文章编号:1671-6833(2008)01-0099-03

Fortran 与 C/C++ 共享模块中的数据和例程

毕苏萍[1], 周振红[2]

(1.郑州大学 土木工程学院,河南 郑州 450001; 2.郑州大学 环境与水利学院,河南 郑州 450001)

摘 要: 多语言的混合编程主要是一种语言编写的程序对另一种语言编写的例程实施调用,所用到的数据主要通过例程参数来传递,当中很少体现出面向对象的程序设计思想.现提出利用 Fortran 90 新引进的模块,实现 Fortran 与 C/C++ 面向对象的混合编程新模式.示例结果表明,C/C++ 既可直接访问 Fortran 模块中的数据和例程;其自身的数据和函数也可封装到模块中,供引用模块的 Fortran 程序单元访问.

关键词: 数值计算;混合编程;调用约定;面向对象;模块

中图分类号: TP 311 **文献标识码**: A

0 引言

模块是 Fortran 90 新引进的一种程序单元,它包含数据、例程(函数和子程序统称为例程)及例程接口等要素的声明,其功能远比 Fortran 77 数据块程序单元来得强大.通常,模块一经别的程序单元引用,当中的数据和例程就可访问了;当然,模块要素的可见性亦可限制在模块内,以提供数据抽象,编写安全、可移植的程序代码[1].从这个意义上说,模块类似于 C++ 中的类,是用来模拟面向对象程序设计[2]主要特性的基础框架.如果能运用模块在 Fortran 与 C/C++ 间交换数据[3]和例程,则会给实际混合计算工程的开发和维护带来极大便利,使两者的混合编程跨上一个新台阶.鉴于此,本文将对 Fortran 与 C/C++ 共享模块中的数据和例程进行系统探讨.

示例程序开发环境为 Compaq Visual Fortran 6.6(Fortran 90/95)/Microsoft Visual C++ 6.0,程序编译采取 Fortran 与 C/C++ 的混合编译[4].

1 模块实体的特殊命名规则

编译时,模块中的实体(数据和例程的统称)与其他外部实体的命名规则不同[5],模块中的实体命名呈现以下形式:

_MODULENAME_mp_ENTITY [@stacksize]

其中,MODULENAME 是模块名,统一被转换成大写;ENTITY 是模块中的实体名,也被转换成大写;mp 是模块名与模块实体名间的分隔符,总采取小写形式.例如:

```
MODULE mymod
      INTEGER a
  CONTAINS
  SUBROUTINE b (j)
        INTEGER j
    END SUBROUTINE
END MODULE
```

该模块在 IA-32 系统下的目标对象文件(.OBJ)中,其 a、b 实体目标名分别为:

_MYMOD_mp_A

_MYMOD_mp_B

在不同的属性编译指令下,上列模块例程 b 的目标例程命名也呈现不同的形式(见表 1 所列).可见,目标模块例程名中的模块名总保持大写,和属性选项(ALIAS 属性除外)无关.

对模块中的数据实体而言,规定属性编译指令没有意义;即便是规定了别名属性选项(Alias),也不会对其目标名产生影响.

因此,当 Fortran 与 C/C++ 共享模块时,可运用 ALIAS 属性选项使模块中的目标例程名呈简单形式;若要模块中的数据目标名呈简单形式,还须使用特殊方法(详见下文).

收稿日期:2007-09-22;**修订日期**:2007-10-30

基金项目:河南省科技攻关项目(0496050905)

作者简介:毕苏萍(1965-),女,河南郑州人,郑州大学高级工程师,主要从事建筑结构检测加固及计算机应用.

表1 不同属性编译指令下的模块例程命名

Tab.1 Naming conventions of module procedures at compilation attributes

属性	选项IA-32系统中的目标模块例程名
None	_MYMOD_mp_B
C	_MYMOD_mp_b
STDCALL	_MYMOD_mp_b@4
ALIAS	覆盖其他选项的命名影响,使目标模块例程保持Alias选项规定的形式

2 C/C++直接访问模块中的数据和例程

现构想这样一个典型示例:学生构造类型由学号和成绩两个成员组成,编写一个函数找出最高分的学生(若最高分有几个学生并列,限定第一个),并将学生平均成绩存放在一全局变量中.要求:算法由Fortran模块例程来实现,主程序由C/C++编写.

下面先给出访问Fortran模块的C语言控制台程序(.c):

```
#include <stdio.h>
extern float Aver;
struct Student{
    int N;
   float Mark;
};
#define STU struct Student
extern STU FindMax(STU s[],int n);
void main(int argc, char * argv[]) {
        STU m,s[3] = {{1001,70},{1002,80},{1003,
90}};
        m = FindMax(s,3);
        printf("%d:%f\n",m.N,m.Mark);
        printf("%f\n",Aver);
}
```

如果是C++控制台程序(.cpp),则需要在外部变量和函数声明中额外添加extern "C"链接选项,以便消除C++编译器给目标名添加的特殊命名修饰.

