高等院校计算机科学与技术专业“十一五”规划教材

编译原理教程

（第三版）

主编　胡元义

西安电子科技大学出版社
http://www.xduph.com

高等院校计算机科学与技术专业“十一五”规划教材

编译原理教程

(第三版)

主　编　胡元义

副主编　邓亚玲　谈姝辰

西安电子科技大学出版社

内 容 简 介

本书系统地介绍了编译程序的设计原理及实现技术，主要内容包括：绪论、词法分析、语法分析、语义分析和中间代码生成、代码优化、目标程序运行时存储空间的组织、目标代码生成、符号表与错误处理等。

在内容的组织上，本书强调知识的实用性，将编译的基本理论与具体的实现技术有机地结合起来，既注重了理论的完整性，化繁为简，又将理论融于具体的实例中，化难为易，以达到准确、清楚地阐述相关概念和原理的目的。本书注重各章节对理论阐述的条理性，书中给出的例子也具有较强的实用性与连贯性，使读者对编译的各个阶段有一个全面、直观的认识。本书采用的算法全部由C语言描述，各章均附有习题。

本书可作为计算机本科专业的教材，也可作为计算机软件工程人员的参考资料。

★ 本书配有电子教案，需要者可登录出版社网站，免费下载。

图书在版编目(CIP)数据

编译原理教程/胡元义主编. —3版. —西安：西安电子科技大学出版社，2010.10

高等院校计算机科学与技术专业“十一五”规划教材

ISBN 978-7-5606-2463-1

Ⅰ. 编… Ⅱ. 胡… 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(2010)第 138681 号

策　　划　马乐惠

责任编辑　张晓燕　马乐惠

出版发行　西安电子科技大学出版社(西安市太白南路 2 号)

电　　话　(029)88242885　88201467　　邮　　编　710071

网　　址　www.xduph.com　　电子邮箱　xdupfxb001@163.com

经　　销　新华书店

印刷单位　陕西天意印务有限责任公司

版　　次　2010 年 10 月第 3 版　2010 年 10 月第 9 次印刷

开　　本　787 毫米×1092 毫米　1/16　印　张　15.5

字　　数　359 千字

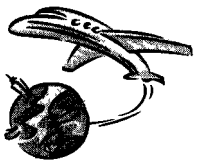
印　　数　34 001～37 000 册

定　　价　23.00 元

ISBN 978-7-5606-2463-1/TP • 1229

XDUP 2755003-9

高等院校计算机科学与技术专业“十一五”规划教材
编审专家委员会名单



前　言

计算机语言之所以能由单一的机器语言发展到现今的数千种高级语言，就是因为有了编译技术。编译技术是计算机科学中发展得最迅速、最成熟的一个分支，它集中体现了计算机发展的成果与精华。

编译原理是计算机专业的一门核心课程，在计算机本科教学中占有十分重要的地位。编译原理课程具有很强的理论性与实践性，读者学习起来普遍感到内容抽象、不易理解。为此，本书采取由浅入深、循序渐进的办法介绍编译原理的基本概念和实现方法。在内容的组织上，本书将编译的基本理论与具体的实现技术有机地结合起来，既注重了理论的完整性，化繁为简，又将理论融于具体的实例中，化难为易，以达到准确、清晰地阐述相关概念和原理的目的。本书注重各章节对理论阐述的条理性，书中给出的例子也具有较强的实用性与连贯性，使读者对编译的各个阶段有一个全面、直观的认识，从而透彻地领悟编译原理的精髓。本书采用的算法全部用 C 语言描述，文法也尽可能采用 C 语言文法。

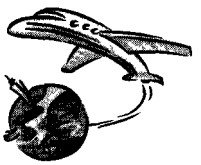
本书共分八章。第一章简要介绍了编译的基本概念。第二章介绍了词法分析的相关内容，主要涉及正规表达式与有限自动机。第三章主要介绍语法分析，首先简要地介绍了文法的有关概念，然后介绍了自顶向下语法分析方法——递归下降分析法和 LL(1)分析法，最后介绍了自底向上语法分析方法——算符优先分析法和 LR 分析法。第四章介绍了语法制导翻译与中间代码生成的有关内容，给出了如何在语法分析的同时进行语义加工并产生出中间代码的方法。第五章介绍了代码优化的有关内容，主要涉及基本块优化和循环优化；此外，还增加了“全局优化概述”一节，以便读者对优化代码有一个全面、完整的了解。第六章介绍了程序运行时存储空间的组织。第七章讨论目标代码生成的有关内容，讲述了如何由中间代码产生出最终目标代码。第八章简要地介绍了符号表的组织与错误处理的方法。书中带“*”的章节可以不讲或选讲。

