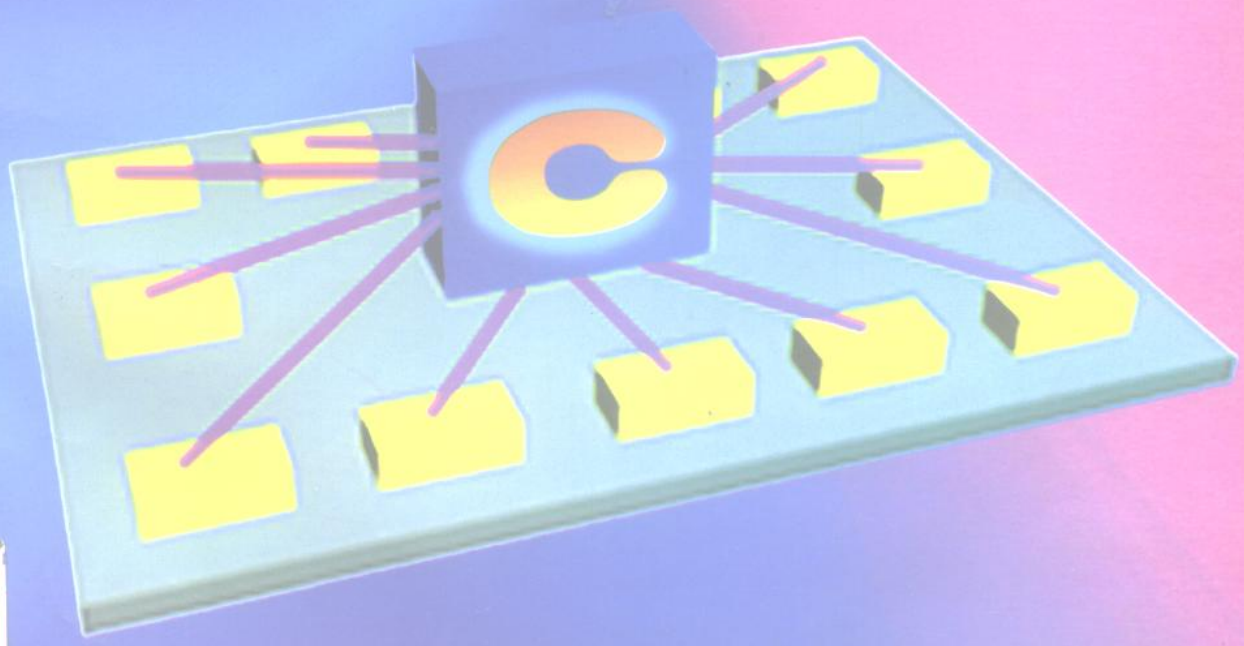
实用C语言接口技术与实例

王振宇 黄立波 编著



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
URL: http:/www.phei.co.cn

实用C语言接口技术与实例

实用C语言接口技术与实例

王振宇　黄立波　编著

電子工業出版社
Publishing House of Electronics Industry

内容简介

本书全面而系统地介绍了目前使用最多的C语言与其它高级语言、操作系统、网络、工作站等的接口编程与实例，如与汇编语言、BASIC、FORTRAN、PASCAL、Ada、Oracle、dBASE(Foxbase)、Sybase、Windows、DOS、UNIX、VMS、NOVELL、SUN等，内容广泛，代表性极强，是一本软件开发人员必备的参考工具书。

本书适合软件开发设计人员、C语言使用人员、中高级程序员人员使用。

书　　名：实用C语言接口技术与实例
著　　者：王振宇　黄立波　编著
责任编辑：高平　吴源

印 刷 者：中国农业出版社印刷厂

出版发行：电子工业出版社出版、发行　URL：http://www.phei.co.cn
　　北京市海淀区万寿路173信箱　邮编100036　发行部电话：68214070
经　　销：各地新华书店经销
排　　版：电子工业出版社计算机排版室
开　　本：787×1092 1/16　印张：13.25　字数：320千字
版　　次：1997年4月第1版　1997年4月第1次印刷
印　　数：5000册
书　　号：ISBN 7-5053-3116-7/TP·1101
定　　价：16.50元

