而从 x 中截去一个前缀和一个后缀之后所余下的部分称为 x 的子串。

例如，若 x=abcd，则

ε,a,ab,abc,abcd 是 x 的前缀，

abcd,bcd,cd,d,ε 是 x 的后缀，

而 ε,a,ab,abc,abcd,bcd,cd,d,bc,b,c 是 x 的子串。

可见，符号串 x 的所有前缀和后缀都是 x 的子串，但 x 的子串不一定是其前缀或后缀。

特别，ε 和 x 既是 x 的前缀和后缀，也是 x 的子串。

此为试读，需要完整PDF请访问：www.ertongbook.com