文章编号:1671-6833(2008)04-0061-04

Visual Fortran 调用 Win32 API 函数

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 水利与环境学院,河南 郑州 450001)

摘 要:如何正确调用 Windows 操作系统所提供的 Win32 API 函数来扩展 Visual Fortran 在窗口管理、内存管理、绘图、多线程等方面的功能,是当前数值计算 Windows 应用程序开发经常面临的问题.对此进行较为系统的探讨,包括 Visual Fortran 所提供的 Win32 API 接口模块是如何组织的,当中的接口是如何封装的,如何按接口定义设置一致的字符串、指针和结构体 Fortran 实参等,并用典型示例加以说明,给 Visual Fortran 平台上的 Win32 API 函数调用提供了操作指南.

关键词:数值计算;应用编程接口;接口模块;调用约定;混合编译

中图分类号:TP 311 文献标识码:A

0 引言

Windows 操作系统自带的应用编程接口 API,由一系列 C 语言函数构成,存放在多个动态链接库 DLL 中,提供了窗口管理、内存管理、绘图、多线程、安全及网络方面的高级功能,以支持 Windows 平台上的应用开发工具进行这些方面的功能扩展.

在数值计算工程开发中,通常需要调用 Win32 API 函数,才能提供相对完善的功能.例如,在 Fortran QuickWin 应用类型下开展跟踪式计算、交互视算[1-2],其计算强度大,需要至少设计两个线程:一个线程用于计算,另一线程用于绘图及界面管理;像查询硬盘序列号这样的特殊要求,Visual Fortran 没有提供专门的系统例程(函数和子程序统称为例程),这一切都需要调用 Win32 API 函数来实现.在 Visual Fortran 平台下,究竟如何才能正确调用 Win32 API 函数,是笔者所要探讨的主要问题.

示例开发环境为 Compaq Visual Fortran 6.6 (以下简称 CVF),语言为 Fortran 90/95[3].

1 CVF 提供的 Win32 API 接口模块

与 Visual Basic、Visual C++的情况不同,Visual Fortran 提供有专门的 Win32 API 接口模块.在 CVF 的系统目录\Include 下,存放有 GDI32、KERNEL32、USER32 等程序源文件(.F90)及对应的模块可执行文件(.MOD).

1.1 Win32 API 接口模块集合 DFWIN

C 语言调用 Win32 API 函数,需要包含 Windows.h 头文件;CVF 调用 Win32 API 函数,则需要引用 DFWIN 系统模块.该模块是 CVF 提供的 Win32 API 接口模块集合,见下列 DFWIN 系统模块定义所示:

```
MODULE DFWIN
    ……
    use dfwbase
    use gdi32
    use kernel32
    ……
END MODULE DFWIN
```

当中,除了 32 位系统下的例程接口模块 dfwbase 外,其他的模块均为与 Win32 API 的 DLL 同名的接口模块.

清楚了 API 接口模块集合 DFWIN 的构成,在实际开发中就可以选择引用具体的 API 接口模块,链接时只链接必要的库,以使生成的执行文件尽可能小.例如,要调用 API 函数 GetSystemTime,可只引用其所在的 KERNEL32 接口模块:USE KERNEL32.

1.2 Win32 API 图形函数的重命名

在 CVF 中开展数值计算,常选择 QuickWin

收稿日期:2008-08-15;修订日期:2008-09-25

基金项目:河南省自然科学基金资助项目(0311011600)

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.

应用程序类型(使用的库为 DFLIB). QuickWin 除提供了 MDI 窗口风格的图形界面外,还提供了一系列简易的图形例程.

这些图形例程来自于 Win32 API 中的图形设备接口 GDI32,但经过了 CVF 包装,使用起来更加方便. 包装时,GDI 函数中的设备环境 DC 参数被转变成焦点窗口的单元号. 假如在一个工程中既引用了 QuickWin 的 DFLIB 系统模块,又直接或间接地引用了 Win32 API 的图形设备接口模块 GDI32,那么在调用图形例程时就有可能出现冲突.

为了解决上述潜在冲突,QuickWin 的一部分图形例程名增加了"QQ"后缀,以区别于同名的 GDI 函数;另一部分 QuickWin 图形例程名仍沿用了 GDI 的函数名,如果在 QuickWin 中需要调用这部分 GDI 函数,则须在 GDI 函数名上增加"MSFWIN$"前缀. 例如,Rectangle 既是 QuickWin 的图形例程又是 GDI 的图形函数,调用其 GDI 形式时须使用函数名"MSFWIN$Rectangle". 在 QuickWin 中,需要重命名的图形例程见表 1 所列.

