免费提供电子教案

高等院校规划教材
计算机科学与技术系列

编译方法

贺 汛 编著

机械工业出版社
CHINA MACHINE PRESS

高等院校规划教材·计算机科学与技术系列

编译方法

贺汛 编著

机械工业出版社

本书介绍关于程序设计语言方面的编译程序设计技术，主要由编译程序的基本结构、语言的基础知识、编译过程各阶段的工作原理与实现方法三大部分组成。

本书内容通俗易懂，叙述简明，突出实践，适合作为高校计算机软件和应用专业的教材使用，亦可供从事计算机应用和软件的工程技术人员阅读自学。与本书配套学习的实验辅导用书——《编译方法学习指导与实践》已由机械工业出版社出版。

图书在版编目(CIP)数据

编译方法/贺汛编著. —北京：机械工业出版社，2007.7
(高等院校规划教材·计算机科学与技术系列)
ISBN 978-7-111-21801-2

Ⅰ.编… Ⅱ.贺… 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 096997 号

机械工业出版社(北京市百万庄大街 22 号 邮政编码 100037)
策　　划：胡毓坚
责任编辑：张　化
责任印制：杨　曦
北京市朝阳展望印刷厂印刷
2007 年 7 月第 1 版·第 1 次印刷
184mm×260mm ·15.5 印张·363 千字
0001—5000 册
标准书号：ISBN 978－7－111－21801－2
定价：23.00 元

凡购本书，如有缺页，倒页，脱页，由本社发行部调换
销售服务热线电话：(010)68326294
购书热线电话：(010)88379639　88379641　88379643
编辑热线电话：(010)88379739

出版说明

计算机技术的发展极大地促进了现代科学技术的发展，明显地加快了社会发展的进程。因此，各国都非常重视计算机教育。

近年来，随着我国信息化建设的全面推进和高等教育的蓬勃发展，高等院校的计算机教育模式也在不断改革，计算机学科的课程体系和教学内容趋于更加科学和合理，计算机教材建设逐渐成熟。在“十五”期间，机械工业出版社组织出版了大量计算机教材，包括“21世纪高等院校计算机教材系列”、“21世纪重点大学规划教材”、“高等院校计算机科学与技术‘十五’规划教材”、“21世纪高等院校应用型规划教材”等，均取得了可喜成果，其中多个品种的教材被评为国家级、省部级的精品教材。

为了进一步满足计算机教育的需求，机械工业出版社策划开发了“高等院校规划教材”。这套教材是在总结我社以往计算机教材出版经验的基础上策划的，同时借鉴了其他出版社同类教材的优点，对我社已有的计算机教材资源进行整合，旨在大幅提高教材质量。我们邀请多所高校的计算机专家、教师及教务部门针对此次计算机教材建设进行了充分的研讨，达成了许多共识，并由此形成了“高等院校规划教材”的体系架构与编写原则，以保证本套教材与各高等院校的办学层次、学科设置和人才培养模式等相匹配，满足其计算机教学的需要。

本套教材包括计算机科学与技术、软件工程、网络工程、信息管理与信息系统、计算机应用技术以及计算机基础教育等系列。其中，计算机科学与技术系列、软件工程系列、网络工程系列和信息管理与信息系统系列是针对高校相应专业方向的课程设置而组织编写的，体系完整，讲解透彻；计算机应用技术系列是针对计算机应用类课程而组织编写的，着重培养学生利用计算机技术解决实际问题的能力；计算机基础教育系列是为大学公共基础课层面的计算机基础教学而设计的，采用通俗易懂的方法讲解计算机的基础理论、常用技术及应用。

本套教材的内容源自致力于教学与科研一线的骨干教师与资深专家的实践经验和研究成果，融合了先进的教学理念，涵盖了计算机领域的核心理论和最新的应用技术，真正在教材体系、内容和方法上做到了创新。同时本套教材根据实际需要配有电子教案、实验指导或多媒体光盘等教学资源，实现了教材的“立体化”建设。本套教材将随着计算机技术的进步和计算机应用领域的扩展而及时改版，并及时吸纳新兴课程和特色课程的教材。我们将努力把这套教材打造成为国家级或省部级精品教材，为高等院校的计算机教育提供更好的服务。

