国外著名高等院校
信息科学与技术优秀教材

WILEY

现代编译程序设计

Modern Compiler Design

Dick Grune
Henri E. Bal
Ceriel J. H. Jacobs
Koen G. Langendoen
著

冯博琴　傅向华　等　译

中文版

人民邮电出版社
POSTS & TELECOM PRESS

图书在版编目（CIP）数据

现代编译程序设计 /（荷）格伦（Grune, D.）等著；冯博琴等译. —北京：人民邮电出版社，2003.9
国外著名高等院校信息科学与技术优秀教材
ISBN 7-115-11149-9

Ⅰ. 现… Ⅱ. ①格… ②冯… 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（2003）第 065535 号

国外著名高等院校信息科学与技术优秀教材

现代编译程序设计

◆ 著 Dick Grune Henri E.Bal Ceriel J.H.Jacobs
Koen G.Langendoen
译 冯博琴 傅向华 等
责任编辑 陈 昇

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
读者热线 010-67132705
北京汉魂图文设计有限公司制作
北京隆昌伟业印刷有限公司印刷
新华书店总店北京发行所经销

◆ 开本：787×1092 1/16
印张：35.5
字数：863 千字 2003 年 9 月第 1 版
印数：1-5 000 册 2003 年 9 月北京第 1 次印刷

著作权合同登记 图字：01－2002－4339 号
ISBN 7-115-11149-9/TP・3362
定价：58.00 元

本书如有印装质量问题，请与本社联系 电话：(010)67129223

内容提要

本书全面地介绍了现代编译技术，结构上分为通用编译技术和高级编译技术两大部分。第一部分介绍通用的编译程序实现技术，包括词法和语法分析、上下文处理、代码生成以及存储器管理的一般方法。第二部分介绍特定范型语言的高级编译技术，包括命令式语言、面向对象语言、逻辑式语言、函数式语言及并行/分布式语言的上下文处理和代码生成等内容。本书注重编译程序的具体实现和优化技术，实例丰富，具有很强的可读性和实用性。

本书可作为高校计算机专业本科和研究生编译程序设计课程的教科书，也可供从事计算机软件开发的人员参考。

出版说明

2001年，教育部印发了《关于“十五”期间普通高等教育教材建设与改革的意见》。该文件明确指出，“九五”期间原国家教委在“抓好重点教材，全面提高质量”方针指导下，调动了各方面的积极性，产生了一大批具有改革特色的新教材。然而随着科学技术的飞速发展，目前高校教材建设工作仍滞后于教学改革的实践，一些教材内容陈旧，不能满足按新的专业目录修订的教学计划和课程设置的需要。为此该文件明确强调，要加强国外教材的引进工作。当前，引进的重点是信息科学与技术和生物科学与技术两大学科的教材。要根据专业（课程）建设的需要，通过深入调查、专家论证，引进国外优秀教材。要注意引进教材的系统配套，加强对引进教材的宣传，促进引进教材的使用和推广。

邓小平同志早在1977年就明确指出：“要引进外国教材，吸收外国教材中有益的东西。”随着我国加入WTO，信息产业的国际竞争将日趋激烈，我们必须尽快培养出大批具有国际竞争能力的高水平信息技术人才。教材是一个很关键的问题，国外的一些优秀教材不但内容新，而且还提供了很多新的研究方法和思考方式。引进国外原版教材，可以促进我国教学水平的提高，提高学生的英语水平和学习能力，保证我们培养出的学生具有国际水准。

为了贯彻中央“科教兴国”的方针，配合国内高等教育教材建设的需要，人民邮电出版社约请有关专家反复论证，与国外知名的教材出版公司合作，陆续引进一些信息科学与技术优秀教材。第一批教材针对计算机专业的主干核心课程，是国外著名高等院校所采用的教材，教材的作者都是在相关领域享有盛名的专家教授。这些教材内容新，反映了计算机科学技术的最新发展，对全面提高我国信息科学与技术的教学水平必将起到巨大的推动作用。

