IBM PC 混合语言编程技术

刘乃琦 编著

電子工業出版社

IBM PC
混合语言编程技术

刘乃琦 编著

电子工业出版社

内容简介

本书是一本实用型技术书籍，介绍了混合语言程序设计的概念、技术和方法；讨论了目前常用的多种高级语言之间的相互调用、参数传递、数据共享与接口技术；还重点讲述了各种高级语言与汇编语言之间的调用和接口，以及内嵌式汇编语言编程。本书按实用型章节分类组成适合教学和阅读的模块结构，各章均有若干实例并附有习题与思考题。

本书适用于广大计算机应用人员和技术人员，也可作为高等院校教师、学生以及有关科技人员的计算机应用的教材或参考书。

IBM PC
混合语言编程技术
刘乃琦 编著
责任编辑 王昌铭
*
电子工业出版社出版（北京市万寿路）
电子工业出版社发行 各地新华书店经售
电子工业出版社计算机排版室排版
机电部科技情报所印刷
*
开本：787×1092 毫米 1/16 印张：11.5 字数：298 千字
1990 年 9 月第 1 版 1991 年 5 月第 2 次印刷
印数：8000-18100 册 定价：4.50 元
ISBN7-5053-1056-9/TP・173

前　言

你想知道什么是混合语言程序设计吗？你想在各种语言之间搭起一座桥梁让它们相互交往，彼此调用对方的功能程序模块吗？你想让高级语言编制的程序通过汇编语言程序去与物理硬件直接打交道吗？你想利用现有的程序、库函数等迅速组成（或者经过少量改进而构成）自己的应用程序吗？混合语言编程能给你提供这方面的技术。一旦你了解了它，熟悉了程序接口的建立和程序编制过程，就会熟能生巧，取得极大的效益和成功。

计算机软件、硬件技术在相互支持、相辅相成的过程中不断地发展，计算机语言的进展也倍使世人注目。目前，在任何一种计算机上都配有若干种语言，它们有各自的特点和适用范围，也不可避免地带有历史的影响和应用的局限。它们通过自身的不断改进、发展，保持了强大的生命力。对于广大计算机技术人员和应用人员来说，面临的问题是如何利用各种语言的优势，在实际工作和应用领域中编写出效率高，费时少，成本低的软件系统。把各种语言的长处组合在一起的思想很早就提出来了，但混合语言程序设计仍受到诸如语言结构，数据与代码生成结构，参数调用和传递结构等不同的限制。随着语言的进展，支持混合编程的功能在新近的语言版本中不断涌现，语言之间功能的相互吸收和弥补也使混合编程技术得以较好地实现。这种技术在开始使用时，仅局限在一个语言系列之内，继而被扩展至同一个运行环境之中。混合语言程序设计技术将扩大已有语言的应用范围，缩短软件开发时间，有效地利用已有开发结果，帮助我们建立功能强大的程序库。用户可以用任何一种源语言来生成程序，并实现程序间的数据共享与子程序共享。

本书是一本实用性的技术书籍，重点在于讨论混合语言编程所必需的概念、语法格式、语言约定和接口方式，使读者阅读本书后，通过众多的实例掌握多种语言之间的相互调用、参数传递、数据共享的方法，并以此为基础编写自己的混合程序。阅读此书要求读者对计算机语言有一定的基础，对编程环境（如编辑、编译、连接过程等）有基本的了解，对目前主流微型机（如IBM PC 系列）有一定的知识。所以，本书不准备就各种语言再讲述其基本的编程方法。

混合语言程序设计并不是十分困难的，一旦读者了解了基本的规则和约定，就可以灵活地进行混合语言编程和程序组合了。本书按实用型模块和层次方式组织，读者很容易找到与自己最有关的和最需要的章节。如果要了解混合语言程序设计的概念、进展以及混合编程入门，可以从第一章开始阅读。如果读者已经熟悉高级语言编程方法，而且主程序多用高级语言编写，可以根据需要选读第三至第七章；这些章节分别详细介绍了各种语言的混合编程技术。如果要专门解决汇编语言的调用问题，可以阅读第七章。各章节中所举的实例只是一些极简单的、为了说明接口方式的程序，并且采用的多是简单参数类型。如果要进一步探讨各种类型参数和数据的传递与处理，读者可以深入阅读第八章。本书内的程序，以及混合编程的有关文件、说明均汇集在一张 5.25 英寸软盘上，可与本书配套使用；需要者请与编者或出版者联系。