对于本套教材的组织出版工作，希望计算机教育界的专家和老师能提出宝贵的意见和建议。衷心感谢计算机教育工作者和广大读者的支持与帮助！

机械工业出版社

前　言

编译技术是计算机语言发展的支柱，也是计算机科学中发展最迅速、最成熟的分支之一。编译原理课程是继《C语言程序设计》、《离散数学》、《数据结构》等课程后，对工科计算机专业开设的一门重要的核心课程，在教学中占有十分重要的地位。编译的一般原理和基本技术不仅适用于构造程序设计语言的编译程序，也适用于各种系统软件、应用软件的设计和实现。在本课程中，前期课程的大量内容得到了广泛的应用，这对这些课程内容的理解和巩固也起到了良好的作用。

本教材共分三大部分。第一部分即第1章，简单介绍编译的基本过程和编译程序的基本结构；第二部分即第2章，介绍有关语言的基本知识，包括语言的概念、语言的形式描述和分析方法；从第3章开始按编译的过程逐一介绍编译的各个阶段的任务、原理和实现的基本技术，包括第3章的词法分析方法即词法分析程序的设计、第4、5章的自顶向下和自底向上两大类语法分析方法、第6章的自底向上语法制导翻译技术、第9章中间代码优化中的基本块优化技术和循环优化技术，以及第10章的基本块目标代码生成方法；另外第7章介绍了编译程序的重要组成部分——符号表，以及有关符号表的组织与操作；第8章介绍了与编译相关的程序运行时的存储空间组织问题。

因为编译技术理论性较强、较难理解，也较为繁琐，所以本书尽量以通俗易懂的语言叙述形式给出详细算法。书中选入了大量例题，在每一章的最后特意列出重点，同时配备了适当的习题，书后还附有较为详细的习题解答。本书中的例题和习题均以C、PASCAL为语言背景，算法以C语言的格式给出，从而做到了与前期课程的较好融合。本教材的参考教学时数为60学时。

与本教材配套的学习辅导用书《编译方法学习指导与实践》已于2004年正式出版，教材中所选习题有一小部分即来自此书（题号上加有“*”标记）。对于这些习题，本书不再给出答案，请读者自行参考。

由于本人水平有限，书中出现的不足和错误，敬请读者批评指正。

编　者

2007年3月4日

目　录

第1章 编译概述

我们知道，计算机所能执行的语言是0/1代码的机器语言，那么一个高级程序设计语言的程序最终是怎样在计算机上得以执行的呢？这就需要对其进行翻译，从而得到等价的0/1代码指令，而编译就是实现这种翻译的方法之一。本章将主要介绍编译程序与程序设计语言之间的关系，并简单叙述编译的过程及编译程序的结构。

1.1 编译程序

1.1.1 编译与程序设计语言

1. 程序设计语言的层次

编译程序是计算机系统的基本组成部分，是一种和计算机的程序设计语言密切相关的系统软件。

我们知道，计算机的程序设计语言分三个层次——机器语言、汇编语言和高级语言。

机器语言程序是由二进制码(0/1代码)组成的，机器可直接执行，但它难写、难读，还与机器的硬件环境关系密切，程序员很难用它直接编程。

汇编语言程序采用汇编指令，编写、阅读都较机器语言容易得多，但它对机器的硬件环境仍有较大的依赖性。计算机是不能直接执行汇编语言程序的，必须将其翻译成机器语言程序才能执行，完成这一翻译任务的是汇编程序。

高级语言程序较汇编语言程序更进了一大步，它采用高级程序设计语言编程，这种语言接近自然语言，易学、易写、易读，并且彻底摆脱了对具体机器硬件环境的依赖，不但便于描述算法，也便于程序移植，所以是使用最为广泛的程序设计语言。当然，计算机也是不能直接运行高级语言程序的，必须将其翻译成机器语言程序后才能运行。

高级语言程序的翻译方式有两种——解释和编译。所谓编译就是将高级语言程序翻译成另一种语言的等价程序，其中高级语言程序称为源程序，翻译生成的等价程序称为目标程序，完成编译任务的程序称为编译程序。目标程序虽可以是机器语言程序，但更常见的是汇编语言程序。源程序、目标程序和编译程序的关系如图1-1所示。