出版国外著名高等院校信息科学与技术优秀教材的工作将是一个长期的、坚持不懈的过程，我社网站（www.ptpress.com.cn）上介绍了我们陆续推出的图书的详细情况，敬请关注。希望广大教师和学生将使用中的意见和建议及时反馈给我们，我们将根据您的反馈不断改进我们的工作，推出更多更好的引进版信息科学与技术教材。

人民邮电出版社

译者序

编译程序是计算机系统软件中非常成功和最具活力的一个分支。20世纪80年代以来，计算机程序设计语言领域有了突飞猛进的发展，程序语言已经不仅有传统的命令式范型，而且包括了面向对象、函数式、逻辑式和并行/分布式范型。新的程序语言的诞生，推动了编译程序设计方法的研究，并使新的编译技术不断引入。新的程序语言要求有更加强大的运行时系统，所幸的是，计算机硬件发展水平——CPU的速度和存储容量为编译技术提供了强有力的支持。

虽然每一个计算机专业的学生未必都有机会亲自参与编译器的设计，也没必要要求大家都具备编写编译器的能力，但我仍然认为编译技术包含的软件设计思想和经典算法对于一个优秀的程序设计人员来说是必备的背景知识，因此国内绝大部分的计算机专业仍然把编译原理作为必修课。

本书是迄今介绍编译程序设计方法最新、最全面的著作之一，可以毫不夸张地说，该书是真正意义上的“现代编译程序设计”。加上之前我国已经引进的两本编译技术方面的著作（*Compilers: Principles, Techniques, and Tools*[1] 和 *Programming Languages: Design and Implementation Fourth Edition*），这三本书对于改进我们的教学是十分有益的，因为现在国内大多数有关编译程序设计的教材都只集中讲述C和Pascal那样的命令式语言的编译方法，而这些书已充分注意到了进入网络时代之后程序设计语言的巨大变化，因此书中不少内容是国内教材中十分鲜见的。我认为这是本书中译本的最主要价值。

本书分为两部分：第一部分是第1章至第5章，介绍传统的编译原理和技术，包括词法分析、语法分析、上下文处理、中间代码生成和优化、存储器管理等；第二部分是第6章至第9章，介绍了面向对象、函数式、逻辑式、并行/分布式语言中使用的特殊的编译技术。该书作者是程序设计语言实现方面的专家，且富有写作经验。本书注重实现和优化技术，增加了实用性同时提供了大量的实例说明，大大降低了学习、理解的难度。

本书可作为计算机专业本科生和研究生编译原理方面的教材或参考书，也可作为软件人员从事软件开发的参考资料。

本书涉及的内容较新，有些术语的译法还未见统一。同时也限于译者水平，对某些内容的理解可能有偏差。在此预先向读者致歉，并望广

大师生不吝指正。

本书由冯博琴组织翻译并统稿。参加翻译的人员有傅向华、韩冰、王自强、何明、张志刚、曹博、贺晓红、范宁、王静、潘晓春。薛亮、汪齐名、胡英萍、夏虹、傅蓉参加了校对工作，傅向华做了大量的组织工作。另外还要感谢吕军老师在翻译过程中所提供的帮助。

[1] 本书英文影印版《编译原理 技术与工具》已由人民邮电出版社出版，书号 7-115-09916。

冯博琴

2003 年 7 月 20 日

前　言

20 世纪 80 年代至 90 年代，当整个世界通过新闻报纸的头版目睹 PC 和 Internet 的兴起时，编译程序设计方法也在悄悄地发展，可以在技术刊物——更重要的是在当今软件使用的编译程序——上看出这种发展。这种发展部分得益于新的程序设计方法，部分得益于对代码生成技术的更好理解，部分得益于更高速的大容量存储器机器的使用。

现在，程序语言领域除包括传统的命令式范型之外，还包括面向对象、函数式、逻辑式和并行/分布式范型，这些都引入了新的编译技术，而且常常比传统的命令式语言需要更好的运行时系统。BURS 技术包含有强大的代码生成技术，因而能极好地处理当今机器的复杂指令集。现代机器的速度和存储器容量允许编译技术和程序语言拥有以前不可想像的特征。现代编译程序设计方法就用于迎接这些挑战。