为了便于读者正确理解有关概念，各章还配有一定数量的习题。这些习题大多选自本科生和研究生的考试试题，也包括作者结合多年教学实践经验设计出来的典型范例，力求使读者抓住重点、突破难点，进一步全面、深入地巩固所学知识。

由于水平所限，书中难免存在一些缺点和不足之处，恳请广大读者批评指正。

编　者

2010 年 3 月



第二版前言

计算机语言之所以能由单一的机器语言发展到现今的数千种高级语言，就是因为有了编译技术。编译技术是计算机科学中发展得最迅速、最成熟的一个分支，它集中体现了计算机发展的成果与精华。

编译原理是计算机专业的一门核心课程，在计算机本科教学中占有十分重要的地位。编译原理课程具有很强的理论性与实践性，读者学习起来普遍感到内容抽象、不易理解。为此，本书采取由浅入深、循序渐进的办法介绍编译原理的基本概念和实现方法。在内容的组织上，本书将编译的基本理论与具体的实现技术有机地结合起来，既注重了理论的完整性，化繁为简，又将理论融于具体的实例中，化难为易，以达到准确、清晰地阐述相关概念和原理的目的。除了各章节对理论阐述的条理性之外，书中给出的例子也具有实用性与连贯性，使读者对编译的各个阶段有一个全面、直观的认识，从而透彻地领悟编译原理的精髓。本书采用的算法全部用 C 语言描述，文法也尽可能采用 C 语言文法。

本书共分八章。第一章简要介绍了编译的基本概念。第二章介绍了词法分析的有关内容，主要涉及正规表达式与有限自动机。第三章主要介绍语法分析，首先简要地介绍了文法的有关概念，然后介绍了自上而下语法分析方法——递归下降分析法和 LL(1)分析法，最后介绍了自下而上语法分析方法——算符优先分析法和 LR 分析法。第四章介绍了语法制导翻译与中间代码生成的有关内容，给出了如何在语法分析的同时进行语义加工并产生出中间代码的方法。第五章介绍了代码优化的有关内容，主要涉及基本块优化和循环优化。第六章介绍了程序运行时存储空间的组织。第七章讨论目标代码生成的有关内容，讲述了如何由中间代码产生出最终的目标代码。第八章简要地介绍了符号表的组织与错误处理的方法。书中带“*”的章节可以不讲或选讲。

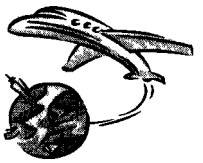
为了便于读者正确理解有关概念，各章还配有一定数量的习题。这些习题大多选自本科生和研究生的考试试题，也包括作者结合多年教学实践经验设计出来的典型范例，力求使读者抓住重点、突破难点，进一步全面、深入地巩固所学知识。书中习题的解答在作者的另一本教学辅导书《<编译原理教程>习题解析与上机指导(第二版)》（西安电子科技大学出版社）中给出。

本书在第一版的教学实践基础上，综合使用本教材师生的反馈意见，对原教材内容进行了修编。由于反馈意见对原教材的整体结构评价较好，因而此次只对本书的内容进行了局部修改，以使所讲授的内容更加清晰和翔实，有益于对编译理论的理解和掌握。

由于作者水平所限，书中难免存在某些缺点和错误，恳请广大读者批评指正，同时对关心本教材的读者表示衷心的感谢！

编　者

2006 年 4 月



第一版前言

计算机语言之所以能由单一的机器语言发展到现今的数千种高级语言，就是因为有了编译技术。编译技术是计算机科学中发展得最迅速、最成熟的一个分支，它集中体现了计算机发展的成果与精华。