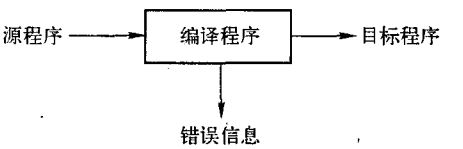

图1-1 源程序、目标程序和编译程序的关系

2. 高级语言程序的处理过程

为了弄清编译程序的作用，有必要回顾一下高级语言的处理过程。

高级语言的处理过程如图1-2所示。

(1) 预处理

一个高级语言程序可以是多模块的，由若干个文件组成，可以含有宏指令，这样的程序应先进行文件的合并和宏展开才能形成完整的源程序，这一工作一般由预处理程序完成。

(2) 编译

形成完整的源程序后，由编译程序对其进行编译。在编译过程中，编译程序若发现语法错误，将报告这些错误，一般会给出错误的个数、错误所在的行、错误的编号及错误信息等。程序员可以根据这些信息重新修改、编辑源程序，然后再次编译。若编译程序未发现错误，即编译成功，生成目标代码程序。

(3) 汇编

编译生成的目标程序可以是机器代码程序，但更常见的是汇编语言程序。如果是汇编语言程序，则还要把它交给汇编程序对其进行汇编，从而生成机器代码程序。

(4) 装配/连接

汇编生成的机器代码程序一般是可再装配的，要用装配/连接程序将其与一些可再装配的目标文件(如库函数等)连接起来，这样才最终生成可执行的绝对机器代码程序。

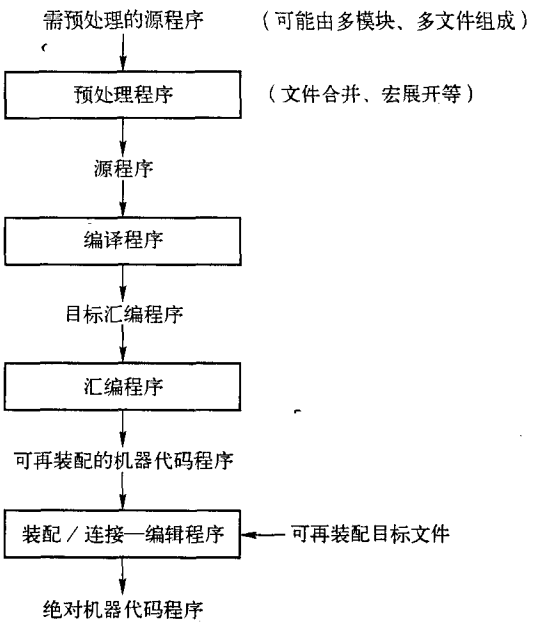

图 1-2 高级语言程序处理过程

由此可见，由于编译程序的存在，使得程序员能够使用高级语言方便地进行编程，而不必考虑与机器有关的繁琐细节，使程序独立于机器。

每一种高级语言都有其相应的编译程序，所以一个计算机系统一般含有多个不同程序设计语言的编译程序。

3. 编译与解释

当然，高级语言的翻译也有另一种方式，即解释。这种方式与编译的不同之处在于它对高级语言的翻译是一句一句进行的，即读入高级语言程序的一条语句就对其进行翻译并执行，然后再读入下一条语句，如此边翻译边执行，直到程序结束。在这一"翻译——执行"过程中是不生成等价的目标代码程序的。所以，如果用解释方式多次执行同一个高级语言程序，则每次执行都必须对这个程序进行一次从头至尾的翻译。

编译和解释这两种翻译方式各有其优点，编译方式只需翻译一次，且目标程序的执行速度快，而解释方式则便于程序的调试。

1.1.2 编译程序的构造

要编译一种新的高级程序设计语言，或要为一种源语言编译出一种新的目标语言，就要构造一个新的编译程序。编译程序的构造与三个方面有关——源语言、目标语言和编译方法。

1. 源语言

源语言是翻译的对象，要翻译它就必须准确理解它的结构、含义和用途，并将这些内容准确描述出来，这是构造编译程序的出发点。

2. 目标语言

目标语言是翻译的结果，翻译成什么样的目标语言决定了在翻译过程中应考虑些什么问

题。如目标语言是汇编语言，则要弄清汇编语言程序的结构、指令系统等；若目标语言是机器语言，则要考虑的问题就多了，要弄清机器的硬件系统结构、操作系统功能、存储分配方式、外设管理方式、文件管理方法等一系列问题。

3. 编译方法

编译方法就是将一种语言翻译成另一种语言的具体方法，这样的方法有很多，本书就是要介绍这些方法。对具体某一种语言而言，究竟采用哪种编译方法，应根据多种因素进行选择，如源语言的特性、目标语言的特性、对编译程序性能的要求（指编译程序本身的质量和标准，如编译的可靠性、速度、目标程序的运行速度和空间使用效率、程序的可移植性和可维护性等）、程序员的编程爱好和习惯等。

