文章编号:1671 - 6833(2011)01 - 0116 - 05

Fortran 与 VB 传递单一字符串的几种方式

周振红[1], 王国宾[1], 毕苏萍[2]

(1. 郑州大学 水利与环境学院,河南 郑州 450001;2. 郑州大学 土木工程学院,河南 郑州 450001)

摘 要: 为使 Fortran 与 VB 混合计算工程中的接口容纳更丰富的数据类型,研究了以内建类型和 OLE 变体类型为载体的单一字符串数据的传递.在阐述了字符串于两种语言中的表达、存储及参数传递特性的基础上,成功实施了含长度和不含长度字符串的传递,根据 CVF 字符串型函数调用机制,提出了 VB 对其调用的变通手段,通过挖掘 CVF 语言扩展功能,解决了字符串指针的传递;基于对 OLE 变体类型结构的本质认识,并利用 CVF 对 Unicode BSTR 的支持,给出了字符串变体类型的传递方式.结合实例,验证了各种传递方法的可行性.

关键词: 混合计算工程;接口;字符串数据;内建类型;OLE 变体类型

中图分类号: TP311 **文献标志码**: A

0 引言

在混合语言计算工程中,各类数据在 Fortran 与其他语言间的传递是开发面临的基本问题[1-3].之前的探索多限于数字数据的传递,很少涉及普遍意义上的字符串数据的传递.由于字符串在 Fortran 与其他语言中的表达、存储存在多样性,其传递特性及例程(子程序或函数)调用机制又依赖语言环境使用的调用约定,导致字符串数据的传递格外棘手,尤其是在 Fortran 与 Visual Basic(VB)间进行传递.鉴于此,笔者将分别以语言自身拥有的内建类型和语言扩展提供的 OLE 变体类型为载体,就上述问题展开深入系统的探讨,以期对单一字符串在这两种语言间的传递给出完整解决方案.

实例开发环境为 Compaq Visual Fortran (CVF)6.6/Fortran 90[4] 和 VB 6.0. CVF 创建的 Fortran 例程置于动态链接库 DLL[5],供 VB 程序调用.

1 内建类型方式

1.1 字符串的表达与存储

VB 自 4.0 版本开始,使用 BSTR 型的长度前缀字符串[6].BSTR 是由 OLE 自动化或 ActiveX 自动化定义的数据类型.一个 BSTR,是一个指向字符串的指针.例如:

```
Dim str As String
str = "help"
```

"help"的存储如图 1 所示,字符串变量 str 相当于指向"help"的指针,即 BSTR 指针.

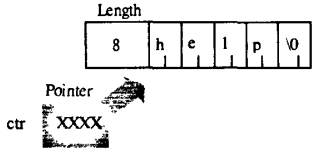

图 1 VB 字符串的存储结构

Fig. 1 Storage structure of VB string

在字符编码上,VB 使用 Unicode 表达字符串,Unicode 是用二个字节表示每个字符的字符编码方案,国际标准组织(ISO)几乎为每种语言的每个字符在 0 到 65535 范围内都定义了一个数字.由于 Win9x 不支持 Unicode,所以 VB 在将字符串传入 API 或 DLL 时,总是将字符串由 Unicode 转换成 ANSI 编码;当接收来自 API 或 DLL 的字符串时,又将字符串由 ANSI 编码转换成 Unicode.

Fortran 字符串使用 ANSI 编码表达,但其每个字符用一个字节存储,且末尾没有结尾字符'\0'.所以,就字符串的字符编码而言,CVF 创建的含字符串参数的 DLL 与 VB 要求的是一致的.

1.2 字符串参数的传递特性

VB 的参数传递有传值方式(ByVal)和缺省

收稿日期:2010 - 07 - 20;修订日期:2010 - 09 - 01

作者简介:周振红(1963 -),男,山东蓬莱人,郑州大学教授,博士,研究方向水利信息技术及计算机应用.

的引用传递(ByRef)二种方式. 若以传值方式传递字符串参数,传递的是字符串本身;若以引用方式传递,则传递的是指向字符串的指针. 不管以何种方式传递,当中的字符串都不包含长度(见图1). 但 CVF 传递的字符串参数可能包含长度也可能不包含长度,取决于属性编译指令[7](见表1).

表 1　CVF 不同编译指令下字符串参数的传递

Tab.1　Passing string arguments with CVF at different compiler directives

参数传递属性	例程约定属性		
	*	C	STDCALL
*	引用传递 (包含长度)	字符串首字符转换为 4 字节整数,并以传值方式传递该整数	字符串首字符转换为 4 字节整数,并以传值方式传递该整数
VALUE	错误	字符串首字符转换为 4 字节整数,并以传值方式传递该整数	字符串首字符转换为 4 字节整数,并以传值方式传递该整数
REFERENCE	引用传递 (可能包含长度)	引用传递 (不含长度)	引用传递 (不含长度)

注：* 指缺省,即不显式规定例程的调用约定或参数的传递方式;表中的缺省约定本质上是标准约定.