在本书成书过程中，得到了电子科技大学计算机系刘心松教授等的热情关注，得到了我的同事和朋友们的热情鼓励；计算机系八五级学生钟亚琳同志在程序验证、调试等方面作了许多可称颂的工作，邹玲声老师也给予了极大的支持和帮助，最后郑颖平副研究员对全稿进行了阅

改。在此表示深切的感谢，他(她)们一丝不苟的治学态度将促使我不断前进。

由于成书仓促及本人水平所限，书中若有不妥之处敬请读者指正并欢迎来信赐教。

编者

1990.1 于电子科技大学计算机系

四川·成都

目录

第一章　概论

目前，在计算机（无论大型机还是微型机）中都配置了各种计算机语言。各种语言都有自己的运行环境，有自己的特点和优势，有自己的应用领域和针对性。随着应用要求与技术的发展，各种语言通过彼此交往，互通有无和取长补短，在竞争中不断发展，不断地引入新的功能，力图充分地利用系统和硬件技术所给予的支持。

决定计算机软件质量的好坏的一个基本因素是所使用的编程语言。如何充分发挥各种计算机语言的特点，如何对某一种语言扬其长避其短，如何在各种语言之间进行彼此交往、调用、参数传递、数据共享，分工合作，以共同完成一个任务，形成一个程序整体和软件整体，这也是当今计算机（尤其是微型计算机）应用发展中形成的又一个新的研究课题，受到了众多的使用者和编程者的关心。混合语言程序设计就是一种实用的编程技术和程序接口技术，在这个技术基础上，使各种语言的功能得以充分的利用；熟练地掌握这种技术，会让你取得意想不到的成功。

1.1　什么是混合语言程序设计

混合语言程序设计（也叫混合语言编程，Mixed－language Programming）是近来推出和发展的一种新的应用型的编程技术，也是一种功能很强的有效的程序组合过程，它是软件接口（Softwave Interface）中有关程序接口技术的一个重要的研究和应用领域，不论对应用程序设计还是系统程序设计都提供了一种有效的手段。什么叫混合语言程序设计呢？即是采用两种或者两种以上的编程语言组合编程，彼此相互调用，进行参数传递，共享数据结构及数据信息，从而形成一种程序实体的过程。针对混合语言编程的技术实施，有人直接把它称为程序接口（program interface），也有人称这种技术为组合程序设计（combined programming）。

为了使读者了解混合编程技术的概念，应用及进展，让我们看看它是如何应运而生的吧。

1.2　当前计算机语言和语言编程的发展

从早期的机器语言到汇编语言，再进展到各种高级语言，计算机语言无时无刻都在向前发展，不断地进行自身的完善。而每一次发展总是针对已有语言的不足，在语言结构本身，在功能上，在使用的灵活性、方便性、易读性等各个方面进行改进，并且不断地有一些全新的语言出现以适应实际应用的需要。

我们知道，机器语言由一系列确定的 0－1 序列组成，可读性差，难以理解记忆，一方面编程困难，另一方面修改、调试和移植程序也很困难。继而出现的汇编语言，虽然它采用了助记符和伪指令的形式代替了那些令人眼花缭乱的 0－1 代码，然而，它仍然较多地涉及到机器的内部结构，极大地依赖于系统的硬件（诸如寄存器、存储器、输入输出口等），用它编程具有较大的复杂性，所以基本上仅由计算机专业人员使用。继而代之的宏汇编语言，它在许多功能和语句上比汇编语言大大前进了一步，有的用法已接近高级语言的程序块调用，功能大大增强。宏汇

编语言的掌握对计算机专业人员而言是必不可少的基本功，所以它仍然受到极大的重视。随着计算机的普及和推广，在计算机研究人员的不断努力下，各种高级语言应运而生。其“高级”之处自然是不再象汇编语言那样具体涉及机器的内部结构（虽然目前多数高级语言已有直接或间接处理机器硬件结构的功能），让实际的硬件系统及机器结构在用户面前是“透明”的，即虽然存在，但用户却看不见，或不必看见。由于这些语言和自然语言有某种程度的近似，使程序结构和程序设计变得易懂、易学、易读、易编，逐渐为许多非计算机专业的广大应用人员所掌握和喜爱。

