文章编号:1671 - 6833(2007)04 - 0063 - 03

Fortran 与 C/C++ 共享公用外部数据

任 慧[1],周振红[2]

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

摘 要:在 Fortran 与 C/C++ 的混合编程中,数据主要通过例程参数传递,这使得被调用例程接口的设计趋于复杂化,有时不能直接利用现有的算法例程.现提出另一种数据传递方式,即在这两种语言间共享公用外部数据,包括全局变量和共用区/结构体.示例结果表明,只要采取适当措施,Fortran 与 C/C++ 共享公用外部数据是完全可行的,从而为这两种语言间传递数据提供了一条新途径.

关键词:数值计算;混合编程;公用外部数据;共用区;结构体

中图分类号:TP 311 **文献标识码**:A

0 引言

此前,数值计算混合编程[1]主要是一种语言编写的程序对另一种语言编写的例程(函数和子程序统称为例程)实施调用,所用到的数据主要通过例程参数来传递.其不足之处是:数据传递方式单一,被调用例程的接口设计趋于复杂化,有时不能直接利用现有的算法例程.

事实上,Fortran 与 C/C++ 同属编译型语言,在语法规则、数据结构等许多方面存在相似之处,其数据传递除通过例程参数进行外,还可通过公用外部变量来进行.公用外部变量包括全局变量或外部变量、Fortran 共用区及 C/C++ 结构体,所有这些变量都可在声明的程序之外进行访问.

笔者将对 Fortran 与 C/C++ 共享公用外部数据(变量)进行较为系统的探讨.示例程序环境采用当前数值计算常用的环境:算法语言为 Fortran 90/95[2] 和 C/C++,开发环境为 Compaq Visual Fortran 6.6/Microsoft Visual C/C++ 6.0,程序编译采取 Fortran 与 C/C++ 的混合编译[3].

1 共享全局变量或外部变量

此处的全局变量或外部变量,系指在一种语言中声明为全局的变量,在另一种语言中作为外部变量来访问.

1.1 Fortran 访问 C/C++ 外部变量

在 C 语言程序中,全局变量须在程序源文件开头、所有函数之外进行声明.下列为仅包含一整型数组全局变量的 C 示例程序(.c):

```
int IDATA[3] = {1,2,3};
```

对应的 C++ 示例程序(.cpp):

```
extern "C" {int IDATA[3] = {1,2,3};}
```

C++ 的命名规则不只是针对函数,对全局变量也产生影响.故此,程序使用了 extern "C"链接选项,以消除 C++ 编译器产生的特定命名修饰.

在缺省调用约定下,Fortran 编译器将目标实体名统一转换成大写.因此,这里的 C/C++ 程序采用大写命名全局变量,以便和 Fortran 引用的目标外部变量名保持一致.

访问上列变量的 Fortran 控制台程序(.f90):

```
program Ex_1
    implicit none
    !DEC$ ATTRIBUTES EXTERN :: idata
    INTEGER idata (3)
    PRINT *,idata
end program Ex_1
```

此处,EXTERN 属性用来通知编译器该变量为外部变量,其声明在别的源文件中进行.

1.2 C/C++ 访问 Fortran 外部变量

Fortran 77 只提供了全局的共用区,共用区包含的变量在引用共用区的程序单元中可访问.

可以将访问 Fortran 全局变量的 C/C++ 程序看着是 Fortran 程序单元,因此,可将单个 Fortran 变量置于共用区,C/C++ 程序通过访问共用区来

收稿日期:2007 - 09 - 10;修订日期:2007 - 10 - 10

作者简介:任 慧(1965 -),女,郑州大学讲师,主要从事电子技术、计算机应用方面的研究.

间接访问其包含的变量.

包含共用区的 Fortran 示例程序(.f90):

```
block data
        implicit none
        real Pi
        common /Pi/ Pi
        data Pi /10.0/
end
```

值得注意的是:数据块中,给变量和存放变量的共用区取了相同的名称.

访问上述变量的 C ++ 控制台程序(.cpp):

```
#include < stdio. h >
extern "C" float PI;
void main( int argc, char * argv[ ])
{ printf("PI = %f",PI); }
```

此处,必须将链接选项(extern "C")声明置于全局范围,这是 C++ 语法规则所要求的. C 程序(.c)既可将链接选项声明置于全局范围,也可置于函数局部范围.

值得一提的是:程序声明的外部变量 PI 为 Fortran 共用区名,而不是共用区中的变量名. 因此,C/C++ 无法访问 Fortran 无名共用区中的变量.

2 共享共用区/结构体

在 1.2 节中,将单个 Fortran 变量置于共用区,并使共用区和变量取相同的名字,如此,置于共用区的变量可以和 C/C++ 的外部变量进行交换. 显然,若共用区包含多个变量,无法使多个变量都取共用区的名字. 在这种情况下,就需要利用结构体与共用区交换数据.