目标读者

我们面向的读者是即将毕业的学生，他们至少曾使用过编译程序并且对编译的概念有一些了解。当这些学生毕业后，要使用现代技术，他们就必须熟悉每一种现代范型的语言处理器。尽管很多大学的课程要求滞后于这方面的考虑，但即将进入人才市场的毕业生不能忽视这些发展。

经验表明，在编译程序构造方面，传统所教的大部分技术只是更基本技术的特例。通常这些特殊的技术仅仅适用于命令式语言。而更基本的技术有更广泛的应用。例如，在严格的后进先出语言中，栈可以作为活动记录的优化表示。因此，本书：

- 集中讲述广泛应用的原理和技术，谨慎地区分本质的东西（=对学生最有用的）和非本质的东西（=仅仅在特殊情况下对学生有用的）；
- 提供直接的实现细节和优化；
- 给出进一步深入学习的阅读材料。

学完本书，学生应该达到如下目标：

- 获得现代编译程序设计方法和构造的完整概念，熟悉它们的实际应用；
- 通过很少的培训时间就能够开始参与每一种现代范型的语言处理器的构造；
- 能够阅读相关文献。

前两个方面为读者提供了坚实的基础，第三个则提供了成长的潜力。

本书的结构

本书从概念上分为两大部分。第一部分从第 1 章到第 5 章，着眼于通

常的程序处理技术，其中有一章存储器管理的内容，涉及编译程序和生成代码的存储器管理。第二部分从第 6 章到第 9 章，覆盖了不同程序设计范型所需要的特殊技术。本书各部分的相互关系可由下表来描述。

	命令式和面向对象程序（第 6 章）	函数式程序（第 7 章）	逻辑式程序（第 8 章）	并行/分布式程序（第 9 章）
怎样做：				
分析（第 2 章）	——	——	——	——
上下文处理（第 3 章）	标识符识别	多态性检查	静态规则匹配	Linda 静态分析
合成（第 4 章）	While 语句代码	列表内涵的代码	结构合一	编组
运行时系统（无章节）	栈	归约机	Warren 抽象机	复制

最左列表示了编译程序构造的 4 个阶段：分析、上下文处理、合成和运行时系统。这一列列出的章节覆盖了相关软件的手工和自动生成，重点是自动生成。其余各列表示了本书的 4 种范型；对每一种范型，表中显示了在编译程序的每个阶段主要论述的一个例子。这些章节（第 6 章到第 9 章）倾向于仅仅包括手工技术，因为所有的自动技术已经在第 2 章到第 4 章中讨论了。

一般来讲，理想的情况下这个表是四方的，所有的单元格都填满——也就是说该表应该是很规则的，但我们看到右上角几个单元格为空而左下角没有“运行时系统”的章节。右上几格应当覆盖这样一些内容，如逻辑式语言文本分析的特殊主题，但是目前的文本分析技术已经足够强大和足够灵活——各种语言非常相似——以至于文本分析技术已经能处理所有的语言范型：由于没有什么问题需要解决，我们对此不作讨论。在左下角空缺的章节应该讨论产生运行时系统的手工和自动技术。不幸的是很少有或者根本就没有关于这方面的文章：运行时系统仍然需由程序员在直觉基础上由手工来做；对这种情况，由于缺乏一般的解决方案，也不进行讨论。

第 1 章通过详细检查一个简单传统的模块化编译/解释程序向读者介绍编译程序设计的方法。在介绍完关于编译程序构造的简单历史和形式文法后，接着介绍几个关于编译程序构造的高级方面的内容。

第 2 章讨论编译的分析阶段：即将程序文本转化为抽象语法树。讨论的是词法分析技术、记号的词法识别以及语法分析等。

第 3 章涵盖了编译的第二个阶段：上下文处理。讨论了几种上下文处理的方法：使用属性文法的自动上下文处理，使用 L-属性文法和 S-属性文法的手工上下文处理，使用符号解释和数据流分析的半自动上下文处理。