为了满足实际应用的需要，计算机语言和语言的支持环境正在不断地发展和增强，其发展的趋势主要表现在以下几个方面。

1. 语言本身的发展

软件的基础是语言，软件质量好坏的一个重要部分也依赖于所采用的语言。除了常规语言（比如目前流行的 BASIC、FORTRAN、Pascal、C、以及 MASM 等语言）仍在不断地完善、不断地发展并显出强大的生命力外，新的语言类型，如面向目标的语言，函数式程序设计语言，逻辑程序设计语言，面向数据库的语言，人工智能语言，并行程序设计语言以及分布式语言都已开始了它们新的生命征程，在激烈的竞争和应用中闪烁出各自的光辉。语言本身的发展加速了软件的发展，也促进了计算机硬件系统的发展。

我们不打算去评价各种语言的所谓“好坏”，一种语言的生命力既取决于历史条件和经济发展条件，也取决于它自身的适应性，最终取决于广大计算机应用人员对它是否承认和接受。目前，各种语言新版本层出不穷，常规语言间的差异日渐缩小，正是这种趋势的表现。

2. 操作系统与语言的一体化

一个语言系统要能够正常地生存、运行并发挥其效能，既与其编译系统的设计有关，也与系统支持环境有关，有了适当的支持环境，语言则可以发挥其强大的功能。

众所周知，语言（确切地说应该是它的编译系统）可以独立地存在于计算机系统中，可以和操作系统完全分开，成为两个独立的部分，在没有或者未装入操作系统的情况下，有的语言本身也能在机器上执行。在语言程序中也可以不经过操作系统，而是通过基本的 I/O 处理程序直接与系统打交道。这些基本的处理程序有人称这为语言的基本库，是语言系统本身带有的。当然，语言的编译系统也可以在操作系统的支持下进行工作，它可以使用或者调用所在机器操作系统的某些原语和系统调用，通过它们完成某些功能性操作。有人常把这样一些由操作系统所提供的调用支持称为语言的系统库。目前，语言研究的另一个方面是把操作系统和语言一体化，这样做既扩大了语言功能，还使系统及语言本身的并行性等功能大为增强，例如，目前流行的 OCCAM 语言即是一例。

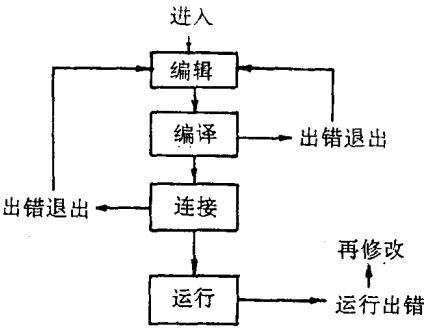

图 1.1　语言程序处理过程

3. 集成软件及语言集成化

在传统的语言程序编制和调试过程中，往往要经过多次修改、编辑、编译、调试而再运行，

这些步骤和过程一般是分别独立的，比如图 1.1 所示的语言程序处理过程。

每一次都要重新启动并运行有关的编译程序，连接程序，编辑程序或者实用程序，长时间地反复调用和退出，形成时间上的浪费，极大地影响了工作效率和软件开发人员的情绪。

软件集成化是近来研究的一个热点，其中一个方面是功能性的集成，目前已经有许多集成化的软件问世，也有人称之为组合软件。这类软件在功能上集中了文本处理，窗口技术，图形技术，存储器分配管理，屏幕及键盘操作等各种功能，这种集成软件的发展正方兴未艾。在语言编译系统的发展上，人们把 Borland 公司的 Turbo 语言系列编译系统软件称为初步集成化的软件，比如 Turbo BASIC、Pascal、C、PROLOGO 等，它们把语言及其支持环境以及相关的实用程序功能结合在一起，在逻辑上和物理上把编辑、编译、连接、排错、调试、运行以及简单的窗口功能集为一体，一旦进入这个环境，就可以在交互引导方式下完成所有的工作。这样，既大大提高了软件编制调试的速度，也给用户提供了一个良好的语言编程环境。正如硬件的集成化提高了低层次的基本运算速度一样，软件及软件工具的集成化提高了高层次的软件开发和信息处理的速度和效率，目前，这种集成软件的发展和研究还在积极进行之中。