编译原理是计算机专业的一门核心课程，在计算机本科教学中占有十分重要的地位。编译原理课程具有很强的理论性与实践性，使读者学习起来普遍感到内容抽象、不易理解。本书采取由浅入深、循序渐进的办法介绍编译原理的基本概念和实现方法。在内容的组织上，本书将编译的基本理论与具体的实现技术有机地结合起来，既注重了理论的完整性，化繁为简，又将理论融于具体的实例中，化难为易，以达到准确、清晰地阐述相关概念和原理的目的。除了各章节对理论阐述的条理性之外，书中给出的例子也具有实用性与连贯性，使读者对编译的各个阶段有一个全面、直观的认识，从而透彻地领悟编译原理的精髓。本书采用的算法全部用 C 语言描述，文法也尽可能采用 C 语言文法。

本书共分八章。第一章简要介绍了编译的基本概念。第二章介绍了词法分析的有关内容，主要涉及正规表达式与有限自动机。第三章主要介绍语法分析，首先简要地介绍了文法的有关概念，然后介绍了自上而下语法分析方法——递归下降分析法和 LL(1)分析法，最后介绍了自下而上语法分析方法——算符优先分析法和 LR 分析法。第四章介绍了语法制导翻译与中间代码生成的有关内容，给出了如何在语法分析的同时进行语义加工并产生出中间代码的方法。第五章介绍了代码优化的有关内容，主要涉及基本块优化和循环优化。第六章介绍了程序运行时存储空间的组织。第七章讨论目标代码生成的有关内容，讲述了如何由中间代码产生出最终的目标代码。第八章简要地介绍了符号表的组织与错误处理的方法。书中带“*”的章节可以不讲或选讲。

为了便于读者正确理解有关概念，各章还配有一定数量的习题。这些习题大多选自本科生和研究生的考试试题，也包括作者结合多年教学实践经验设计出来的典型范例，力求使读者抓住重点、突破难点，进一步全面、深入地巩固所学知识。书中习题的解答在作者的另一本教学辅导书《<编译原理教程>习题解析与上机指导》（西安电子科技大学出版社）中给出。

由于水平所限，书中难免存在一些缺点和错误，恳请广大读者批评指正。

编　者

2003 年 3 月

目　录

第一章 绪 论

计算机的诞生是科学发展史上的一个里程碑。经过半个多世纪的发展，计算机已经改变了人类生活、工作的各个方面，成为人类不可缺少的工具。计算机之所以能够如此广泛地被应用，应当归功于高级程序设计语言。计算机语言之所以能由最初单一的机器语言发展到现今数千种高级语言，就是因为有了编译程序。没有高级语言，计算机的推广应用是难以实现的；而没有编译程序，高级语言就无法使用。编译理论与技术也是计算机科学中发展得最迅速、最成熟的一个分支，它集中体现了计算机发展的成果与精华。

1.1 程序设计语言和编译程序

为了处理和解决实际问题，每一种计算机都具有其特定的功能，而这些功能是通过计算机执行一系列相应的操作来实现的。计算机所能执行的每一种操作称为一条指令，计算机能够执行的全部指令集合就是该计算机的指令系统。由于计算机硬件的器件特性，决定了计算机本身只能直接接受由 0 和 1 编码的二进制指令和数据，这种二进制形式的指令集合称为该计算机的机器语言，它是计算机惟一能够直接识别并接受的语言。

用机器语言编写程序很不方便且容易出错，编写出来的程序也难以调试、阅读和交流。为此，出现了用助记符代替机器语言(二进制编码)的另一种语言，这就是汇编语言。汇编语言是建立在机器语言之上的，因为它是机器语言的符号化形式，所以较机器语言直观；但是计算机并不能直接识别这种符号化语言，用汇编语言编写的程序必须翻译成机器语言之后才能执行，这种“翻译”是通过专门的软件——汇编程序实现的。

尽管汇编语言与机器语言相比在阅读和理解上有了长足的进步，但其依赖具体机器的特性是无法改变的，这给程序设计增加了难度。随着计算机应用需求的不断增长，出现了更加接近人类自然语言的功能更强、抽象级别更高的面向各种应用的高级语言。高级语言已经从具体机器中抽象出来，摆脱了依赖具体机器的问题。用高级语言编制的程序几乎能够在不改动的情况下在不同种类的计算机上运行且不易出错，这是汇编语言难以做到的，但高级语言程序翻译(编译)成最终能够直接执行的机器语言程序的难度却大大增加了。