值得一提的是,Win32 API 采用标准约定,标准约定也是 VB 调用 API 或 DLL 函数时默认的调用约定.

根据调用约定匹配的原则,可以从表 1 中归纳出二种可行的组合:一是默认情况下 CVF 使用缺省的约定和参数传递方式,传递的字符串参数包含长度(该长度为 4 字节长整型数,紧接字符串地址以传值方式传递);二是显式声明 STDCALL 约定属性和 REFERENCE 参数传递属性,传递的字符串参数不包含长度.

1.3　传递含长度字符串

根据前述字符串参数的传递特性,在默认情况下,CVF 传递的字符串参数包含长度;要求 VB 以传值方式传递字符串和代表字符串长度的长整型二个参数与之对应. 假设 CVF 定义如下的子程序原型或接口:

```
Subroutine ReturnString (StrArg)
    !DEC$ ATTRIBUTES DLLEXPORT, ALIAS : 'ReturnString' :: ReturnString
      Character * ( * ) :: StrArg
End Subroutine
```

其中,DLLEXPORT 为例程导出属性,以导出例程供外部程序调用;ALIAS 为别名属性,以固定生成的目标例程名,使得在 VB 中声明 DLL 例程原型时不必察看 DLL 的导出例程列表,直接使用这里的例程别名;StrArg 为字符串参数,在当前情况下只能声明为假定长度字符串. VB 中一致的 DLL 例程原型为:

```
Declare Sub ReturnString Lib "PSingleStr.dll" (ByVal StrArg As String, ByVal StrLen As Long)
```

调用时,实参字符串须是定长字符串,长整型实参取对应实参字符串的长度值.

1.4　传递不含长度字符串

若显式规定 Fortran 例程的约定为标准约定(STDCALL)、字符串参数的传递方式为引用传递(REFERENCE),那么传递的字符串不含长度(见表 1). 在这种情况下,VB 以传值方式传递字符串本身即可. 假设 CVF 定义如下的子程序原型或接口:

```
Subroutine ReturnString (StrArg)
    !DEC$ ATTRIBUTES STDCALL, DLLEXPORT, ALIAS :'ReturnString' :: ReturnString
    !DEC$ ATTRIBUTES REFERENCE :: StrArg
      Character :: StrArg * 20
End Subroutine
```

其中,字符串参数须声明为固定长度字符串形式. VB 中一致的 DLL 例程原型为:

```
Declare Sub ReturnString Lib "PSingleStr.dll" (ByVal StrArg As String)
```

调用时,实参字符串仍须是定长字符串.

1.5　VB 调用 Fortran 字符串型函数

Fortran 字符串型函数,是指函数的返回结果为 Fortran 字符串. 在缺省约定或标准约定下,CVF 在实现字符串型函数调用时,编译器自动添加两个额外参数到参数列表的开头:第一个参数是一个字符指针,指向函数结果字符串的存储单元;第二个参数是一个无负号整型数,代表函数结果字符串的长度.

根据这一机制,VB 要调用 Fortran 字符串型

函数,完全可以将其按具有额外含长度字符串参数的子程序看待,即:在声明对应的DLL子程序原型时,在参数列表开头添加一个字符串参数和一个长整型参数,二者均以传值方式传递,并按输出参数对待.实例(类似于传递含长度字符串,代码略)的运行结果,验证了这种变换的可行性.

值得注意的是,CVF在定义Fortran字符串型函数接口时,若采用缺省约定,需将函数名声明为假定长度字符串;若显式声明约定属性STDCALL,则需将函数名声明为固定长度字符串.不管是哪种情况,VB中对应的实参字符串都须采取定长字符串.

1.6 传递字符串指针

1.6.1 CVF提供的指针

根据前述字符串参数的传递特性,VB若以引用方式传递,则传递的是指向字符串的指针.与此相一致,Fortran例程中的参数也应声明为字符串指针.不过,Fortran 90的属性指针类似于C++中的引用,不是严格意义上表示地址的指针.对此,CVF语言扩展提供了真实指针,它采取"Pointer(指针变量,目标变量)"的形式声明指针,并通过规定目标变量的数据类型来间接规定指针变量的数据类型,指针变量的值代表目标变量的基地址.要声明字符串指针参数,可采取下列形式:

```
Poimter(p, mystring)
Character * (20) :: mystring
```

p即为指向字符串mystring的指针.

1.6.2 CVF对ANSI BSTR字符串的支持

1.1提到:当VB字符串传入API或DLL时,Unicode被转换成ANSI编码.那么,VB以引用方式传入API或DLL的字符串指针,实质是指向ANSI字符串的指针,即ANSI BSTR指针.