4. 混合语言与程序接口

语言的另一个应用方面和程序设计技术发展方向是混合语言程序设计。混合语言并不是一种新出现的自成系统的新型语言，而是一种程序接口技术。它是在现有语言的基础上实现不同语言程序之间的相互调用。它既可以发挥各种语言的长处，完成各种特殊的功能，也能及时解决单独一种语言不能解决的某些问题。此外，还能利用目前已有的各种程序模块、目标库程序模块（或经过稍微修改）在短时间内组成新的应用程序，从而缩短软件的开发时间、降低成本、提高软件的效益。要完成混合语言设计，就要建立各种语言之间的接口，最常用的是各种语言之间相互调用的接口方式，也即程序间的接口。

1.3 软件接口与程序接口

软件接口是指不同类型的软件在同一运行环境下彼此之间进行的程序模块调用，参数传递，数据共享，功能相互支持与补充的方式与技术。目前，各种计算机系统都有极其丰富的软件资源，其中包括了各种实用程序，工具软件，应用软件包，以及数据库等等，也包括了以各种语言编制的各类程序，它们可用来分别完成不同的功能。

软件接口从应用的角度来看有下面几种：

1. 语言程序间的接口

这种接口技术又称为调用型接口（CALL TYPE），即在一种语言程序中调用由其它语言编写的程序模块，其间也含有少量的参数传递。调用方式大多采用 CALL 方式（虽然有的语言程序中并不出现 CALL），所以，又称为程序调用型接口。

2. 数据共享型的接口

在软件编制和开发工作中，如果开发的软件规格较小，具有较简单的科学计算或者数据处理，那么，只选择一种高级语言或者数据库语言来编程是可行的。然而，对于要开发功能很强，规模很大的大型应用软件，系统软件以及图形库支持软件等，并且软件中既具有大量数据处理，又具有大量科学运算，甚至具有较强的实时功能的系统来说，采用单一的高级语言或者数据库语言就很难完成预期的计划和功能，而且在编程中也会遇到极大的麻烦。因为各种语言都有其长处，也有其局限性，一种语言鞭长莫及之处，正是另一种语言驾轻就熟之区，集各语言之

精华，扬长避短，就可以大大提高系统的效率、功能和灵活性。

数据共享型接口是混合语言程序设计中一个极重要的方面，在具体处理过程中有两种情况。第一种是属于缓冲通信型的方式，即在参数传递的基础上进行诸如数据块，字符串等的传送，通过缓冲存储区形成数据共享。如图 1.2 所示

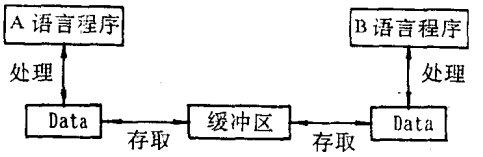

图 1.2　一种数据共享型方式

这里，缓冲区可以是分别的单独局部存储区，各自互不干扰，也可以是一种类似临界区的共享数据区，根据不同情况有相应的操作处理。

第二种是数据文件共享型，各个语言本身对数据或文本组成数据文件的型式，在数据文件的格式或者存储格式上，有的语言之间是相同或者类似的，有的则不同（也即两种语言间的文件不能通用）。此时，进行这种类型的操作要按照不同的文件格式进行一定的转换，才能交付给另一语言程序使用，如图 1.3 所示

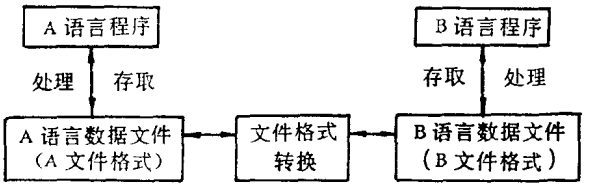

图 1.3　数据文件共享型方式

我们知道，各种语言及数据库的文件都有各自的文件格式。文件格式是进行文件存取的依据，而文件的共享，其关键也是文件的格式问题。例如：A 语言文件格式为 AS，B 语言的文件格式为 BS；A 语言若要想访问或者存取 B 语言的文件，就必须先将 BS 格式转化为 AS 格式，继而才能对这个文件进行操作。所以，实现文件共享的途径之一是将所需要的文件转化成具有所需要的格式的文件，如 dBASE 与其它语言之间对数据文件的共享办法就是一个例子。