由于汇编语言和机器语言一样都是面向机器的，故相对于面向用户的高级语言来说，它们都被称为低级语言，而 FORTRAN、PASCAL、C、ADA、Java 这类面向应用的语言则称为高级语言。因此，编译程序就是指这样一种程序，通过它能够将用高级语言编写的源程序转换成与之在逻辑上等价的低级语言形式的目标程序，见图 1-1。

图 1-1　编译程序的功能

一个高级语言程序的执行通常分为两个阶段，即编译阶段和运行阶段，如图 1-2 所示。编译阶段将源程序变换成目标程序；运行阶段则由所生成的目标程序连同运行系统(数据空间分配子程序、标准函数程序等)接受程序的初始数据作为输入，运行后输出计算结果。

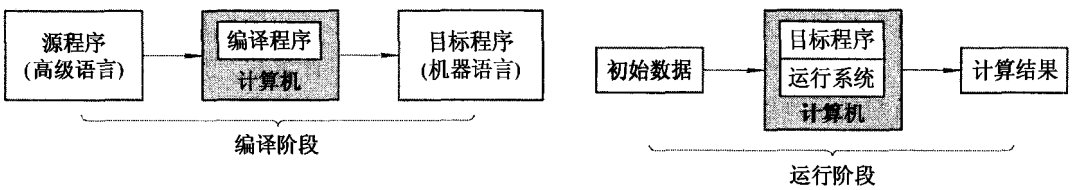


图 1-2　源程序的编译和运行阶段

如果编译生成的目标程序是汇编语言形式的，那么在编译与运行阶段之间还要添加一个汇编阶段，它将编译生成的汇编语言目标程序再经过汇编程序变换成机器语言目标程序，如图 1-3 所示。

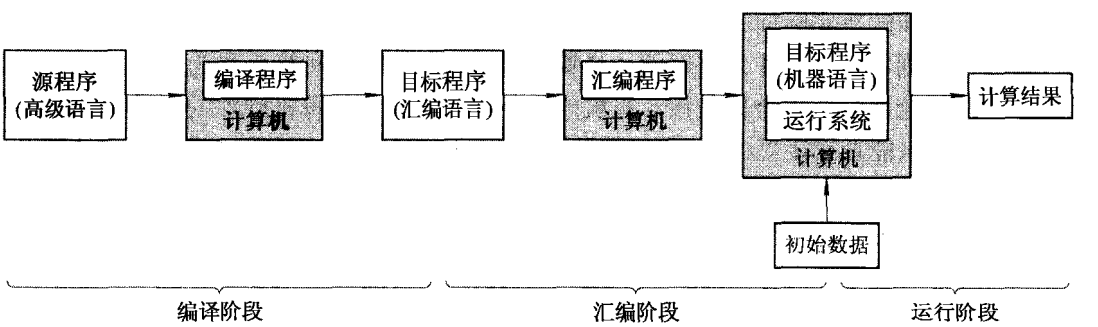


图 1-3　源程序的编译、汇编和运行阶段

用高级语言编写的程序也可通过解释程序来执行。解释程序也是一种翻译程序，它将源程序作为输入，一条语句一条语句地读入并解释执行，如图 1-4 所示。解释程序与编译程序的主要区别是：编译程序将源程序翻译成目标程序后再执行该目标程序；而解释程序则逐条读出源程序中的语句并解释执行，即在解释程序的执行过程中并不产生目标程序。典型的解释型高级语言是 BASIC 语言。

图 1-4　解释程序解释执行过程示意

1.2 编译程序的历史及发展

20 世纪 40 年代，由于冯·诺伊曼在存储-程序计算机方面的开创性工作，计算机可以执行编写的一串代码或程序，这些程序最初都是用机器语言(Machine Language)编写的。机器语言就是计算机能够执行的全部指令集合的二进制形式，例如：