上列主程序,数据结构包含一全局变量和学生构造类型;定义的函数返回结构体,其结构体数组参数采取地址传递(即传递其第一个成员的地址),第二个参数采取值传递;在C/C++缺省调用约定(__cdecl)下,外部变量目标名为_Aver,外部函数目标名为_FindMax.

据此,给出下列对应的Fortran实现模块(.f90)

```
MODULE Examp
  IMPLICIT NONE
  REAL :: Aver
  common /A/ Aver
  ! DEC$ ATTRIBUTES ALIAS:'_Aver':: A
  TYPE Student
      SEQUENCE
      INTEGER N
      REAL Mark
  END TYPE Student
  CONTAINS
      FUNCTION FindMax(Stud,M)
          ! DEC$ ATTRIBUTES C, ALIAS:'_Find-
Max':: FindMax
          INTEGER M
          TYPE(Student) Stud(M), FINDMAX
          INTEGER :: I, J = 0
          REAL :: T = 0.0,S = 0.0
          DO I = 1, M
            IF(T < Stud(I)%Mark) THEN
              J = Stud(I)%N
              T = Stud(I)%Mark
            ENDIF
          END DO
          FINDMAX % N = J
          FINDMAX % Mark = T
          DO I = 1, M
            S = S + Stud(I).Mark
          END DO
          Aver = S/M
      END FUNCTION
END MODULE Examp
```

(1)由前述模块的特殊命名规则可知,变量Aver的目标名为_EXAMP_mp_AVER,且无法通过规定属性编译指令来改变.为了和C程序中的对应变量目标名一样,此处"偷梁换柱":将变量Aver置于共用区,并规定共用区全局变量的目标名为_Aver.这样,Fortran模块中引用的是变量Aver,而C/C++程序通过同名外部变量/共用区来间接访问该变量.

(2)为了和C/C++结构体相对应,除要求派生类型的成员在次序、数据类型上对应外,还需在声明的成员开头规定SEQUENCE关键字,以使派生类型成员按声明次序存储.

(3)对函数FindMax规定了与C对应函数一致的目标名_FindMax和调用约定C;在C调用约定下,标量参数采取值传递,数组参数仍为引用传递(即传递数组第一个元素的地址).

3　C/C++ 的数据和函数封装到模块中

上述 C/C++ 直接访问 Fortran 模块中的数据和例程;反之,也可将 C/C++ 的数据和函数置于模块(模块充当了 C/C++ 的包装子)中,Fortran 程序通过引用模块来间接访问 C/C++ 的数据和函数.

为了全面反映各类数据在两种语言间的交换,下列示例进行了特别设计:用 C/C++ 实现求两个单精度实型数(构成一结构体)平方根的简单算法,待求平方根的两数由结构体指针/引用参数传入,所求的平方根存放在一全局变量中.

用 C 实现的算法程序(.c)为:

```
#include <math.h>
float R;
struct SQR {
    float a;
    float b; };
void c_sqrt (struct SQR *s) {
    R = (float) sqrt(s->a*s->a + s->b*s->
b); }
```

用 C++ 实现的算法程序(.cpp)为:

```
#include <math.h>
extern "C" float R;
struct SQR{
    float a;
    float b; };
extern "C" void c_sqrt (SQR &s) {
    R = (float) sqrt(s.a*s.a + s.b*s.b); }
```

此处体现了 C++ 与 C 的几个不同点:

(1)C++ 声明结构体变量只使用"类型名",而 C 必须使用"struct + 类型名";

(2)为实现地址传递,C++ 还可使用方便的引用参数,而 C 只能使用指针参数;

(3)编译时,C++ 编译器会给目标实体名添加和编译器版本有关的特殊修饰.通常,使用 extern "C"链接选项将其消除,以便相应的 Fortran 目标实体名与之统一.

用来包装上列 C/C++ 程序的 Fortran 模块(.f90):

```
MODULE Cproc
    REAL res
    common /r/ res
    TYPE SQR
        SEQUENCE
        REAL a
        REAL b
    END TYPE SQR
    INTERFACE
        SUBROUTINE C_Sqrt (s)
            ! DEC$ ATTRIBUTES C :: C_Sqrt
            ! DEC$ ATTRIBUTES REFERENCE :: S
            REAL (s)
            END SUBROUTINE
    END INTERFACE
END MODULE
```

在例程接口块中声明了和 C/C++ 缺省调用约定(__cdecl)匹配的 C 约定,目标例程名被转换成小写,和前列的 C/C++ 函数目标名一致;C 约定下参数传递采取值传递,故此又给参数规定了 REFERENCE 属性,以使其最终采取引用传递(即地址传递);前列 C/C++ 函数参数传递的是结构体第一个成员的地址,所以此处设置了与第一个成员同类型的引用参数.