序

软件开发所用语言的选择受诸多因素的制约，必须考虑应用领域的特点、运行效率高、开发效率高、符合程序员习惯、符合行业传统与规范等等。由于没有一种语言能同时满足这些要求，多种语言的混合编程就应运而生了。C 语言作为一种高级语言，当然也有这种一般意义下的混合编程问题。而要混合编程就得有接口。

但是，对 C 语言来说还有它的特殊性：即 C 语言除了用作应用软件的实现语言之外，它还被广泛地用作操作系统、窗口系统和网络软件的实现语言，因而在相应的 C 程序中要经常同操作系统的系统调用、窗口系统的命令函数和网络的命令打交道，或者是调用，或者是被调用，而打这些交道都需要通过接口。

本书就想成为一本阐明 C 语言接口的工具书。然而，要达到这个目标又谈何容易！上面说了 C 语言要对接的对象有那么多，不同的机器，不同的操作系统。C 语言又有各个不同厂家提供的编译系统的不同版本，实现的语言有差异，实现的方法和库函数都各有不同，有关资料甚多。这样一本字数受到严格限制的书是很难全面覆盖的，只能由编著者略带主观色彩加以取舍：机器大体上以微机为主，也涉及了 SUN 工作站等；操作系统大体上以 PC DOS 为主，也涉及了 UNIX 和 VMS；编译系统则以 Microsoft C 为主，也涉及了 Turbo C 和 C++的编译系统。

本书按以下的次序展开：第一章讨论混合编程的一般概念；第二章是 C 语言同汇编语言的接口；第三章是 C 语言同其它高级语言的接口，主要涉及 BASIC、FORTRAN、Pascal 和 Ada；第四章阐述 C 语言同数据库的接口，主要涉及 dBASE(Foxbase)、Oracle 和 Sybase 等；第五章是关于 C 语言同窗口系统 Microsoft Windows 和 X-Windows 的接口；第六章阐述 C 语言同操作系统的接口，主要涉及 PC DOS、UNIX 和 VMS；第七章则是 C 语言同网络系统的接口，讨论网络环境下的 C 程序设计。

在本书编写过程中，参考了国内已出版的几本有关混合编程、语言接口、窗口系统和网络软件的著作，书名已列在参考文献中。谨向这些著作的作者表示衷心的感谢，对没有联系上的作者还要表示歉意。

电子工业出版社编辑吴源、高平同志在本书编写过程中从选题到定稿都给予大力支持。梁祥丰社长在汉期间也曾给予关心，谨此一并致谢。

王振宇　黄立波

目　录

第一章　混合编程与语言接口

1.1　混合编程的概念

到二十世纪九十年代的今天，计算机的应用不仅深入到科学技术的各个领域，而且涉及到社会生活的方方面面。在这种情况下，软件的开发已成为计算机应用的“瓶颈”，其中程序的编制工作又占很大份量。为了提高程序的质量和编程效率，已经发展了一种“多种语言混合编程”技术。由于C语言自身的优点和它同UNIX操作系统的密切关系，C语言日益广泛地流行和普及起来，C语言同其它语言的混合编程和接口问题也成为大家关注的热点。

1.1.1　混合编程需求的提出

对一项软件工程或程序开发工作而言，首要目标是保证达到软件或程序的所需质量，其次是尽可能降低开发的成本。

软件或程序的质量，首先是指能正确实现所需的功能。在此前提下，还可提出其它的要求。在计算机发展的早期，要求的是占用尽可能少的计算机资源，在当时就是指令条数尽可能少，占用的存储空间尽可能小。随着时间的推移和计算机技术的进步，在多数应用领域，时间和空间资源不再是关键性的制约因素。软件技术的进步提供了能正确实现所需功能的必要手段。而另一方面，由于软件开发成本在整个计算机应用成本中所占比重的不断提高，软件维护工作重要性的上升，软件的开发效率和易维护性就成为越来越突出的关注点。