为了实现结构体与共用区交换数据,必须考虑两者在存储成员变量上的差别. Fortran 以紧凑格式存储共用区成员变量,并遵循下列规则[4]:

- 诸如 BYTE、INTEGER(1)、LOGICAL(1)和 CHARACTER 类型的单个成员变量,紧接着前一个成员变量或数组的地址存储.
- 其他类型的单个成员变量,紧接着前一个成员变量或数组的偶数地址存储.
- 除 CHARACTER 类型外的数组成员,紧接着前一个成员变量或数组的偶数地址存储;CHARACTER 类型的数组成员,总是紧接着前一个成员变量或数组的地址存储.
- 共用区从 4 字节对齐的地址开始存储.

正因为共用区存在上述封装规则,在 C/C++ 结构体与 Fortran 共用区交换数据时,必须确保结构体成员与共用区成员的对齐方式是一致的. 可以考虑在这两种语言中只使用 4 字节和 8 字节数据类型;或者在 C/C++ 程序中使用 pragma pack(2)预处理指令,来硬性规定结构体采取共用区的封装模式.

2.1 C/C++ 直接访问 Fortran 共用区

下列 Fortran 示例程序(.f90),其共用区只包含 4 字节单精度实型数和 8 字节双精度实型数组:

```
block data
    implicit none
    REAL(4) x, y
    REAL(8) z(3)
    COMMON / Really / x, y, z
    !DEC$ATTRIBUTES ALIAS:'Really'::Really
    data x, y, z /1.0,2.0,10.0,20.0,30.0/
end
```

此处,对数据块中的共用区规定了 ALIAS 属性,以使共用区目标名和 C/C++ 结构体变量的混合大小写目标名保持一致.

访问共用区的 C++ 控制台程序(.cpp):

```
#include < stdio. h >
extern "C" struct {
    float x, y;
    double z[3];
  } Really;
void main( void) {
printf("x = %f,y = %f\n",Really. x,Really. y);
printf("z[0] = %f,z[1] = %f,z[2] = %f\n",Really. z[0],Really. z[1],Really. z[2]);}
```

上列 Fortran 共用区成员只使用了 4 字节、8 字节变量,可以和 C/C++ 结构体成员自然对齐. 在这种情况下,pragma pack(2)预处理指令是可选的. 但若规定了 pragma pack(2)预处理指令,则必须配套规定 pragma pack( ),以使结构体恢复先前的封装模式.

2.2 Fortran 间接访问 C/C++ 结构体

既然可以采取措施使结构体和共用区保持相同的封装模式,那么就可以像传递数组参数那样:通过传递结构体/共用区第一个成员的地址,来间接访问结构体/共用区中的每一个成员.

在下列 C++ 示例程序中,函数 initcb 设置了一结构体引用参数. 引用参数和指针参数作用相同,都是传递参数的地址. 结构体引用参数传递的是结构体首地址,即结构体第一个成员的地址.

```
#pragma pack(2)
  struct block_type
  { double x;
    int n;};
#pragma pack()
extern "C" void initcb( block_type & block_hed ) {
    block_hed.n = 1;
    block_hed.x = 10.0;}
```

调用上列函数的 Fortran 控制台程序为：

```
program Ex_2
  implicit none
  INTERFACE
      SUBROUTINE initcb (BLOCK)
        !DEC$ ATTRIBUTES C :: initcb
!DEC$ ATTRIBUTES REFERENCE :: BLOCK
        REAL(8) BLOCK
        END SUBROUTINE
  END INTERFACE
  INTEGER(4) n
  REAL(8) x
  COMMON / / x,n
  CALL initcb( x )
  PRINT *,"n = ",n,",x = ",x
end program Ex_2
```

此处，共用区取什么名称无关紧要，甚至可以是无名共用区，关键要使共用区的成员与结构体的成员在顺序和数据类型上一一对应.

在建立与 C ++ 函数对应的例程接口块时，规定了 C 调用约定，以便与 C ++ 的缺省调用约定(__cdecl)相匹配；在 C 调用约定下，目标例程名统一转换成小写，因此在 C ++ 程序中特意用小写命名函数；在 C 调用约定下，参数传递采取值传递，因此给参数 BLOCK 规定了引用属性，以使其与 C ++ 函数引用参数的传递方式保持一致；尽管在 32 位系统下地址编码为 4 字节长整型数，但不同类型变量的存储单元大小是不同的，所以 BLOCK 声明为双精度实型数，以便和结构体第一个成员的数据类型相对应.

3　结论

共享公用外部数据，为 Fortran 与 C/C++ 间的数据传递提供了一条新途径. 单个 Fortran 变量置于同名共用区，借此可以实现与 C/C++ 全局变量的数据交换. 通过采取适当措施，使共用区和结构体保持相同的封装模式，就能够在两种语言间共享共用区/结构体.

参考文献：

[1] 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，2001，34(2)：84 - 87.

[2] 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.

[3] 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、4111.

[4] 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 Common External Data

REN Hui[1], ZHOU Zhen - hong[2]

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

Abstract: In mixed - language programming with Fortran and C/C++, data are mainly passed through calling argument lists, resulting in routine interfaces becoming complex and making it hard to use directly some routines at hand. In the paper, a new way of data passing is presented, namely Fortran and C/C ++ sharing common external data, including global variables and common blocks/ structures. It has been proved by the experiment to be workable only of appropriate measures are taken, so it is a new method in which data are passed between Fortran and C/C++.

Key words: numerical computation; mixed - language programming; common external data; common block; structure