限于篇幅，本书重点放在第一种程序接口即调用型接口的建立和应用上。

1.4　混合语言程序设计的进展与应用

计算机语言发展到今天，除了汇编语言外，已涌现了各种类型，各种面向的高级程序设计语言，目前比较常用的如 BASIC，FORTRAN，Pascal，C，ADA 语言和 LISP，PROLOG 以及数据库语言 dBASE 等等。各种语言结构各异但又具有一定的共性。在不同的应用环境中有可能选用不同的语言以完成某些特殊的要求。

熟悉计算机语言的读者都知道，BASIC 语言作为一种简单易懂的计算机算法语言，适用于一般的数值计算和事务处理，其人机交互性强，使用灵活、方便。它的实现方式具有解释型和编译型两种，近年来后者明显取代了前者，而且从原来的非结构化逐步形成有利于结构化编程的语言结构，尤其是后期加入的图形功能和音响控制功能很受人们的青睐。不过，在数值允许范

围、变量个数、数组维数、自定义函数的自变量个数，以及在速度等方面都有所限制。因此，它对于大型、复杂的数据处理，高速或者实时应用就显得不能令人满意。

Pascal 语言是一种适合于结构程序设计的语言，其功能强，数据类型和结构丰富，其程序结构简单明了，可读性很强，是一种很好的程序设计教学的范例型语言，既用于数值及非数值问题的描述，又能很好地用于系统程序设计。

国际上目前流行较广的适用于科学计算的高级语言首推 FORTRAN，其优点在于标准化程度高，便于程序交换，较易优化，执行效率较高，而且，它拥有的高精度型的数据结构与运算结构，为其它高级语言所不及，故倍受科技人员的欢迎。

近年来应用进展迅速的 C 语言是一种功能很强，使用灵活，可扩性强的语言，由于它具有许多类似于汇编程序设计技术的功能，如位操作等，又具有高级语言所有的结构化性等特点，C 语言也被称为“低级的高级语言”。它特别适合于系统程序设计。众所周知，著名的 UNIX 操作系统就是用 C 语言编写的杰作，UNIX 操作系统的巨大成功也伴随了 C 语言的巨大成功。此外，C 语言除了自身拥有的基本库和基于操作系统的系统库函数外，还允许用户建立自己的应用库，形成了一种可扩性很强的语言。近来各种图形软件，工作站系统软件，窗口软件均为 C 语言用武之地。不过其可读性稍欠佳，曾有人批评说：“编程太自由了反而不好理解”。

另一个深受人们重视，作用极大的语言是汇编语言，它面向机器，执行效率高，代码紧凑，常用来解决与机器硬件有关问题的编程。汇编语言是真正能体现具体的程序设计技术、风格和艺术的语言，可以编写出各种风格回异、引人入胜的程序。尽管用它编程稍嫌复杂，不便理解，然而一旦掌握，则可熟能生巧，汇编语言在实时控制、接口技术以及图形图象处理中倍受赞誉。

此外，多用于商业事务、经济管理领域的 COBOL 语言，用于人工智能和逻辑程序设计的 LISP、PROLOG 等语言，流行于数据库编程的 dBASE 等各种语言都显示出了自身的优势和特点。把各种语言的优势和特点尽可能地发挥出来，这就是混合语言程序设计的一个重要目的。再则，现存的语言已经生成和建立了无数功能强大的各种实用程序、库程序、工具程序，如何利用这些现有的模块，缩短新程序的开发周期，节省人力物力，并达到软件升级的目的，这也是混合语言程序设计的另一个目的。

八十年代初期，随着计算机技术和计算机语言的发展，为了解决应用领域中出现的问题，满足软件开发人员的需要，从而占领软件市场，世界各厂商、软件公司在花大力气开发新软件和集成软件的同时，把注意力投向了混合语言编程及程序接口问题，开发了一系列产品。美国著名的 Microsoft 公司率先推出了 MS 系列语言的混合程序设计，Borland 公司推出了 Turbo 系列的混合程序设计接口，Ashton－Tate 公司给出了 dBASE 与其它高级语言的连接方式。目前，虽然多数情况是针对同一语言系列的程序接口，不过，从处于同一运行环境下的程序模块来看，原则上是可以完成程序模块的相互调用的，复杂的问题在于参数的传递和数据的共享，不过，遵循语言之间的各种约定规则，问题是不难解决的。

