文章编号:1671-6833(2003)03-0010-04

Visual Fortran 基于 Win32 DLL 的混合编程技术

周振红[1], 宋宇伟[1], 郭恒亮[1], 杨国录[2]

(1.郑州大学环境与水利学院,河南 郑州 450002; 2.武汉大学水利水电学院,湖北 武汉 430072)

摘 要:Visual Fortran 与 Visual C++、Visual Basic、Delphi 的混合编程,可将Fortran 编制的数学模型融入到各种辅助决策综合信息系统中.其混合编程实现的途径为:在 Visual Fortran 中,将执行计算的各种过程集成 Win32 DLL;在其它基于对象的语言中导入 DLL,并对Fortran DLL 输出的过程实施调用.实现的关键是:在调用约定匹配的前提下,使 Visual C++、Visual Basic、Delphi 程序中声明的外部过程原型,与 Fortran DLL 中的导出过程在目标过程名、参数类型及参数传递上保持一致.

关键词:Visual Fortran; DLL; 混合编程; 调用约定; 参数传递

中图分类号:TP 311 **文献标识码**:A

0 引言

目前,各行各业都在大力开展适合本行业的综合信息系统建设,以应对全球信息化浪潮的挑战.而且,所开发的综合信息系统正由以信息管理为主,向以智能化辅助决策为主的方向发展,这意味着所开发的综合信息系统将需要嵌入各种数学模型.

当前,实现数值计算的主流平台是Visual Fortran,但它却不适合作为应用系统的集成开发平台,数学模型只能以"组件"形式融入到其它语言工具开发的应用系统中.在 32 位 Windows 操作系统中,"组件"可以是动态链接库 DLL,也可以是 COM 组件.最新版本的 Visual Fortran 6.5 可以直接建造这两种形式的"组件",不过当前应用最普遍的还是DLL 组件.故此,本文重点探讨在 Visual Fortran 6.5 中如何建造 Win32 DLL,以及如何在 Visual Basic 6.0[1]、Visual C++ 6.0[2] 和 Delphi 6.0[3] 中分别对其过程实施调用的混合编程问题.

1 混合编程的基础

在 Visual Fortran 与 Visual C++、Visual Basic 和 Delphi 之间进行混合编程之所以可行,是因为这些语言实现过程(子程序和函数的统称)的机制大致相同:函数名具有返回值,子程序名没有返回值.表 1 列出了过程在不同语言中的对应关系.

开展混合编程的另一个重要前提是数据类型的精确对应.虽然Fortran、Visual C++、Visual Basic 和 Delphi 中的数据类型不尽相同,但就数值计算常用到的整型和实型而言,它们之间有着精确的对应关系,见表 2 所列.

表 1 不同语言所对应的过程

Tab.1 Corresponding routine in different languages

语言	函数(有返回值)	子程序(无返回值)
Fortran	FUNCTION	SUBROUTINE
Visual C++	function	(void) function
Visual Basic	Function	Sub
Delphi	Function	Procedure

表 2 不同语言所对应的整型和实型

Tab.2 Corresponding integer and real in different languages

Fortran	Visual C++	Visual Basic	Delphi
INTEGER(2)	short	Integer	SmallInt
INTEGER(4),INTEGER	int,long	Long	LongInt,Integer
REAL(4),REAL	float	Single	Single
REAL(8)	double	Double	Double,Real

2 调用约定的匹配

调用约定决定了程序对过程如何调用,如何传递参数,如何命名目标过程名.在单一语言环境中,过程调用统一采用默认的调用约定,头文件(或Fortran 接口块)保持调用约定在调用程序和

收稿日期:2003-04-29;**修订日期**:2003-06-26

基金项目:国家自然科学基金资助项目(50099620);河南省高校青年骨干教师计划资助项目

作者简介:周振红(1963-),男,山东省蓬莱市人,郑州大学副教授,博士,主要从事GIS 教学与科研.