1.2 编译过程概述

编译程序的工作过程是比较复杂的，一般划分成若干阶段进行。每个阶段以源程序的一种表示形式作为输入，对其进行一定的分析和翻译，并将其转换成另一种表示形式输出，以作为下一阶段的输入，完成进一步的翻译工作。编译过程各阶段的典型划分如图 1-3 所示。本节以下内容将对编译各阶段的工作进行简单介绍。

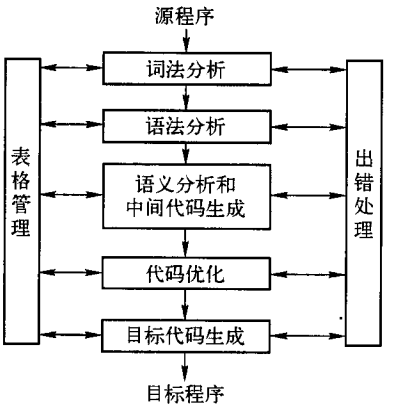

图 1-3 编译阶段划分

1.2.1 词法分析

词法分析阶段完成的主要任务是从左到右扫描源程序，逐一读入构成源程序的字符流，识别出其中的一个个单词，识别出的单词称单词符号，也简称符号。

单词是高级语言程序中有实际意义的最小语法单位，单词由字符构成。每种高级语言都规定了一个字符集，单词可以是字符集中的单个字符，如 C 语言中的“+”、“*”等，也可以由字符集中的字符按一定的规则组合而成，如 C 语言中的自增符号“++”、保留字“char”、标识符“proc”等，而程序就是由这样的单词构成的。词法分析就是要把组成源程序的单词按顺序一一识别出来。

单词的构成规则称为词法规则或构词法，单词识别的依据就是词法规则。

单词识别出来后编译程序要将其转换成一种格式统一的、比较便于机器处理的内码形式，如二元式，其中指出了单词的类别和自身值。

【例 1-1】C 语言的语句

```
z=x+a%3*(int)(x+y)%2/7;
```

识别出的已转换成二元式的单词序列为

(1)（标识符，z）

(2)（等号，=）

(3)（标识符，x）

(4)（加号，+）

(5)(标识符,a)
(6)(取余号,%)
(7)(整数,3)
(8)(乘号,*)
(9)(左括号,()
(10)(保留字,int)
(11)(右括号,))
(12)(左括号,()
(13)(标识符,x)
(14)(加号,+)
(15)(标识符,y)
(16)(右括号,))
(17)(取余号,%)
(18)(整数,2)
(19)(除号,/)
(20)(整数,7)
(21)(分号,;)

词法分析阶段识别出的单词将作为编译的第二阶段语法分析的输入,由语法分析程序作进一步的分析。

1.2.2 语法分析

语法分析阶段完成的任务是"组词成句",即根据单词分析阶段识别出的单词分析出组成源程序的各类语法单位,并指出其中的语法错误。

源程序的单词构成了各种语法单位,如表达式、语句、……乃至整个程序。每种语言都规定了各种语法单位的构成规则,这种规则称为语法规则。语法分析就是按语法规则来分析源程序(单词序列)是否构成正确的语法单位。一个语言的词法规则和语法规则定义了一个程序的形式结构。

例如,对于符号串:

```
z=x+a%3*y
```

在C语言中是一个赋值语句,语法分析的任务就是要分析出该符号串首先是一个变量(z),然后是一个赋值号(=),接下来的"x+a%3*y"是一个语法结构正确的算术表达式,同时分析出这样的一个符号串构成了C语言的一个正确的语法单位——赋值语句。