对于一个具体应用而言，应用特性以及对制约因素和综合因素的考虑也各有不同：例如，一个实时控制系统既可能要用到并经常访问计算机的硬件特征，又有对定时和处理时间长短的要求（根据需要，又分软实时和硬实时）；对于一个MIS（管理信息系统），可能对计算机的硬件特征访问很少，但对机器的响应时间有一定要求；对一个科学计算程序而言，重要的是把计算算法正确地表达出来，程序可能根本不会涉及计算机的硬件特征，对处理时间的要求一般不太苛刻。对于要多次运行的程序，重要的是运行效率，对开发效率的要求相对要低些；而对于运行次数较少的程序，可能重要的就是开发效率了。

如果把对软件生存期的支持考虑进去，我们把这些要求大致归纳以下几个方面：

运行效率
对硬件特征的访问能力
开发效率
程序的易读性、易理解性、易维护性
程序的可移植性（对机器的独立性）
允许并入已有软部件或使用某些软件开发工具开发的软部件的能力
支持大型软件开发的能力

列举的这几个方面既不是完备的，也不是相互无关的。

为了改善软件质量和提高软件开发效率，计算机发展早期使用“手工编程”方式现在已经

基本淘汰，人们正在大量使用软件开发工具乃至集成化的软件开发环境。各种计算机编程语言就是这些工具中的一族。

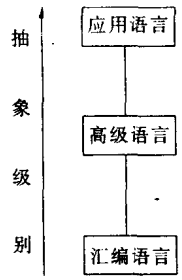


图 1.1-1 语言的抽象级别

几十年的计算机发展史中的一个重要方面是计算机语言的发展史。人们要告诉计算机需要它做什么事，必须通过一定的语言。从最初的机器语言和汇编语言，逐步发展了几百种高级语言，但汇编语言依然存在。以后，又发展了面向某一类应用的语言，我们不妨称之为“应用语言”(如图 1.1-1 所示)，而现在汇编语言和高级语言还是依然存在，这当然是由于它们都有各自存在的价值。例如，撇开某种语言本身的优缺点不说，由于它在历史上有过辉煌的地位，已经积累了大量用该语言编制的软件，这些软件经过了大量运行和充分调试而成为宝贵的软件资源，它依然会有生命力，即使发现了有某些缺陷，也不会淘汰它，而是使它逐步完善，或者使它能同其它语言协同工作(混合编程)。

用 Dijkstra 的话说，“抽象”是把握客观世界的重要手段。我们也可以用“抽象”这个观点来看待计算机语言。

机器语言(指令系统)是计算机硬件的抽象；汇编语言是机器语言的符号抽象，其中已经开始有了自二进制数到数学中的“数”的抽象；

高级语言(算法语言)是机器语言到数学(算法)表示的抽象，其中包括了数据抽象(数据结构和数据类型)、处理抽象(语句和控制结构)和功能抽象(过程和函数)。

应用语言本质上是高级语言，不过它更面向一类应用，其中更突出了功能抽象。

对于编程工作而言，这些语言在表达能力、表达的方便程度、易理解性和维护性、编程效率和运行效率方面各有不同。

要指出的是，对任何一个具体语言而言，前述要求中有些是不可兼得的。实际上，没有任何单一语言在这几个方面都提供令人满意的能力，因而有些应用程序的编制需要用到不止一个语言去描述它，这就是所谓的“多种语言混合编程”。它的基本目的就是能在同一个程序中充分发挥多种语言的优势，取得最大的综合效益。

多种语言混合编程的通常做法是首先根据应用需要选定一个主语言。多数应用会要求选一个高级语言或应用语言作为主语言(当然也不排除用汇编语言作主语言)，用以表达程序的顶层结构。程序中的某些模块或过程(或函数)可依需要用其它语言实现，当然可以是程序员自己实现或利用别人的程序部件，在后一种情况，就是不同形式的软件重用。

从软件工程规范的角度说，多数应用领域并不规定使用什么语言，虽然有些领域有某种习惯使用的语言。有些领域，例如军用，对语言的使用有些规定或限制，包括对混合编程的规定和汇编语言使用的限制。