对此,CVF通过DFCOM模块提供了函数SysStringByteLen和SysAllocStringByteLen予以支持.前者用于获取参数指针指向的BSTR字符串的长度字节数,并将该BSTR字符串映射为Fortran 90字符串;后者由Fortran 90字符串构建新的BSTR字符串,并返回指向新建BSTR字符串的指针.假设VB中声明如下的DLL子程序原型:

```
Declare Sub ReturnString Lib "PSingleStr.dll"
(ByRef Str As String)
```

CVF定义的Fortran子程序原型或接口为:

```
Subroutine ReturnString (p)
  Use dfcom  ! 含有SysStringByteLen和SysAllocStringByteLen
  ! DEC $ ATTRIBUTES DLLEXPORT, ALIAS : 'ReturnString' :: ReturnString
  Pointer(p, mystring)
  Character * (20) :: mystring

  Interface  ! 建立SysAllocStringByteLen的接口块
    Integer * 4 Function SysAllocStringByteLen(psz, len)
    !DEC$ ATTRIBUTES ALIAS : '_SysAllocStringByteLen@8' :: SysAllocStringByteLen
    !DEC$ ATTRIBUTES VALUE :: psz, len
    Integer * 4 :: psz, len
    End Function SysAllocStringByteLen
  End Interface
End Subroutine
```

子程序实现体中的关键语句:

```
length = SysStringByteLen(p)
outstring = mystring(1:length) // " + F90字串"
p = SysAllocStringByteLen( loc(outstring), len_trim(outstring) )
```

其功能:提取经由CVF字符串指针传入的ANSI BSTR字符串;改写后,将该指针指向新建的ANSI BSTR字符串,传回VB.

值得注意的是,API函数SysAllocStringByteLen位于Oleauto.h,调用前需建立其接口块,而且它的两个参数均以传值方式传递,编译生成的目标函数名为_SysAllocStringByteLen@8;VB中对应的实参字符串须采取非定长字符串.

2 OLE变体类型方式

2.1 OLE变体类型的结构

OLE自动化或ActiveX自动化、COM组件对象模型[8],广泛使用称之为变体(Variant)的数据类型,它能够表达一种语言中除构造类型外的几乎所有数据类型.变体类型实际上为结构体,其中一个成员用来存储变体元素的类型,另一个成员则用来存储变体元素的值(其余的成员为Microsoft所保留).Windows操作系统下的二次平台几乎都支持OLE自动化、COM,自然也支持OLE变体类型.CVF提供的Fortran 90版变体类型的定义为:

```
TYPE VARIANT
```

```
    SEQUENCE
    INTEGER * 2              VT
    INTEGER * 2              RESERVED1
    INTEGER * 2              RESERVED2
    INTEGER * 2              RESERVED3
  RECORD/VARIANT_UNION/      VU
  END TYPE VARIANT
```

其中：VT 指变体元素的类型；VU 指变体元素的值.

VB 对变体类型的支持是透明的. 声明变量时若不指定具体的数据类型，缺省为变体类型；赋值时，变体元素的类型自动取所赋值数据的具体类型. 换言之，VB 中的变体类型就像一条变色龙，可在不同场合代表不同的数据类型.

2.2　CVF 对 Unicode BSTR 字符串的支持

在 VB 中，若 BSTR 字符串以变体参数作载体，当传入 API 或 DLL 时 BSTR 字符串仍为 Unicode，其字符串变体元素类型为 VT_BSTR，元素值为 PTR_VAL，即字符串变体元素实际为指向 Unicode 字符串的 BSTR 指针.

Fortran 90 字符串采用 ANSI 编码，当操作经由变体参数传入的 Unicode BSTR 字符串时，需要进行 Fortran 90 字符串与 Unicode BSTR 字符串的转换. 对此，CVF 通过 DFCOM 模块提供了转换函数 ConvertBSTRToString 和 ConvertStringToBSTR. 它们的接口为：

```
INTEGER * 4 FUNCTION ConvertBSTRToString( bstr, string)
    USE OLEAUT32
    USE DFNLS
    INTEGER * 4, INTENT( IN )    :: bstr
    CHARACTER * ( * ), INTENT( OUT )    :: string
END FUNCTION ConvertBSTRToString

INTEGER * 4 FUNCTION ConvertStringToBSTR( string)
    USE OLEAUT32
    USE DFNLS
    CHARACTER * ( * ), INTENT( IN )    :: string
END FUNCTION ConvertStringToBSTR
```