C7 06 0000 0002

表示在 IBM PC 上使用的 Intel 8x86 处理器将数字 2 移至地址 0000(十六进制)的指令。用机器语言编写程序很不方便且容易出错，因此这种代码形式很快就被汇编语言(Assembly Language)取代。在汇编语言中，指令和存储地址都以符号形式给出。例如，汇编语言指令

MOV X, 2

就与前面的机器指令等价(假设符号存储地址 X 是 0000)。汇编程序(Assembler)将汇编语言的符号代码和存储地址翻译成与机器语言相对应的二进制代码。汇编语言大大提高了编程的速度和准确性，至今人们仍在使用它，在存储容量小和速度快的要求下尤其如此。但是，汇编语言依赖于具体机器的特性是无法改变的，这给编程和调试增加了难度。很明显，编程技术发展的下一个重要步骤就是用更简洁的数学定义或自然语言来描述和编写程序，它应与任何机器无关，而且也可通过一个翻译程序将其翻译为可在计算机上直接执行的二进制代码。例如，前面的汇编语言代码 MOV X, 2 可以写成一个简洁的与机器无关的形式 X=2。

1954～1957 年，IBM 的 John Backus 带领一个研究小组对 FORTRAN 语言及其编译器进行了开发。但是，由于对编译程序的理论及技术研究刚刚开始，这个语言的开发付出了巨大的辛劳。与此同时，波兰语言学家 Noam Chomsky 开始了他的自然语言结构研究。Noam Chomsky 根据文法(Grammar，产生语言的规则)的难易程度及识别它们所需的算法对语言进行了分类，定义了 0 型、1 型、2 型和 3 型这四类文法及其相应的形式语言，并分别与相应的识别系统相联系。2 型文法(上下文无关文法，Context-free Grammar)被证明是程序设计语言中最有用的文法，它代表着目前程序设计语言结构的标准。Noam Chomsky 的研究结果最终使得编译器结构异常简单，甚至还具有自动化功能。有限自动机(Finite Automata)和正规表达式(Regular Expression)与上下文无关文法紧密相关，它们与 Noam Chomsky 的 3 型文法相对应，并引出了表示程序设计语言的单词符号形式。接着又产生了生成有效目标代码的方法——这就是最初的编译器，它们被沿用至今。随着对语法分析研究的深入，重点转到对编译程序的自动生成的研究。开发的这种程序最初被称为编译程序的编译器，但因为它们仅仅能够自动完成编译器的部分工作，所以更确切地应称为分析程序生成器(Parser Generator)。这些程序中最著名的是 Steve Johnson 在 1975 年为 UNIX 系统编写的语法分析器自动生成工具 YACC(Yet Another Compiler-Compiler)。类似地，有限自动机的研究也产生出另一种称为词法分析器的自动生成工具(Scanner Generator)Lex。

20 世纪 70 年代后期和 80 年代初，大量的研究都关注于编译器其他部分的自动生成，其中包括代码生成。这些努力并未取得多少成功，主要是由于这部分工作过于复杂，这方面的内容在此不再赘述。

现今编译器的发展包括了更为复杂的算法应用程序，它用于简化或推断程序中的信息，

这又与具有此类功能的更为复杂的程序设计语言发展结合在一起。其中典型的有用于函数语言编译 Hindley-Milner 类型检查的统一算法。目前，编译器已经越来越成为基于窗口的交互开发环境(Interactive Development Environment，IDE)的一部分，这个开发环境包括了编辑器、链接程序、调试程序以及项目管理程序。尽管近年来对 IDE 进行了大量的研究，但是基本的编译器设计在近 20 年中都没有多大的改变。

现代编译技术已转向对并行编译的研究，由于本书重点是介绍经典的编译理论和技术，因此，对并行编译的发展不再介绍。

1.3　编译过程和编译程序结构

编译程序的工作过程是指从输入源程序开始到输出目标程序为止的整个过程。此过程是非常复杂的。一般来说，整个编译过程可以划分成五个阶段：词法分析阶段、语法分析阶段、语义分析和中间代码生成阶段、优化阶段和目标代码生成阶段。