语法单位的表示一般采用语法树的形式,如上面的赋值语句表示成的语法树如图1-4所示。

语法分析的方法分为自顶向下分析法和自底向上分析法两大类。本书第4、第5两章将介绍这两类语法分析方法。

1.2.3 语义分析和中间代码生成

这一阶段完成的任务是:在语法分析识别出正确的语法单位后,进一步对其语义进行分

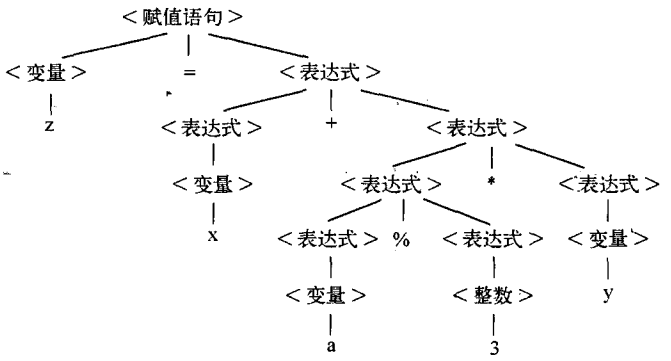

图 1-4　语法树

析,分析出该语法单位具体的动作意义,即要干什么,进而进行初步的翻译,生成与源程序等价的中间代码程序。语义由语义规则给出,语义规则定义了一个程序所表示的实际意义。

将源程序直接翻译成目标代码是一项复杂的工作。为了降低翻译的复杂程度,编译程序采用了一种中间代码,也称中间语言。在编译时,翻译工作将分两步进行——先把源程序翻译成等价的中间代码程序,再把中间代码程序翻译成目标代码程序。这样做还有另一个好处,就是便于进行代码优化。

显然,中间代码的指令应结构简单,含义明确,易于实现源程序——中间代码——目标代码三者之间的转换。中间代码的形式有多种,常用的有逆波兰式、三元式、四元式等。例如,四元式代码的形式为

(运算符,运算对象 1,运算对象 2,结果)

【例 1-2】C 语言的语句"z = x + a%3 * y"经前面几阶段分析后,可以翻译成以下四元式序列:

(1) (%　a　3　t_1)
(2) (*　t_1　y　t_2)
(3) (+　x　t_2　t_3)
(4) (=　t_3　_　z)

另外,中间代码的使用还有利于进行代码的优化工作。

1.2.4　代码优化

这一阶段是对中间代码进行改造,作等价的加工变换,以便生成更为有效、更节省时间和空间的目标代码。

【例 1-3】例 1-2 中语句"z = x + a%3 * y"的四元式序列可改造成

(1) (%　a　3　t_1)
(2) (*　t_1　y　t_2)
(3) (+　x　t_2　z)

代码优化的技术多种多样,如删除公共子表达式、强度削弱、代码外提、合并已知量等。

需要指出的是,这一阶段并不是编译程序所必需的,不优化而直接进入下一阶段(生成目标代码)也是可以的,但有了这一阶段可以使生成的目标代码更为有效。

1.2.5 目标代码生成

这是编译的最后一个阶段，这一阶段的任务是将中间代码程序变换成目标代码程序，从而完成最终的翻译。

目标代码可以是特定机器上的绝对指令代码，也可以是可重定位的指令代码或汇编指令代码等低级语言代码。

这一阶段任务的实现与硬件系统的结构、目标指令的选择、变量存储空间的分配、寄存器、后缓寄存器的调度等均有关系。

1.2.6 表格管理和出错处理

编译过程划分为以上五个阶段，为了完成这些工作，编译程序还要建立并使用一些表格。这些表格记录各种信息，而这些信息是在编译的各个阶段被陆续发现并登录进去的，也是在编译各阶段的分析和翻译中都需要用到的，因而表格管理涉及编译的每个阶段。

另外，若编译过程中发现源程序有错误，就应进行出错处理。出错处理可以是报错、纠错等。在编译的各个阶段都有可能发现源程序中的错误，所以出错处理也遍布编译的每个阶段。

以上介绍的就是编译过程的典型处理模式，而事实上，并非所有的编译过程都有这些阶段，有时可能不生成中间代码，也有时可能不进行代码优化。

对于编译的这五个阶段，我们也常常将它们划分成两大部分。前三个阶段为一部分，称为分析部分；后两个阶段为另一部分，称综合部分。

1.3 编译程序的结构

1.3.1 编译程序的结构

编译各个阶段的工作用相应的程序模块完成，这些模块组合到一起就构成了完整的编译程序。编译程序的典型结构如图 1-5 所示。

1.3.2 编译阶段的组合

前面所介绍的编译阶段的划分是编译程序的逻辑组织，实际组织程序时，一般要将各阶段的工作进行一定的组合。

1. “前后端”组合方式

将那些主要依赖于源语言，而与目标机器无关的工作组合到一起，构成前端程序，这些工作可以包括词法分析、语法分析、语义分析与中间代码生成、某些与目标机器无关的代码优化，以及此间的表格管理和出错处理等。