从大类来说，混合编程分为三种，即

高级语言与汇编语言的混合编程

不同高级语言之间的混合编程

高级语言与应用语言之间的混合编程

下面几小节将逐一进行讨论。

1.1.2 高级语言与汇编语言的混合编程

对软件设计人员和程序员来说，采用高级语言与汇编语言的混合编程的方式，通常是一个

需要慎重考虑的选择，要充分考虑应用的需要和汇编语言及高级语言的主要特征。下表列出了这些特征。

表 1.1-1 高级语言与汇编语言的主要特征

	汇编语言	高级语言
运行效率	高	低
访问硬件	高	低
开发效率	低	高
易读性	低	高
机器独立性	低	高
软件兼容能力	低	高

表中的“高”、“低”是相对而言的。其实，高级语言经过优化编译系统产生的目标代码也有相当高的运行效率，表中这一栏的“低”是说：不论如何，相对于汇编语言的运行效率，它总是要低一些。高级语言与汇编语言混合编程的主要形式是以一个高级语言为主语言，嵌入若干汇编语言代码段或用汇编语言实现的过程或子程序。几乎所有高级语言的所有编译系统的所有版本都提供了同汇编语言混合编程的能力。

本来，人们使用高级语言的好处之一就是可以摆脱使用二进制编码、机器指令或汇编指令的繁琐细节，将注意力集中在正确实现所需的算法，而不必担心编译系统产生的目标代码的执行对操作环境和操作系统的依赖性，但事物总是有两面性：有些程序恰恰又要用到操作环境的某些特征。不得已只有在某些程序段采用汇编语言编程。采用这种混合编程的目的不外乎：

(1) 某些处理需要访问机器的硬件特征，例如，系统时钟、外部数据固定写入某地址，终端特性，外部中断，机器状态，网络通讯等；这些特性用高级语言表达比较困难或不可能，因而必须采用汇编语言来编制它们。

(2) 某些处理需频繁运行，单次运行速度的稍许提高就会显著提高整个程序的运行效率。曾有人提出一个有名的 20-80 法则：程序正文长度中 80%的部分的运行时间只占程序总运行时间的 20%，而正文长度中的另外 20%的运行时间却要占程序总运行时间的 80%。就是说，相同长度的正文段所占用的运行时间可能各不相同，而且可能相差很大。因此提高某些运行频繁的程序段的运行速度就显得必要了。采用汇编语言来编制这些程序段就是解决这个问题的有效手段之一。

(3) 有现成的汇编语言程序段可用。

采用汇编语言和高级语言混合编程方式就充分发挥两种语言的长处。

要指出的是，国内外的各种软件开发规范都对汇编语言代码段的插入作了专门的规定和限制，例如汇编语言代码段所占比例的上限，使用时需经有关软件管理部门或开发委托方的同意或批准。这主要是由于汇编语言代码段的插入会带来可读性的降低、维护难度和软件生成难度的增加等。

1.1.3 不同高级语言之间的混合编程

我们用表 1.1-2 来列举几种主要高级语言的特征。

表 1.1-2 几种高级语言的主要特征

	C	Basic	Fortran	Pascal	Ada
结构化	一般	差	差	好	好
结构丰富	一般	差	一般	好	好
模块化	好	差	一般	差	好
封装特性	一般	差	差	差	好
可用程序库	多	多	多	不多	少
低级设施	好	差	差	差	好
实时性	一般	差	一般	差	好
优化编译	有	有	有	有	有
可读性	差	差	一般	好	好
标准化程度	高	差	好	好	好
类型检查	差	差	好	好	好
运行时检查	差	差	差	差	好

关于表 1.1-2 有几点说明：

① 这只是本书作者个人对这些语言所具特征的看法，决不想说服谁使用什么语言或放弃使用某一语言；

② 有的语言有多种变种，例如 Basic 和 Fortran。变种之间在特征方面也存在差异，结构化 Basic 和 Fortran 当然在结构化特征上已有了很大改进；