1. 词法分析

词法分析的任务是输入源程序，对构成源程序的字符串进行扫描和分解，识别出一个个单词符号，如基本字(if、for、begin 等)、标识符、常数、运算符和界符(如“(”、“)”、“=”、“；”)等，将所识别出的单词用统一长度的标准形式(也称内部码)来表示，以便于后继语法工作的进行。因此，词法分析工作是将源程序中的字符串变换成单词符号流的过程，词法分析所遵循的是语言的构词规则。

2. 语法分析

语法分析的任务是在词法分析的基础上，根据语言的语法规则(文法规则)把单词符号流分解成各类语法单位(语法范畴)，如“短语”、“子句”、“句子(语句)”、“程序段”和“程序”。通过语法分析可以确定整个输入串是否构成一个语法上正确的“程序”。语法分析所遵循的是语言的语法规则，语法规则通常用上下文无关文法描述。

3. 语义分析和中间代码生成

这一阶段的任务是对各类不同语法范畴按语言的语义进行初步翻译，包含两个方面的工作：一是对每种语法范畴进行静态语义检查，如变量是否定义、类型是否正确等；二是在语义检查正确的情况下进行中间代码的翻译。注意，中间代码是介于高级语言的语句和低级语言的指令之间的一种独立于具体硬件的记号系统，它既有一定程度的抽象，又与低级语言的指令十分接近，因此转换为目标代码比较容易。把语法范畴翻译成中间代码所遵循的是语言的语义规则，常见的中间代码有四元式、三元式、间接三元式和逆波兰记号等。

4. 优化

优化的任务是对前阶段产生的中间代码进行等价变换或改造，以期获得更为高效(节省时间和空间)的目标代码。常用的优化措施有删除冗余运算、删除无用赋值、合并已知量、循环优化等。例如，其值并不随循环而发生变化的运算可提到进入循环前计算一次，而不必在循环中每次循环都进行计算。优化所遵循的原则是程序的等价变换规则。

5. 目标代码生成

这一阶段的任务是把中间代码(或经优化处理之后)变换成特定机器上的机器语言程序

或汇编语言程序，实现最终的翻译工作。最后阶段的工作因为目标语言的关系而十分依赖硬件系统，即如何充分利用机器现有的寄存器，合理地选择指令，生成尽可能短且有效的目标代码，这些都与目标机器的硬件结构有关。

上述编译过程的五个阶段是编译程序工作时的动态特征，编译程序的结构可以按照这五个阶段的任务分模块进行设计。编译程序的结构示意如图 1-5 所示。

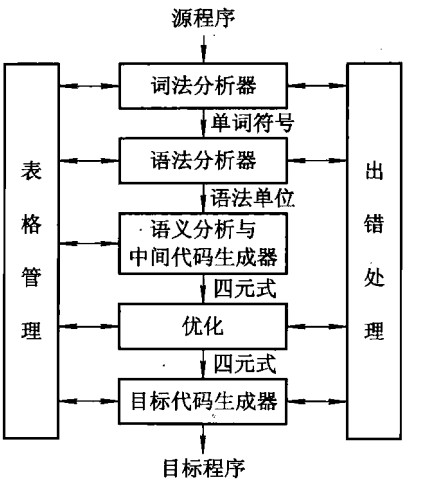


图 1-5 编译程序结构示意

编译过程中源程序的各种信息被保留在不同的表格里，编译各阶段的工作都涉及到构造、查找或更新有关的表格，编译过程的绝大部分时间都用在造表、查表和更新表格的事务上，因此，编译程序中还应包括一个表格管理程序。

出错处理与编译的各个阶段都有联系，与前三个阶段的联系尤为密切。出错处理程序应在发现错误后，将错误的有关信息如错误类型、出错地点等向用户报告。此外，为了尽可能多地发现错误，应在发现错误后还能继续编译下去，以便发现更多的错误。

一个编译过程可分为一遍、两遍或多遍完成，每一遍完成所规定的任务。例如，第一遍只完成词法分析的任务，第二遍完成语法分析和语义加工工作并生成中间代码，第三遍再实现代码优化和目标代码生成。当然，也可一遍即完成整个编译工作。至于一个编译过程究竟应分为几遍，如何划分，这和源程序语言的结构与目标机器的特征有关。分多遍完成编译过程可以使整个编译程序的逻辑结构更清晰一点，但遍数多势必增加读写中间文件的次数，从而消耗过多的时间。