表 1 QuickWin 重命名的 Win32 API 图形函数

Tab. 1 Renamed Win32 API graphical routines in QuickWin

QuickWin 例程	重命名的 Win32 API 函数
ARC	MSFWIN$Arc
ELLIPSE	MSFWIN$Ellipse
FLOODFILL	MSFWIN$FloodFill
GETBKCOLOR	MSFWIN$GetBkColor
GETPIXEL	MSFWIN$GetPixel
GETTEXTCOLOR	MSFWIN$GetTextColor
LINETO	MSFWIN$LineTo
PIE	MSFWIN$Pie
POLYGON	MSFWIN$Polygon
RECTANGLE	MSFWIN$Rectangle
SETBKCOLOR	MSFWIN$SetBkColor
SETPIXEL	MSFWIN$SetPixel
SETTEXTCOLOR	MSFWIN$SetTextColor

2 通过接口模块调用 Win32 API 函数

由 C 语言编写的 Win32 API 函数,可以看成是 Fortran 程序的外部例程. Fortran 程序调用外部例程时,通常需要建立外部例程的接口块,以使编译器产生正确的调用. 为此,CVF 提供了上述 Win32 API 接口模块.

2.1 Win32 API 的接口定义

在 CVF 提供的 Win32 API 接口模块中,接口定义都呈现大致相同的形式. 以 API 函数 SetWindowText 为例,该函数用来设置窗口的标题,它在 USER32 接口模块中的接口定义为:

```
INTERFACE
    FUNCTION SetWindowText( hWnd,lpString)
      USE DFWINTY
      integer(BOOL) :: SetWindowText ! BOOL
        ! DEC$ATTRIBUTES DEFAULT, STDCALL, DECORATE, ALIAS:'SetWindowTextA' ::
          SetWindowText
        integer(HANDLE) hWnd ! HWND hWnd
        ! DEC$ ATTRIBUTES REFERENCE, ALLOW_NULL :: lpString
        character * ( * ) lpString ! LPCSTR lpString
    END FUNCTION
END INTERFACE
```

其中:

(1) INTERFACE 为 Fortran 90/95 接口定义的关键字.

(2) DFWINTY 为 Win32 常量、数据类型的声明模块.

(3)! DEC$ ATTRIBUTES,通过属性规定例程、参数的编译选项:

(a) DEFAULT,忽略对例程命名约定、调用约定产生影响的编译选项;

(b) STDCALL,规定调用约定为 STDCALL;

(c) ALIAS,将编译产生的目标例程名限定为引号内的名称,以便与 Win32 API 函数编译产生的目标函数名相统一;

(d) DECORATE,和 ALIAS 配套使用,根据采用的调用约定,使目标例程名的首尾添加相应的修饰;

(e) REFERENCE,覆盖调用约定对参数传递产生的影响,使参数采取引用方式(即传址方式)传递;

(f) ALLOW_NULL,允许引用参数传递空指针.

值得注意的是:Win32 API 接口的调用约定采用了 STDCALL,尽管该约定和 CVF 的缺省约定相匹配,但在该约定下 CVF 的例程参数统一按值方式传递(数组参数仍采取引用传递),而不是

Fortran 习惯上的引用传递. 所以,在查看 Win32 API 的接口定义时,要注意参数是否有 REFERENCE 选项声明,若没有,则参数的传递方式为值传递.

2.2 设置与 API 函数虚参一致的 Fortran 实参

在此,着重探讨三类典型的 Win32 API 函数参数:字符串、指针和结构体.

2.2.1 字符串参数

在 C 语言中,有字符串常量但没有字符串变量,通常用字符指针指向字符串,其字符指针参数采取地址传递;在结构上,C 字符串常量以空格('\0')结尾.

在缺省约定下,CVF 以引用方式传递隐含长度的字符串参数,这个隐含长度是 4 字节的无负号整型数,紧接字符串地址以值方式传递. 通过规定属性编译指令,可以改变字符串的这种默认传递方式[4]. 例如:在上列 API 接口中,对函数 SetWindowText 规定了 STDCALL 调用约定,对其中的字符串参数 lpString 规定了 REFERENCE(引用)传递方式,在这样的编译选项设置下,字符串参数以引用方式传递,且不包含字符串长度.

因此,在上列 API 函数接口定义下,两种语言的字符串参数在传递方式上是一致的. 至于结构上的差异,可以在给 Fortran 字符串实参变量赋值时加以解决. 下列示例程序通过调用 Win32 API 函数 SetWindowText,将 QuickWin 的框架窗口标题设为"计算和绘图":

```
PROGRAM Ex_1
    USE DFLIB
    USE USER32
    IMPLICIT NONE
    INTEGER(BOOL) iRet
    CHARACTER * 10 Str
    ! Str = "计算和绘图"C          ! 方式一
    Str = "计算和绘图" // CHAR(0) ! 方式二
    iRet = SetWindowText ( GetHwndQQ ( QWIN
$FRAMEWINDOW), Str)
END PROGRAM
```

程序中演示了两种方式:一种是在字符串末尾添加 C 符号,将 Fortran 字符串转换成 C 字符串;另一种是在字符串末尾直接添加空格字符.

2.2.2 指针参数

如果在 Win32 SDK 文档中展示的 API 函数参数为指针,那么在 Win32 API 接口模块的接口定义中大多设置了 REFERENCE 参数选项,使其以引用方式传递(即地址传递);但有一部分指针并没有设置 REFERENCE 参数选项,而是以值方式传递代表地址编码的 4 字节整型数.