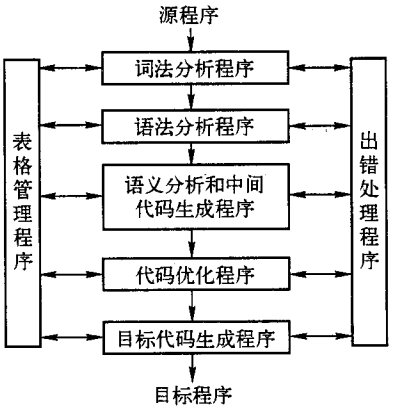

图 1-5 编译程序结构

将与源语言无关、只与中间代码有关、主要依赖于目标机器的工作组合到一起，构成后端程序。这些工作可以包括某些与目标机器有关的代码优化、目标代码生成，以及相关的表格管理和出错处理等。

一种高级语言若要在不同的机器上运行，由于目标机器不同，需要构造不同的编译程序；反之，不同的高级语言在同一机器上运行，也需要构造不同的编译程序。因此需要构造大量的编译程序，而构造编译程序的工作量是很大的。采用“前后端”组合的方式能较好地解决这一问题。

若一种语言要在不同的机器上运行，我们可构造一个前端程序，然后再为每种机器构造相应的后端程序。这样，一个前端程序加上不同机器该语言的后端程序就产生了该语言的多个编译程序，如图 1-6a 所示。

反之，若同一机器上要运行不同高级语言，可以采用同样的中间代码构造各自的前端程序，然后再构造一个共同的后端程序。这样，不同语言的前端程序加上同一个后端程序就产生了该机器不同语言的编译程序，如图 1-6b 所示。

例如在 Java 环境中，定义了一种虚拟机代码 Bytecode，只要实际操作平台实现了 Bytecode 的解释，该平台即可执行各种 Java 程序。

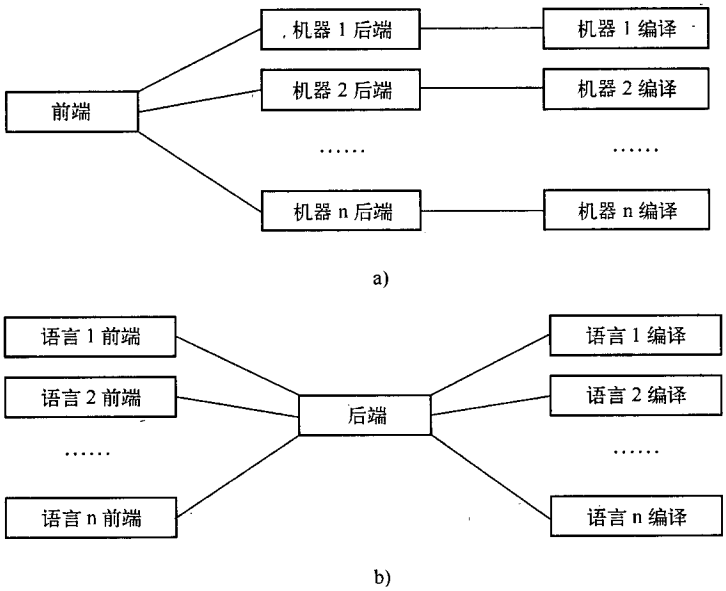

图 1-6

a) 一“前端”+多“后端”　b) 多“前端”+一“后端”

2. 按“遍”组合方式

“遍”是编译中的一个概念，对源程序或等价的中间语言程序从头到尾扫描，完成规定的任务，并生成新的中间结果或目标程序，称一“遍”。在一“遍”中可以完成编译的一个或多个阶段的任务，例如仅完成词法分析，或完成词法分析和语法分析等，在一“遍”中甚至还可以完成整个编译。所以编译程序可以是一“遍”的，也可以是多“遍”的。编译程序若是多“遍”的，则上一“遍”的输出就是下一“遍”的输入。

如果按“遍”组织，则应根据源语言的特征和机器的特征来决定编译程序究竟划分成几“遍”。例如，某源语言规定变量的使用可在变量的说明之前，则编译时第一遍肯定不能为含有未说明变量的表达式生成代码，故这种语言的编译程序至少应划分为两遍。另外，编译程序的工作环境(机器)也会影响遍的划分。