混合编程方式允许用户根据自己的需要将几种编程源语言（例如：BASIC、C、FORTRAN、Pascal 以及宏汇编 MASM 等）组合成一种程序实体，这些源语言之间可互相访问、调用和传递参数。事实上，所有语言的扩充语言库中的例行程序对于混合语言编程来说，也是可以调用的。用户利用混合语言编程可以有效地使用机器的汇编语言，可以用任何一种熟悉的高级语言来设计程序的主体，而把那些使用频度很高的，速度要求很快的程序用汇编语言编写并作为程序块调用。

混合语言程序设计还可以完成从一种源语言到另一种源语言之间的传输和变换，比如，如

果要把一个大型的 FORTRAN 程序转换成对应的 C 语言程序，可以把这个 FORTRAN 程序的各个子程序用相应的 C 函数过程一个一个地替换，对应的功能的 C 代码马上就可以经联机得到。

混合编程特别适用于那些想尽快完成程序开发和推出或者销售自己的程序库的用户，他们可以采用混合编程技术用现存的任何源语言来生成程序库而且改动性很小。

混合语言编程技术与程序接口的进展是迅速的，以后的章节内容好比一张菜单，请读者根据自己的需要来选择阅读。

1.5 本书的组织与阅读

本书是实用性的技术书籍，读者可以根据自己的兴趣选择各章节的内容。在阅读中，本书所采用的术语及表达方式与各种语言用户手册中采用的概念是一致的，不过，对下述术语的概念需要作一说明。

Routine——可以被另一种语言调用的任何函数、子程序、子例行程序或者过程，其概念类似于汇编程序中的一个过程，采用 Routine 也是为了避免与 Pascal 中关键字 procedure 相混淆。

parameter——直接在两个程序(Routine)之间传送的一批数据，外部数据可以被所有的程序所共享，不属于数据传递。而术语 argument 在某些情况下可以与 parameter 互换使用，不过 argument 多指特殊的数值或者已给定参数的表达式。

interface——在不同的语言格式间进行有效通信的一种方式和方法，对于高级语言来说，接口通常是通过某种形式说明而建立起来的。

本书所举的程序实例和运行环境是针对目前主流微机 IBM PC 系列，Microsoft 的 MS 语言系列和 Quick 系列，以及 Borland 的 Turbo 语言系列。目前，这些语言的发展都提供了混合语言程序设计的接口，这些语言的版本可以是：MS 语言系列 C 语言(V3.0，V5.0 或 V5.1)，Pascal 语言(V3.32 或 V4.0)，FORTRAN 语言(V4.0，V4.1 或 V5.0)，BASIC 语言(V5.36 或 V6.0)；Quick 系列有 Quick C 语言(V1.0 或 V2.0)，Quick BASIC(V4.0)；Turbo 语言系列有 Turbo C (V1.0)、Turbo Pascal 语言(V3.01，V4.0，V5.0 或 V5.5)、Turbo BASIC(V1.1)以及 MS 的宏汇编 MASM(V3.0，V4.0，V5.0 或 V5.1)，读者可以利用这些语言版本进行混合语言程序设计。

在下述章节中利用上述版本之一举了一些实例，这些实例是极简单的，只是为了说明接口如何建立和说明的，读者可以根据这些规则编写自己的混合程序。在每一章末有一些练习题，通过练习，读者也能巩固所学到的混合语言编程技术。不一定非要从头开始阅读本书，每一章都是一个独立的部分。混合语言程序设计是一种实用技术，只有通过具体实践才能掌握它并取得有效的结果。

第二章　混合语言的编程环境和数据结构

本章主要讨论了如何在两种语言程序模块之间建立一个接口；描述了混合语言编程的环境要求，以及语言间的命名约定、调用约定和参数传递约定；也讨论了程序的运行环境，存储器模式以及语言库的问题，并通过一些实例作了说明。举例中一般采用整型参数，这种参数的传递相对来说是最容易的，至于较复杂的数据类型的共享与处理，如对字符串，数组数据等的处理，将在专门章节中进行讨论(第八章)。

2.1　混合语言编程的环境要求

混合语言编程的环境是什么，各语言之间有什么不同，如何在这些不同点上建立语言程序间的接口，这是混合语言编程首先要了解的问题。因为以各种源语言编制的程序最终都要在同一个运行环境下执行，那么，它们之间的关联是什么？在什么时候和如何来进行语言程序间的调用，下面的叙述可使读者逐步了解整个概貌。