第 4 章讲述编译的合成阶段，包含解释和代码生成。代码生成部分主要是关于机器代码生成部分；特定范型构造所需的中间代码在第 6 章到第 9 章讨论。

第 5 章关注的是存储器管理技术，这些技术既可以用于编译程序又可用于编译程序生成的程序中。

第 6 章到第 9 章讨论的是在编译不同范型时遇到的特定问题，这些范型包括命令式、面向对象、函数式、逻辑式和并行/分布式。命令式的编译程序和面向对象的编译程序很相似，一起放在第 6 章中讨论。

附录 A 讨论一种可能的试验性的面向对象编译程序构造方法，在这种方法里试图利用面向对象的概念来简化编译程序的设计。

本书中某些主题采用一种非传统的方式进行讨论，并适当地采用了一些被证明是正确的用词。

词法分析与自底向上语法分析都基于加点项，而不是基于 Thompson 的 NFA 结构。在自底向上模式匹配、统一词法分析、LR 语法分析和自底向上代码生成中加点项是一个很重要的工具。传统的词法分析算法只是项操作的低级实现。我们认为，词法分析与语法分析的不同处理是历史上人为造成的，在现代软件中，词法和语法层次之间的差异趋于消失。

尽管属性文法对编译程序设计的影响很有限，但人们仍然对属性文法给予了相当多的注意。它们仍然是自动上下文处理的唯一途径，我们希望当前的讨论能够有助于降低它们应用的门槛。

与其他编译程序设计方法的书相比，在本书中，我们对作为重要数据的函数作了更深入的讨论。自从在 Algol 60 中有一个很好的开始后，函数作为可操作的数据在诸如 C、Pascal 和 Ada（尽管 Ada 95 恢复了一些）等语言中的地位已经下降了很多。然而，一些现代概念的实现，如函数式和逻辑式语言、迭代器和继续（continuation），都要求将函数作为一个一般的数据进行操作。实现的基本内容在命令式和面向对象语言的章节涉及；更专门的内容部分在讲述其他范型的章节给出。

在 1.11 节中，为便于理解，我们对在本书中使用的概要代码进行了说明。

附加的材料，包括习题的部分答案、本书中所有的图和代码，都可以通过 John Wiley 的 Web 页面访问到。

作为课程教材

本书比我们大学规定的每次两小时总共 13 次的编译设计方法课程包含了更多的内容，因此必须有所取舍。根据听课者的知识完备程度，例如，传统的介绍性的课程应该包括下面章节：

第 1 章；

第 2 章开始到 2.1.7 节；2.1.10 节；2.1.11 节；2.2 节到 2.2.4.5 节；2.2.5 节到 2.2.5.7 节；

第 3 章开始到 3.1.2 节；3.1.7 节到 3.1.10 节；3.2 节到 3.2.2.2 节；3.2.3 节；

第 4 章开始到 4.1 节；4.2 节到 4.2.4.3 节；4.2.6 节到 4.2.6.4 节；4.2.11 节；

第 5 章开始到 5.1.1.1 节；5.2 节到 5.2.4 节；

第 6 章开始到 6.2.3.2 节；6.2.4 节到 6.2.10 节；6.4 节到 6.4.2.3 节。

更高级一点的课程应该包括第 1 章到第 6 章的所有的内容，不含 3.1 节。也可以包括第 7 章到第 9 章中的一章，或许还可以加上附录 A。

高级课程应当跳过大多数介绍性内容而把注意力集中于介绍性课程所忽略的部分：3.1 节和第 5 章到第 9 章所有的内容，并加上附录 A。

致谢

我们对以下的人满怀感激：Mirjam Bakker、Raoul Bhoedjang、Wilfred Dittmer、Thomer M.Gil、Ben N.Hasni、Bert Huijben、Jaco A. Imthorn、John Romein、Tim Ruhl 和许多我们不知道其名字的审校者，他们自愿花费时间和精力阅读本书的草稿并给出许多有用、有时是非常详细的建议。并感谢 Ronald Veldema 提供 Pentium 处理器的代码段。