其中，ConvertBSTRToString 函数用于将 BSTR 指针参数所指的 Unicode BSTR 字符串转换成 Fortran 90 字符串，函数返回该 Fortran 90 字符串的长度字节数；ConvertStringToBSTR 函数则用于将 Fortran 90 字符串转换成 Unicode BSTR 字符串，函数返回指向该 Unicode BSTR 字符串的 BSTR 指针. 假设 VB 中进行如下的 DLL 子程序原型声明：

```
Declare Sub ReturnString Lib "PSingleStr. dll" (ByRef VarString As Variant)
```

CVF 定义的 Fortran 子程序原型或接口为：

```
Subroutine ReturnString ( VarStr)
  USE DFCOM
  ! DEC$ ATTRIBUTES DLLEXPORT, ALIAS : 'ReturnString' :: ReturnString
  TYPE( VARIANT) :: VarStr ! 声明变体类型参数
End Subroutine
```

子程序实现体中的关键语句：

```
length = ConvertBSTRToString( VarStr % VU % PTR_VAL, f90string1)
f90string2 = f90string1 ( 1 : length) // " + F90 字串"
VarStr % VU % PTR_VAL = ConvertStringToBSTR( f90string2)
```

其功能：将经由变体参数传入的 Unicode BSTR 字符串转换成 Fortran 90 字符串，再将改写后的 Fortran 90 字符串转换成 Unicode BSTR 字符串，并通过该变体参数传回 VB.

值得一提的是，VarStr 参数是双向传递参数，调用时 VB 以字符串赋值变体实参，就隐式规定了变体元素类型为 VT_BSTR，所以 Fortran 例程中可直接对变体参数的元素值(PTR_VAL) 进行访问或赋值，而不必显式规定变体参数的元素类型.

3　结论

针对 Fortran 与 VB 传递单一字符串，笔者提供了内建类型方式和 OLE 变体类型方式二类传递途径. 其中，内建类型方式又给出了传递含长度字符串、不含长度字符串和字符串指针三种方式，并提出了 VB 调用 Fortran 字符串型函数的变通手段. 从而，对单一字符串在这两种语言间的传递给出了完整的解决方案，为字符串数组、含字符串自定义类型的传递及单一字符串在其他语言间的传递提供了借鉴.

参考文献：

[1]　任慧，周振红，张成才. 混合计算工程中复合数据

的传递(Ⅰ)—数组[J]. 武汉大学学报:工学版,2008,41(3):71-76.
[2] 任慧,周振红,张成才. 混合计算工程中复合数据的传递(Ⅱ)—派生类型[J]. 武汉大学学报:工学版,2008,41(4):63-68.
[3] 周振红,毕苏萍,张成才. 混合计算工程中复合数据的传递(Ⅲ)—派生类型内嵌数组[J]. 武汉大学学报:工学版,2009,42(4):482-486.
[4] 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.
[5] 毕苏萍,周振红. Visual Fortran 创建 Win32 API 式的 DLL [J]. 计算机工程与设计,2008,29(18):4868-4871.
[6] Visual Basic[EB/OL](2009-1-15). http://www.microsoft.com/msdn/.
[7] Compaq Visual Fortran[EB/OL](2007-12-17). http://www.compaq.com/fortran/.
[8] 毕苏萍,张军,周振红. CVF 对创建 Fortran COM 组件的支持[J]. 郑州大学学报:工学版,2009,30(2):88-90.

Several Ways to Pass Single String between Fortran and VB

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

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

Abstract: To make an interface contain richer data types in Fortran/VB mixed-language programming computation engineering, it is researched to pass single string by built-in and OLE variant types. On the basis of a discussion of string representation, storage and argument passing for the two languages, strings with a length or not are successfully passed. According to string function calling mechanism of CVF, an alternative approach is proposed to call Fortran string functions with VB. Having taken full advantage of CVF language extension, it is solved to pass a string pointer. Based on recognition of the structure of OLE variant types, the way to pass string variant types is presented with the support of CVF for Unicode BSTR. The feasibility of above-mentioned passing methods has been demonstrated by demo programming.

Key words: mixed-language programming computation engineering; interface; string data; built-in type; OLE variant type

(上接第 106 页)

Research for Interactivity of Smoothed Particle Hydrodynamics Fluid Simulation

TAN Tong-de[1], GAO Zhi-guo[1], ZHAO Hong-ling[1], SHI Qi-bo[1]

(1. School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China)

Abstract: A fluid simulation algorithm of particles based on physical theory of hydrodynamic simulation and using and extending the smooth particle hydrodynamics technique was put forward to enhance the interactivity of virtual water. The interaction problem of virtual water with static entities and dynamic entities in virtual scenes was solved by extending the basic algorithm of SPH (smoothed particle hydrodynamics) and the develoment method of programing was given. Experiments show that the virtual water requirement of real-time rendering was satisfied and the interactive problem of the water with static objects and dynamic objects was also solved by the algorithm.

Key words: fluid simulation; smoothed particle hydrodynamics; static entity; dynamic entity; interactivity