软件接口的一个重要类型是程序间的接口，统称程序接口。这种程序接口方式一般以调用型方式为主，由于各种语言在结构等细节上规则不同，必须考虑程序模块之间的接口，了解造成不同的原因并找出对应解决办法。这些不同点除了语言自身的结构与规则不同外，在程序调用时一般表现在如下一些方面：

1. 命名约定和使用的不同

各个语言在编译时对变名的处理以及对目标文件中标识符名的生成方式各有不同。以MS系列语言为例，FORTRAN语言将截取程序中所有变名(如变量名、常数名、过程名、函数名等标识符)的前6个字符作为目标名。C语言和Pascal语言一般截取名字前的8个字符作为目标名，此外，C语言还自动在名称前方加上一个下划线“_”，如使fact变成_fact。汇编语言可以接收承认31个字符为目标名，而BASIC语言的目标名则最多可以容纳40个字符，并在编译时自动消除名字中的类型描述字符(如%、&、#等)。所以，在混合语言编程中必须考虑命名的约定问题。

2. 参数传递的方式不同

在各种语言程序中，参数的传递方式一般有四种，即传值(call by value)，传址(call by reference)，传名(call by name)，传结果(call by result)，最常用的是传值和传址两种。

两个程序之间的参数传递必须配合默契，方能得到正确的结果，才能真正达到数据共享。所以应当了解每一种语言在其缺省情况下的参数传递方式是什么，它可以允许几种传递方法，需要采用什么样的技术才可以改变参数的传递方式等等。

另外一个与参数传递有关的问题是各个语言对堆栈的操作。由于今后大部分数据和参数是通过堆栈传送的，所以必须了解各语言的堆栈结构，生成方式，参数入栈方法等等。我们知道，对于BASIC(编译型)，FORTRAN和Pascal语言，其参数压栈的顺序是与参数在参数表中出现的顺序相同的，即顺序从左到右，而C语言却刚好相反，顺序从右到左，例如，过程fact(a，b，c，d)的参数压栈结果就如图2.1所示。图中左边对应Pascal等语言的压栈情况，右边对应C

语言参数的压栈情况。而且，参数在堆栈中所占的字节与调用类型，参数类型等都有关系，也必须弄清楚，因为它涉及到堆栈的恢复和正常返回。此外，还有字符串，数组等特殊数据的定义，说明及传递，都必须按某种约定进行，使两种语言都按彼此兼容的方式对同一数据类型或者结构进行操作和处理。

3. 调用方式与实现

混合语言编程的程序接口通常都包含了一个调用的过程，即对其它的外部程序过程进行调用。这些外部过程既可以是用本语言编写的，也可以是以任何一种语言编写的。它可以是一个函数调用，一个过程调用或者是一个子程序调用。例如，一个 BASIC 语言主程序模块在执行过程中可能需要一项特殊的任务，这个任务可以单独编写完成，而且可以不用 BASIC 语言来编写，不作为 BASIC 本身的子程序，而是采用另外语言编程，以后再对此程序模块进行调用。

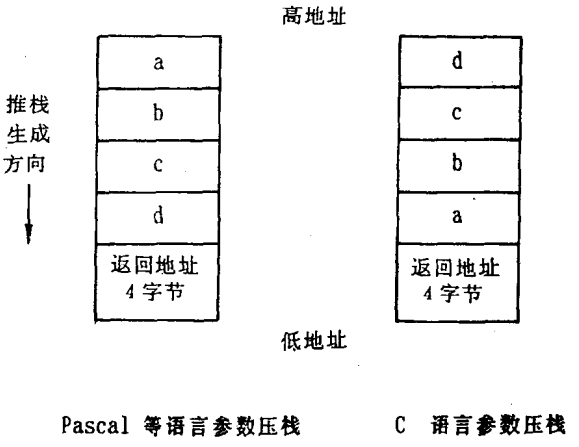

图 2.1 不同的参数压栈结构

混合语言涉及用多种语言书写的多个程序模块，这些程序模块不是采用同一个编译程序，而是分别采用不同的编译程序或者汇编程序来编译或汇编的。例如，用 BASIC 语言写的主程序可以用编译型 BASIC 的编译程序编译，另一个用 C 语言写的被调程序用 C 语言编译程序编译，然后把两个目标文件连接到一起。这种混合语言调用的结构可从图 2.2 看出，这里展示的是 BASIC 语言程序对 C 语言程序模块的调用。