③ 同一语言在各种计算机上又有多种版本的实现，实际支持的特征也有差别，编译系统的性能和优化程序也不尽相同。

不同高级语言之间的混合编程主要用于这样一些场合：

(1) 软件项目按必须遵守的软件开发规范或其它因素所选的主语言虽然能满足主要要求或部分要求，却不能满足另一些要求。例如，有些程序部件需要访问某些低级设施，而主语言不具备这种能力，却有另一个高级语言具有这种能力；

(2) 项目的开发需要调用某些用别的高级语言已经开发出来的软部件，以利于提高软件质量或缩短开发周期；

(3) 项目的开发没有规定必须采用何种高级语言，给程序员留下一定的自由度去选择自己熟悉或喜爱的语言。

一般来说，并非任何两种高级语言都能混合编程。首先，解释型高级语言不能同编译型高级语言混合编程。其次，在同一个计算机平台(包括硬件和操作系统)上，如果两个高级语言的编译系统是由不同厂家提供的，一般也不能混合编程。即使是同一厂家提供的编译系统，在混合编程时也存在若干限制和必须遵守的接口约定。

高级语言之间混合编程的主要方式是从主语言发出若干过程或函数调用，而这些过程或函数是用另一个高级语言编制的。两种以上高级语言混合编程的情况和需要相对来说较少出现，而一种高级语言作为主语言，既同另一个高级语言混合编程，又同汇编语言混合编程的情况则屡见不鲜。

1.1.4 高级语言与应用语言之间的混合编程

表 1.1-3 比较了应用语言与高级语言的某些特征。

表 1.1-3 应用语言与高级语言某些特征的比较

	应用语言	高级语言
功能抽象	高	较低
计算能力	较低	高
适应性	低	高
开发效率	高	较低
运行效率	较低	高

我们所说的应用语言指数据库语言、图形语言等。前面说过，同其它语言相比，应用语言更突出功能抽象，这也正是应用语言的优点和长处所在。例如，如果用其它语言实现一个数据库查询操作，可能需要几十行源代码，而用 SQL，对用户来说，一个语句就够了。数据库系统会根据这个语句找到要查询的数据库，再按查询条件找出一组符合条件的记录等等；再如用高级语言绘制一个五角星，可能也得几十个语句，而用专门的图形语言，也许一个语句足矣。

高级语言与应用语言之间的混合编程有两种方式：一种是以高级语言作为主语言，调用应用语言的结构；另一种以应用语言作为主语言，调用以高级语言编制的过程或函数。

当采用高级语言作为主语言时，对应用语言结构的调用又分两种方式：即嵌入式和过程式。所谓嵌入式就是在高级语言程序中直接嵌入加了特殊标志的应用语言结构，这自然是相当于对原来的高级语言作了扩充，因而必须对源代码进行预处理才可提交给该高级语言的编译系统；所谓过程式，是用主语言写出若干过程头，而这些过程是通过应用语言系统的某些结构实现的。

由以上几节的分析可以看出，混合编程的方式大致如下：

(1) 首先选定一个主语言；

(2) 根据软件开发工作的需要和软件开发环境的资源选定一种或多种混合编程语言。从抽象级别上说，这些语言可以比主语言高，也可以比主语言低；

(3) 混合编程可以是语句级的，也可以是子程序(或函数、例程)级的；

(4) 根据选择，遵守必要的约定，使用规定的接口。

1.2 混合编程接口问题

上面已多次提到多种语言的混合编程方式问题。不论用什么方式，混合编程都要涉及接口问题。

实际上，在各种语言的设计和完善过程中，都已经不同程度地考虑了混合编程问题，预留了或增加了同其它语言混合编程的接口。我们把这些接口都分成两部分，一个是接口约定，另一个是接口机构。

1.2.1 接口约定

所谓接口约定，就是为实现混合编程程序员在编程时必需遵守的规则，这些规则中有些是带普遍性的，有些则因混合编程的语言和方式而异。这里要特别指出，我们这本书是以 C 语言为中心讨论混合编程的接口问题的，而恰好在这一方面 C 语言同其它语言有很大不同。例如，Basic、Fortran 和 Pascal 有相同的名字约定、调用约定和参数传递约定，C 语言却不在其列。