值得注意的是:在建立 C/C++ 函数接口块时,前面定义的派生类型不可用.这意味着,模块不能用来包装带有结构体值参数的 C/C++ 函数.

通过引用模块(C/C++ 包装子)来间接访问 C/C++ 数据和函数的 Fortran 程序为:

```
PROGRAM Ex_2
    USE Cproc
    IMPLICIT NONE
    TYPE(SQR) X
    x % a = 3.0 ; x % b = 4.0
    CALL C_Sqrt (x%a)
    PRINT *,Res
END PROGRAM
```

4　结语

(1)示例结果表明,C/C++ 既可直接访问 Fortran 模块中的数据和例程;其自身的数据和函数也可封装到模块中,Fortran 通过引用模块包装子来间接访问 C/C++ 的数据和函数.因而,使 Fortran 与 C/C++ 的混合编程跨上了一个新台阶.

(2)运用本文提出的借助共用区全局变量的方法,可以避开模块实体的特殊命名规则,使 C/C++ 在命名与 Fortran 共享的数据实体时不再受限制.

(3)利用模块在两种语言间共享例程时,结构体/派生类型可以作为函数的结果;但作为参数时只能采取地址传递;C/C++ 声明结构体指针/引用参数,Fortran 声明与结构体第一个成员同类型的引用参数.

(下转第 109 页)

[3] 郝伏勤,李 群,黄锦辉. 黄河水资源保护科学研究所文献[DB]. 郑州:黄河流域水资源保护局,2005.
[4] 吴 诩,李永乐,胡庆军. 应用数理统计[M]. 长沙:国防科技大学出版社,1995:161 - 186.
[5] 《现代应用数学手册》编委会. 现代应用数学手册:概率统计与随机过程[M]. 北京:清华大学出版社,1999.

The Forecast of Water Quality Based on Artificial Neural Networks and Regression Analysis

LI Yi - fang, CHENG Wan - li, LIU Jian - ting

(College of Mathematics and Information Science, North China Institute of Water Conservancy and Hydroelectric Power, Zhengzhou 450011, China)

Abstract: As to the abnormal phenomenon in the forecast of artificial neural networks, the method, in which the forecast range from the regression analysis model is used to control the abnormal phenomenon, has been adopted. In the forecast of the water quality of Yellow River in Sanmenxia, the average accuracy of the quantity of ammonia and nitrogen before the control of ANN is only 50.05%, which is because the forecast number is very different of the accurate number in June 2006, the relative error of the forecast number reach up to 214.88 percent, beyond the forecast range of regression, in order to have effect on the whole accuracy. The accuracy of this month is 90.08%, the average accuracy reaches up to 80.79%; the whole forecast accuracy is proved obviously. The practice shows that the method is effective to eliminate the abnormal phenomenon in the artificial neural networks.

Key words: regression analysis; artificial neural networks; water quality forecast

(上接第 101 页)

参考文献:

[1] 周振红,郭恒亮,张君静,等. Fortran 90/95 高级程序设计[M]. 郑州:黄河水利出版社,2005.
[2] 王丽娟,孙西超,底松茂,等. 软件复用与基于面向对象框架的软件开发方法[J]. 郑州大学学报:工学版,2003,24(3):24 - 28.
[3] 任 慧,周振红. Fortran 与 C/C ++ 共享公用外部数据[J]. 郑州大学学报:工学版,2007,28(4)63 - 65.
[4] 任 慧,周振红,张成才. Fortran 与 C/C ++ 的混合编译[J]. 计算机工程与设计,2007,28(17):4096 - 4098.
[5] 周振红,徐进军,毕苏萍,等. Intel Visual Fortran 应用程序开发[M]. 郑州:黄河水利出版社,2006.

Fortran and C/C ++ Sharing Data and Routines in Modules

BI Su - ping[1], ZHOU Zhen - hong[2]

(1. School of Civil Engineering, Zhengzhou University, Zhengzhou 450001, China; 2. School of Environment and Water Conservancy, Zhengzhou University, Zhengzhou 450001, China)

Abstract: In mixed - language programming, routines programmed with one language are called with the other, and the data passed through calling argument lists, so no object - oriented programming is embodied in this process. In this paper, based on the module recommended by Fortran 90, a new way of object - oriented mixed - language programming with the two languages is presented. It has been proved from the experiment that not only C/C ++ is able to directly access the data and routines in the module, but also C/C ++ data and routines to be encapsulated into the module in order to be accessed by the Fortran unit using the module.

Key words: numerical computation; mixed - language programming; calling convention; object - oriented; module