被调用过程之间的一致性,因此不存在调用约定不一致的问题.但在混合编程中,由于不同语言编写的程序不能分享同一头文件,加之各自的默认调用约定又不完全一致,这就有可能导致程序执行错误的调用,使内存或堆栈出现"腐烂"现象.所以,调用约定的匹配是混合编程所要解决的关键问题.其主要体现在:

(1) 调用程序以怎样的顺序向被调用过程传递参数;

(2) 调用程序和被调用过程由哪一方来完成清理调用后的堆栈;

(3) 是否能进行可选参数的传递;

(4) 参数传递是采取值传递还是引用传递;

(5) 如何命名目标代码中的过程名.

表 3 列出了 Visual Fortran 的调用约定(C、STDCALL 及缺省约定) 及其影响.三个约定统一使用堆栈传递过程参数,且传递方向从左到右,即,当执行过程调用时,调用程序将参数从右到左依次压入堆栈,被调用过程在接收参数时,将参数从左到右依次从堆栈中弹出.

表 3　Visual Fortran 调用约定及其影响

Tab.3　Calling convention and its affecting aspects of Visual Fortran

影响方面	调用约定		
	缺省	C	STDCALL
清理堆栈	Callee	Caller	Callee
参数传递	引用传递	值传递	值传递
可选参数	/	√	/
目标过程名	_name@n	_name	_name@n
目标过程名大小写	大写	小写	小写

调用约定规定了过程中所有参数的传递,而 REFERENCE 和 VALUE 属性则最终决定参数的传递(在任何情况下,数组参数只能以引用方式进行传递).如:

```
Subroutine SumInt(a,b,c)
!DEC$ ATTRIBUTES C::SumInt
!C 约定规定过程SumInt 的参数传递为值传递
!DEC$ ATTRIBUTES VALUE::a,b
!VALUE 属性规定参数a,b 为值传递
!DEC$ ATTRIBUTES REFERENCE::c
!REFERENCE 属性规定参数c 为引用传递
```

Visual Fortran 也提供了限制目标过程名的手段.通过在 Fortran 外部过程中添加别名(ALIAS)属性声明,就可限定目标文件中过程名的具体形式.如:

```
!DEC$ ATTRIBUTES ALIAS:'Sub':SUB
!SUB 为源代码过程名,Sub 为目标过程名
```

Visual C++、Visual Basic 和 Delphi 与 Visual Fortran 相匹配的调用约定列于表 4.

表 4　Fortran 与其它语言相匹配的调用约定

Tab.4　Fortran's calling convention matching other languages' ones

Fortran	Visual C++	Visual Basic	Delphi
缺省	_stdcall	缺省	Stdcall
C	_cdecl	/	Cdecl
STDCALL	_stdcall	缺省	Stdcall

3　建造 Fortran DLL

本文给出一个完整的实例,以说明 Visual Fortran 与其它语言混合编程的实施.该实例在 Visual Fortran 中建造一个 Win32 Dynamic-Link Library 工程,并输出一个实现冒泡排序算法的子程序;然后,Visual Basic、Delphi 和 Visual C++ 分别调用上述过程.

实现冒泡排序算法的子程序为:

```
Subroutine FSort(a,n)
    Implicit None                                    ! 强制类型声明
    !DEC$ ATTRIBUTES STDCALL, DLLEXPORT::FSort       ! 调用约定采用STDCALL
    !DEC$ ATTRIBUTES ALIAS:"FSort"::FSort            ! 限定目标过程名为FSort
    !DEC$ ATTRIBUTES REFERENCE::a                    ! 形参a 为引用传递
    !DEC$ ATTRIBUTES VALUE::n                        ! 形参n 为值传递
    Integer(4)::n,a(n),i,j,t                         ! 声明长整型变量
        DO i=1,n-1
          DO j=i,n
            IF(a(i)>a(j)) THEN
              t=a(i);a(i)=a(j);a(j)=t
            END IF
```

```
        END DO
      END DO
End Subroutine
```

上述过程中,只有声明了DLLEXPORT 属性,才能从DLL 中导出过程,该过程也才能为外部程序所调用;调用约定采用STDCALL,以便其它语言工具以使用Windows API 的方式来调用这里的输出过程;使用ALIAS 属性直接限定产生的目标过程名,以便Visual Basic 和Delphi 应用程序在声明外部过程时可以采取简单的形式.