接口约定分三个方面：

1. 命名约定

用以解决不同语言在命名方面的差别所带来的问题。各种语言对用以标识程序对象的标识符(或称名字、名)都有自己的规定，因而在混合编程时必须有一套转换规则。程序员要遵守它，相应的语言编译程序要实现它。例如，在编译一个混合编程子程序时，要将它的名字按命名约定作相应的改变，形成子程序的目标代码的名字，以供调用。

表 1.2-1 各语言的命名约定

语言-编译系统	命名约定的字符处理	受理字符个数
Basic	将名字中的字母转换成大写	40(MS)
Fortran	将名字中的字母转换成大写	6 (MS)
Pascal	将名字中的字母转换成大写	8 (MS)
C	将下划线插入到名字之前	8 (MS)
Ada	不变	依编译系统而异
汇编	将名字中的字母转换成大写	依计算机系统而异

由此表可以看出，C 语言的命名约定同 BASIC，FORTRAN 和 Pascal 很不相同。

2. 调用约定

调用约定主要解决子程序的参数传递顺序问题。子程序的调用者和被调用者之间并不直接传递参数，一般是通过堆栈进行的。调用约定规定子程序调用者以什么顺序将子程序的实在参数推入堆栈，被调用者以什么顺序从堆栈中取走实在参数。此外，它还规定子程序调用返回时是由被调用的子程序还是由子程序调用者释放堆栈中的这次调用的参数区。各种语言在这些方面不相同或不完全相同，因而在混合编程时必须有一套转换规则。程序员要遵守它，相应的语言编译程序要实现它。

表 1.2-2 各语言的调用约定

语言	调用约定
Basic	调用者依参数在源程序中出现的次序自左至右压入堆栈
	被调用者以相应顺序取出参数
	被调用者恢复堆栈
Fortran	调用者依参数在源程序中出现的次序自左至右压入堆栈
	被调用者以相应顺序取出参数
	被调用者恢复堆栈
Pascal	调用者依参数在源程序中出现的次序自左至右压入堆栈
	被调用者以相应顺序取出参数
	被调用者恢复堆栈
C	调用者依参数在源程序中出现的次序自右至左压入堆栈
	被调用者以相应顺序取出参数
	调用者恢复堆栈

由此表可以看出，C 语言的调用约定同 BASIC，FORTRAN 和 Pascal 也很不相同。

3. 参数传递约定

由于子程序的参数可以是简单的数值或逻辑值，也可能是结构化的数据类型，如数组或记录，这曾经给参数的传递带来过困难。Ada 以前的所有高级语言在子程序调用时的参数传递是比较混乱的，有按值传递（有的语言叫值形参），按引用传递（有的语言叫作按地址传递，有的语言叫变量形参）两大类。有些编译系统又进一步把按引用传递分为按远引用（远地址）传递和按近引用（近地址）传递。因而在混合编程时必须有一套转换规则。程序员要遵守它，相应的语言编译程序要实现它。

表 1.2-3 几种语言的参数传递约定

语言	参数传递约定
Basic	近引用、远引用、按值
Fortran	近引用、远引用、按值
Pascal	近引用、远引用、按值
C	近引用、远引用、按值
Meridian_Ada	三种模式：in，out，in out

Ada 的参数传递在概念上完全避免了按值或按引用等令人困扰的问题，在形参和实参之间只存在值的“复制”关系，只有三种概念简单的模式：

in　　在子程序调用时，将实参的值复制给形参，形参是个常量

out　　在子程序返回时，将形参的值复制给实参，形参是个变量

in out　　在子程序调用时，将实参的值复制给形参，且在子程序返回时，将形参的值复制给实参。形参是个变量

当 Ada 程序调用 C 语言的子程序时，后两种模式，即 out 和 in out 可用 C 语言的形参实现之。

各种语言在参数传递类型说明缺省时的处理各不相同，见表 1.2-4。

表 1.2-4 各种语言在参数传递类型说明缺省时的处理

语言	参数传递类型说明缺省时的处理
Basic	所有参数都按近引用
Fortran	所有参数都按远引用
Pascal	所以参数都按值
C	近数组（小模式）按近引用
	远数组（大模式）按远引用
	非数组参数按值