1.4 编译程序的开发

由于计算机语言功能的完善、硬件结构的发展、环境界面的友好等都对编译程序提出了更多、更高的要求，因而构造一个编译系统并非易事。虽然编译理论和编译技术的不断发展已使编译程序的生产周期不断缩短，但是要研制完成一个编译程序仍需要相当长的时间，工作也相当艰巨。因此，如何高效、高质量地生成一个编译程序一直是计算机系统设计人员追求的目标。

编译程序的任务是把源程序翻译成某台计算机上的目标程序，因此，开发人员首先要熟悉这种源程序语言，对源程序语言的语法和语义要有准确无误的理解。此外，开发人员还需确定编译程序的开发方案及方法，这是编译开发过程中最关键的一步，其作用是使编译程序具有易读性和易改性，以便将来对编译程序的功能进行更新和扩充。选择合适的语言编写编译程序也是非常重要的，语言选择不当会使开发出来的编译程序可靠性较差，难以维护且质量也无法保证。目前大部分编译程序都是用 PASCAL、C 和 ADA 这类高级语言编写的，不仅减少了开发的工作量，也缩短了开发周期。最后，开发人员对目标机要有深入的研究，这样才能充分利用目标机的硬件资源和特点，产生质量较高的目标程序。

编译程序的开发常常采用自编译、交叉编译、自展和移植等技术实现。

1. 自编译

用某种高级语言书写自己的编译程序称为自编译。例如，假定 A 机器上已有一个 PASCAL 语言可以运行，则可以用 PASCAL 语言编写出一个功能更强的 PASCAL 语言编译程序，然后借助于原有的 PASCAL 编译程序对新编写的 PASCAL 编译程序进行编译，从而在编译后即得到一个能在 A 机器上运行的功能更强的 PASCAL 编译程序。

2. 交叉编译

交叉编译是指用 A 机器上的编译程序来产生可在 B 机器上运行的目标代码。例如，若 A 机器上已有 C 语言可以运行，则可用 A 机器中的 C 语言编写一个编译程序，它的源程序是 C 语言程序，而产生的目标程序则是基于 B 机器的，即能够在 B 机器上执行的低级语言程序。

以上两种方法都假定已经有了一个系统程序设计语言可以使用，若没有可使用的系统程序设计语言，则可采用自展或移植的办法来开发编译程序。

3. 自展

自展的方法是：首先确定一个非常简单的核心语言 L_0，然后用机器语言或汇编语言编写出它的编译程序 T_0；再把语言 L_0 扩充到 L_1，此时有 $L_0 \subset L_1$，并用 L_0 编写 L_1 的编译程序 T_1(即自编译)；然后再把语言 L_1 扩充为 L_2，此时有 $L_1 \subset L_2$，并用 L_1 编写 L_2 的编译程序 T_2……这样不断扩展下去，直到完成所要求的编译程序为止。

4. 移植

移植是指 A 机器上的某种高级语言的编译程序稍加改动后能够在 B 机器上运行。一个程序若能较容易地从 A 机器搬到 B 机器上运行，则称该程序是可移植的。移植具有一定的局限性。

用系统程序设计语言来编写编译程序虽然缩短了开发周期并提高了编译程序的质量，但实现的自动化程度不高。实现编译程序的最高境界是能够有一个自动生成编译程序的软件工具，只要把源程序的定义以及机器语言的描述输入到该软件中，就能自动生成这个语言的编译程序，如图 1-6 所示。

图 1-6　编译程序自动生成示意

计算机科学家和软件工作者为了实现编译程序的自动生成进行了大量研究工作，随着形式语言学研究的发展和编译程序自动生成工作的进展，已经出现了一些编译程序自动生成系统，如 UNIX 操作系统下的软件工具 Lex 和 YACC 等。

1.5　构造编译程序所应具备的知识内容

要在某一台机器上为某种语言构造一个编译程序，必须掌握下述三方面的内容：

(1) 对被编译的源语言(如 C、PASCAL 等)，要深刻理解其结构(语法)和含义。例如，下面的 for 循环语句：

```
for(i=1;i<=10+i;i++)
    x=x+1;
```