4 调用DLL 中的过程

4.1 Visual Basic

Visual Basic 6.0本身不能创建标准的Win32 DLL,但它能够使用Win32 DLL.Visual Basic 支持Visual Fortran 中的缺省约定和STDCALL 约定,但在声明外部过程时,不能显式规定所使用的调用约定.外部过程的声明位置可以是标准模块,也可以是窗体对象模块.实例程序采取在窗体中声明Visual Fortran 创建的FDLL 的输出过程FSort:

Private Declare Sub FSort Lib "Fdll.dll" (ByRef arr As Long, ByVal N As Long)

输出过程一经声明,就可以像Visual Basic 自身的过程一样来使用.需要注意的是:

(1) DLL 要么置于系统搜索路径(WINNT\System32),要么以全路径的形式出现;

(2) DLL 入口点(FSort)严格区分字母大小写.声明时,过程名必须和建造DLL 时产生的目标过程名一样;

(3) 缺省情况下,Visual Basic 以引用(ByRef)方式传递参数.若要进行值传递,须在参数前添加ByVal 声明;

(4) 数组形参以引用传递的单个变量进行声明,调用时传递某个数组片段的首元素.如:Call FSort(arr(2),2),这里实际传递的是arr(2) 和arr(3) 二个元素.

4.2 Delphi

Delphi 的调用约定有Register(缺省)、Pascal、Cdecl、Stdcall 和Safecall,能够与Visual Fortran 匹配的调用约定只有Cdecl 和Stdcall,见表4所列.

Delphi 的过程参数包括值参数(缺省)、变量参数(var)、输出参数(out)和常量参数(const).其中,值参数和常量参数以值方式传递;变量参数和输出参数则以引用方式传递.

Delphi 引入动态链接库的方式有两种:静态装载和动态装载,前者需要声明外部过程;后者需要利用Windows API 函数(LoadLibrary、GetProcAddress 和FreeLibrary).实例程序采取静态装载方式,并建立一个接口单元FUnit:

```
unit FUnit;
  interface
    procedure FSort(var a:integer;N:integer);
stdcall;
  implementation
    procedure FSort;external 'FDLL.dll';
end.
```

这里的Stdcall 约定不能省略,否则Delphi 将使用Register 缺省约定,该约定使用寄存器传递参数.当中的FDLL 可以置于Windows 系统路径中.

在主单元的接口部分引用上述接口单元,在主单元的实现部分就可使用FSort 过程.

4.3 Visual C++

Visual C++ 共有三种调用约定:_cdecl(缺省)、_stdcall 及_fastcall(该约定使用CPU 寄存器传递函数参数),能与Fortran 匹配的调用约定见表4.Fortran 与Visual C++ 的混合编程,首先要去掉Visual C++ 对函数名添加的修饰,即,在声明外部过程时,使用C 修饰符(extern "C").另外,Visual C++ 中的缺省参数传递为值传递,若要进行地址传递,可声明引用或指针参数.

为了简便起见,我们在Visual C++ 6.0中创建一个Win32 Console Application 工程,并在主函数前声明如下的函数原型:

```
extern "C" {void _stdcall fsort(int &a,int b);}
```

当Visual C++ 使用C 修饰符(extern "C")限定外部函数时,产生的目标函数名的形式为_FunctionName@n,即在原函数名前后分别添加了单一下划线和参数表所占字节数.这种形式和Visual Fortran 使用STDCALL 约定所产生的目标过程名是基本统一的,惟一的差别是:Visual Fortran 的目标过程名为小写,Visual C++ 的则保持原函数名大小写不变.针对这种情况,我们在建造Fortran DLL 时,将声明的别名(ALIAS)属性去掉;在Visual C++ 中声明对应的小写函数名,这样,两者所产生的目标过程名也就完全统一了.