1.2.2 接口机构

接口机构就是用以指明采用的命名约定、调用约定和参数传递约定的手段。各语言之间的差别就更大了。表 1.2-5 列出了同 C 语言混合编程有关的接口机构及其含义。

表 1.2-5 几种语言的接口机构

语言	接口机构及含义
Basic	DECLARE 语句中的 CDECL，指出该 Basic 例程使用 C 的命名和调用约定
Fortran	INTERFACE 语句中的和子例程定义语句中的[C]，指出该 Fortran 例程使用 C 的命名和调用约定
Pascal	EXTERN 语句和子例程定义语句中的[C]，指出该 Pascal 例程使用 C 的命名和调用约定
C	EXTERN 语句和函数定义语句中的[Pascal]和[Fortran]，指出该 C 例程使用 Pascal 和 Fortran 的命名和调用约定
Ada	子程序规格说明中的编用 INTERFACE(C,...)，指出该例程使用 C 的命名和调用约定

1.3 C 语言混合编程与接口

本书所讨论的 C 语言混合编程可分成两大类：第一类是以 C 语言为主语言，同其它语言混合编程；第二类是以一个非 C 语言的其它语言为主语言，同 C 语言混合编程。至于 C 语言和什么语言可以混合编程，并不是能任意组合的。除了语言特征和应用需要之外，还要取决于具体的语言编译系统，例如是那个厂家的产品及其版本。尤其是对于不同的高级语言来说，同一厂家的编译系统产品是保证它们能混合编程的重要条件。

要指出的是，对于一个具体的主语言和一个具体的混合编程语言双方来说，所需要的接口是不对称的：以本书的讨论对象为例，设 XXX 是一个与 C 不同的计算机语言，那么以 C 语言为主语言、同 XXX 混合编程所需的接口和以 XXX 为主语言、同 C 混合编程所需的接口是不同的。

第二章　C语言与汇编语言的接口

这一章主要讨论C语言与汇编语言的接口问题，包括从C语言程序调用汇编语言程序和从汇编语言程序调用C语言程序。C和汇编语言混合编程的具体方法随C编译和宏汇编的不同而有所不同。用Microsoft宏汇编编写的模块一般只能同Microsoft C的模块连接，而用TASM编写的模块只能同Turbo C编写的模块连接。C和汇编语言混合编程除应遵守C语言的调用规则外，还要考虑到汇编语言的许多特点。本章将首先介绍混合编程必须解决的几个问题，最后以Microsoft C的例子加以说明。

2.1　C语言与汇编语言混合编程规则

2.1.1　参数传递规则

C语言在调用一个函数时，传给函数的参数是按从右到左的顺序压入堆栈的。例如，如果发出调用"callit(a,b,c)"，则先压入c，接着压入b，最后压入a。

有些类型的变量是先转换成另一类型后才压入堆栈的：字符型(char)转换成整型(int)，无符号字符型(unsigned char)转换成无符号整型(unsigned int)，浮点型(float)转换成双精度型(double)。结构(structure)是整个压入栈的，也是按相反顺序。对于数组，压入堆栈的是指向数组的指针。近指针是16位，远指针是32位，且先压入段寄存器值，后压入偏移量。表2.1-1列出了压入参数时所占的字节数。

表2.1-1　不同类型参数在堆栈中的字节数

类型	转换后的类型	在堆栈中的字节数
char	int	2
unsigned char	unsigned int	2
short		2
unsigned	short	2
int		2
unsigned	int	2
long		4
unsigned long		4
float	double	8
double		8
near pointer		2
far pointer		4
array	指向数组的指针	4或2
structure		n

如果是从C中调用汇编，则汇编语言子程序的格式应如下所示：

```
mov bp, sp
sub sp, nl                    ;子程序内的自动变量
```