感激 Simon Plumtree、Gaynor Redvers-Mutton、Dawn Booth 和 John Wiley & Sons 出版公司的 Jane Kerr，他们对本书的撰写过程给予了帮助和鼓励。Lambert Meertens 热心地提供了关于老的 ABC 编译程序的信息，Ralph Griswold 提供了 Icon 编译程序。

感谢 Vrije 大学的 Faculteit Wiskunde en Informatica（现在是 Faculteit der Exacte Wetenschappen 的一部分）提供的支持，在本书编写过程中使用了他们的设备。

于阿姆斯特丹　2000 年 5 月

目 录

第1章 导论

从最一般的形式上看，**编译程序**是一种程序，它接受某种语言的程序文本作为输入，而后输出另外一种语言的程序文本，而保持文本的原意。这个过程称为**翻译**，正如该文本是自然语言那样。几乎所有的编译程序仅仅能将一种输入语言（即**源语言**）翻译成另一种输出语言（即**目标语言**）。源语言和目标语言差距很大是正常的，源语言可能是C，而目标语言可能是适应Pentium系列处理器的机器代码。编写编译程序本身的语言称为**实现语言**。

需要这样一种翻译的主要原因是拥有能够运行翻译过的程序的硬件设备，或者更精确地说：让硬件执行由程序语义所描述的动作。毕竟，硬件设备是唯一真正拥有计算能力的资源。运行一个翻译过的程序经常包括以一定格式供给输入数据，而后得到其他格式的输出数据。输入数据可以来自不同的源，例如文件、键盘和网络数据包。同样，输出数据也可以输出到不同的地方，例如文件、显示器和打印机。

要得到翻译过的程序，我们需要运行编译程序，而编译程序是另外一个程序，它的输入数据是一定格式的源程序文本文件，它的输出数据是可执行代码格式的文件。这里有一个巧妙之处，包含可执行代码的文件通常被默认转化成为可运行程序，而在少数操作系统中，还需要一些动作，例如设置“执行”属性。

要得到编译程序，我们需要运行另外一个编译程序，它的输入包含编译程序源文件并且它将为源文件产生可执行代码，正如它为其他任何源程序文本所做的那样。编译和运行一个编译程序的过程在图1.1中描述。编译程序能够编译编译程序，这听起来似乎很容易混淆。当源语言也是实现语言并且将被编译的源文本事实上是编译程序自身的一个新版本时，这个过程称为**自举**（bootstrapping）。术语“bootstrapping”来源于我们所熟知的德国民间故事《吹牛大王历险记》，不过故事的主角von Münchhausen男爵（1720～1797）是通过拉扯自己的发辫而不是靴襻（bootstraps）而脱离沼泽的（作者：匿名，1840年）。

编译和文件转换并无根本的不同，但在某种程度上二者确实有所差别。我们应该清楚，编译的输入有一个性质称作语义——即输入的意思——它（语义）必须在处理过程中被保留，而在传统的文件转换程序中语义并不是轻易可以辨别的，例如把EBCDIC转换到ASCII。另一方面，GIF到JPEG的转化程序必须保留图片的视觉印象，视觉印象在某种意义上也可以称为语义。总而言之，一个编译程序就是一个大的文件转换程序。

编译程序能够神奇地工作是因为两个因素：

- 输入是某种语言从而具有一种结构，该结构在语言的参考手册中给出了描述；
- 输入的语义由这种结构来描述，并且附着在这种结构之上。

这些因素使编译程序能够“理解”程序并且在**语义表示**中收集语义。

对于目标语言而言，也存在同样的两个因素。这使得编译程序可以把搜集的语义以目标语言的形式再现出来。如何在细节上实现这个过程是本书的主题。

编译程序中进行源语言文本分析的那部分称为**前端**，而进行目标语言合成的部分称为**后端**，见图1.2。如果编译程序设计得非常清晰，前端完全不会感知目标语言的存在而后端也不会感知源语言的存