Visual C ++ 6.0 导入 Visual Fortran 创建的 DLL 有两条途径:一是建立 Visual C ++ 工程对 Fortran DLL 工程的依赖(Dependencies)关系;二是将DLL 连同其LIB 文件一起置于 Visual C ++ 工程路径中,并将 LIB 文件添加到 Project /Settings /Link /Category :General 中的 Object /library modules 编辑框中,这也是本文所采取的途径.

上述将过程fsort 的第一个参数声明为引用,调用时可用数组片段的首元素进行替换;若将其声明为指针,则调用时须用数组片段的首元素的地址进行替换.

5　数组参数的特殊性

在数值计算编程中,数组的利用率是很高的.和其它语言相比,Fortran 90/95 标准中的数组是最具特色和功能最为强大的工具,比如,对数组整体和数组片段的操作,隐式循环操作等.但在Fortran 与其它语言的混合编程中,数组参数比较特殊,需要特别对待.

5.1　数组的存储结构

在存储方式上,Fortran 和 Visual Basic 是按列存放的,而 Visual C ++ 和 Delphi 则按行存放.因此,在传递多维数组参数时,若两种语言的数组存储方式不同,需要在调用前后对数组形状进行调整.如:二行三列数组 A(2×3),其Fortran 数组表示为 A(2,3),与之对应的 C/C ++ 数组为 A[3][2].

不管数组是按列存放还是按行存放,数组在内存中都占据一串连续的存储单元.

5.2　数组参数的传递

在Fortran 中,不管外部过程使用何种调用约定,数组参数只能以引用方式进行传递(即地址传递).Visual C ++、Visual Basic 及 Delphi 声明 Fortran 外部过程原型时,在数组形参对应的位置上,不能声明任何形式的数组,只能声明一同类型的引用变量(或指针变量).调用时,用数组(或数组片段)的第一个元素进行替换(若是指针变量,Visual C ++ 须用代表数组首地址的数组名或某一数组片段首元素的地址进行替换)即可.

尽管不同语言的数组起始下标规定不同(Fortran 的起始下标为 1,Visual C ++ 的为 0,Visual Basic 和 Delphi 则可显式规定起始下标),但由于数组在内存中占据的是一串连续的存储单元,所以只要传递了数组(或数组片段)的首地址,其后续元素就一一被传递.

参考文献:

[1]　周振红,杨国录,周洞汝,等.FORTRAN 与 VISUAL BASIC 混合编程的研究[J].武汉水利电力大学学报,1999,4(2):85～87.

[2]　周振红,颜国红,吴虹娟.Fortran 与 Visual C ++ 混合编程研究[J].武汉大学学报(工学版),2001,4(2):84～87.

[3]　周振红,李端有,谈　戈.Fortran 与 Delphi 混合编程[J].计算机应用,2001,4(4):91～93.

[4]　周振江,张君静,陈崎峰,等.基于 Visual Fortran 的交互视算技术[J].郑州大学学报(工学版),2003,24(1):66～69.

Mixed Language Programming of Visual Fortran Based on Win32 DLL

ZHOU Zhen-hong[1], SONG Yu-wei[1], GUO Heng-liang[1], YANG Guo-lu[2]

(1.College of Environmental & Hydraulic Engineering, Zhengzhou University, Zhengzhou 450002, China; 2.College of Water Resources & Hydropower, Wuhan University, Wuhan 430072, China)

Abstract: Mathematical models programmed by Fortran may be embedded into various comprehensive information systems by means of mixed language programming with Visual Fortran and Visual C ++, Visual Basic, and Delphi. The method is, computing routines are integrated into Win32 DLL in Visual Fortran, and then DLL is imported and routines called in Visual C ++, Visual Basic and Delphi. The key to carrying out mixed language programming, is, calling convention, naming convention, argument type and passing should be consistently reconciled between Visual Fortran and Visual C ++, Visual Basic as well as Delphi, respectively.

Key words: Visual Fortran; DLL; mixed language programming; calling convention; argument passing