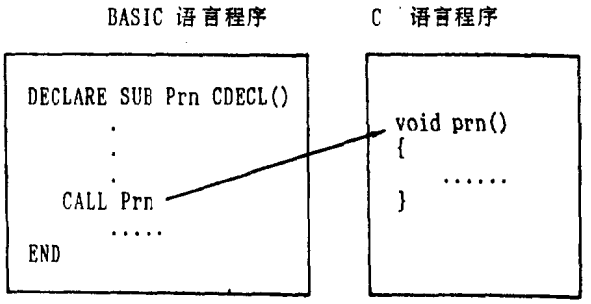

图 2.2 混合程序调用示意图

在这个例子中，BASIC 程序对 C 语言程序模块的调用是通过 CALL Prn 进行的，它在形式上完全与 BASIC 语言对其本身子程序的调用类似。不过，混合语言之间的调用与 BASIC 语言本身两个程序模块之间的调用是不同的，也就是说：

1) 子程序 prn 实际上是用 C 语言程序实现的，采用的是标准的 C 语言语法。

2) BASIC 语言中的调用实现是通过 DECLARE 语句来引入的，而且利用了 CDECL 关键字来产生与 C 语言兼容的实体。可见，这个 DECLARE 语句是这个混合语言程序设计中的一个“接口”语句，其作用将在第三章详细讨论。每一种语言都提供了它们自己的接口形式。

在调用的实施中，尽管语言的句法语法不同，但函数，过程和子程序（以及 FORTRAN 语言中的子例行程序）的调用是类似的，主要的差别在于某些程序需要返回数值，而另外一些程序则不返回数值。我们可以把具有返回值的程序互换，也可以把无返回值的程序互换。当然，这里指的程序是泛指可以被其他程序模块调用的任何函数、过程和子程序。表 2.1 列出了不同语言中的程序间调用的程序模块对应情况。

表 2.1 语言间等效的程序调用

语言	有返回值	无返回值
BASIC	FUNCTION procedure	subprogram
C	function	(void)function
FORTRAN	function	sunbroutine
PASCAL	function	procedure
MASM	procedure	procedure

从这个表可以看出，一个 BASIC 程序模块可以通过一个子程序调用来对 FORTRAN 语言的子例行程序进行调用。但是，如果要对于 FORTRAN 语言中的函数进行调用，则 BASIC 程序也应当使用对应的函数型调用，即 Function 调用，否则，前述调用虽可以完成，但返回数值却丢失了。当然，这里应当提示一下，在 BASIC 语言中的自定义（DEF FN）函数是不能被调用的；另外，GOSUB 语句所引入的子程序也不能由另外的语言所调用。

综上所述，为了正确地完成混合语言程序设计，针对上述三个方面的问题，我们需要分别讨论语言间的命名约定，调用约定和参数传递约定的要求。

2.2 命名约定要求

什么叫命名约定（naming convention）？命名约定也叫变名约定，即是为了解决不同语言对名称标识符（如变量名，参数名，过程名，函数名等）的不同处理，对目标文件名的不同的长度限制的约定。这种约定是由编译程序在把一个程序块放入目标文件中去之前改变这个程序的名称。而在我们进行混合语言调用时，采用一个兼容的、大家都承认的名称是非常重要的，如果被调程序名在各自的目标文件中是用不同的名称存放的，那么，连接程序就不可能找到这个程序模块，自然也匹配不了。只有通过出错信息告诉编程者调用中含有“未定义”的或者“未解决”的外部名称，而不能完成实际的连接。

以 Microsoft 公司语言系列编译器为例，它把编译后形成的机器码放入目标文件，同时也把所有需要进行公共访问的程序模块名和变量名也放在其中，以便连接程序 LINKER 把在一个程序模块中的调用程序名与在另一个程序模块中定义的程序名相比较，最后确定两者是否匹配。上述名称和标识符都用 ASCII 码格式进行存储，编程者可以采用调试程序 DEBUG 来把一个目标文件的名称字节取出，以观察到它们是如何存放的。

BASIC，FORTRAN 和 Pascal 语言大致上采用了同样的命名约定，它们把名称的每一个字母都转换成大写字母，在 BASIC 语言中还把这些变名带有的类型说明符（如：%、&、!、#、$）等自动去掉，然后放入对应的目标文件。