比如:查询硬盘序列号需要调用 API 函数 GetVolumeInformation,该函数位于 KERNEL32 接口模块,其第 4 个参数代表硬盘序列号,该参数在 SDK 文档中展示的数据类型为 LPDWORD,代表指向 32 位无负号整型数的指针,但在接口定义中只声明该参数为 4 字节整型,并没有规定 REFERENCE 编译选项.

既然接口定义要求以值方式传递地址编码值,那么在设置 Fortran 实参时,就需要对存储硬盘序列号的整型变量取地址,如下列代码段所示:

```
    ……
integer(4) lpVolumeSerialNumber
    ……
bRC = GetVolumeInformation ( lpszdrivename,
Volume,50, &
        % loc ( lpVolumeSerialNumber), NULL,
NULL, lpszSystemName, 32)
    ……
```

其中,% loc(lpVolumeSerialNumber)是对实参变量 lpVolumeSerialNumber 取地址.

调用时,尽管该参数的地址编码值不发生改变,但对应存储单元内的数据可以改变. 如果不对该实参变量取地址,那么以值方式传递的是存储单元内的数据,即实参变量本身;函数调用后,该实参变量的值不发生改变.

2.2.3 结构体/派生类型参数

Win32 API 函数中还有一类典型的参数 —结构体. 在 C 语言中,结构体按其成员的声明次序存储;结构体参数采取地址传递,且传递的是其首地址(即第一个成员的地址). 对此,CVF 在类型定义模块 DFWINTY 中使用了 SEQUENCE 关键字来声明派生类型/结构体,保证了派生类型按其成员的声明次序存储;在 Win32 API 接口模块中使用了 REFERENCE 编译选项来声明派生类型参数,保证了派生类型参数按引用方式(即传址方式)传递,从而使 Fortran 派生类型实参与 C 语言的结构体虚参相一致.

这样一来,若要调用带结构体参数的 Win32 API 函数,只需根据 Win32 SDK 文档中的结构体名在 DFWINTY 模块中查找相对应的派生类型定义(派生类型名在相应的结构体名前加了 T_ 前缀),据此声明和使用派生类型实参变量.

例如,Win32 API 函数 GetSystemTime 的唯一参数为 SYSTEMTIME 类型的结构体,其对应的派生类型为 T_SYSTEMTIME. 下列为调用 GetSystemTime 函数的示例程序:

```
PROGRAM Ex_2
    USE KERNEL32

    IMPLICIT NONE
    TYPE (T_SYSTEMTIME) MYTIME

    CALL GetSystemTime(MYTIME)
    PRINT *,' Current UTC time hour and minute:', Mytime.wHour, Mytime.wMinute
END PROGRAM
```

2.3 Fortran 程序与 API 函数的混合编译

通过前面的分析,我们知道:CVF 提供的 Win32 API 接口模块,相当于 Windows.h 的 Fortran 版.

通过接口模块调用 Win32 API 函数的 CVF 工程,编译时,将生成与接口定义规定的 API 函数原型一致的目标函数;链接时,根据目标函数链接系统路径下对应的 Win32 API 的 DLL 库. 这里,实际上采取了 Fortran 程序与 C 语言 Win32 API 函数的混合编译[5].

由文献[5]可知:按接口模块中的接口定义规定一致的字符串、指针、结构体/派生类型等 Fortran 实参,进而进行相应的例程调用,可以保证 Fortran 程序与 C 语言 Win32 API 函数在堆栈管理、目标例程命名、参数的传递方式及其数据类型的对应等约定方面达成一致,从而产生正确的例程调用.

3 结束语

为 Visual Fortran 平台上的 Win32 API 函数调用提供了操作指南,为数值计算 Windows 应用程序开发扫除了障碍. 实际开发中,根据 Win32 SDK 文档所展示的 API 函数接口,查看对应的 Win32 API 接口模块中的接口定义,明确例程的实现机制(函数或子程序)、参数的数据类型、存储结构及其传递方式等接口信息;调用时,则须依据 Win32 API 函数的接口信息,设置一致的 Fortran 实参. 当中,尤其要注意 API 指针参数以值方式传递地址编码的情况.

参考文献:

[1] 周振红,杨国录,周洞汝. 基于组件的水力数值模拟可视化系统[J]. 水科学进展,2002,13(1):9-13.

[2] 周振红,张君静,陈峙峰,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.

[3] 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.

[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.

[5] 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.

How to Call Win32 API functions in Visual Fortran

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

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

Abstract: Up to now, it has been a problem how to correctly call Win32 API functions with Visual Fortran expanding its functionality of window management, memory management, graphics support, threading, etc., in the development of numerical computation Win32-based applications. A thorough discussion on Win32 API being called in Visual Fortran is offered, how Win32 API is organized, how its interface encapsulated, and how strings, pointer and structure arguments of Win32 API set in Fortran, etc., and typical examples are presented. As a result, a practical guideline is provided for calling Win32 API functions with Visual Fortran.

Key words: numerical computation; Application Programming Interface; interface module; calling convention; mixed-language compiling