使用 Delphi 访问 NetWare 资源

吴秀清　陈晓辉　（合肥中国科学技术大学　230027）

摘要：本文详细介绍了在 Windows 环境中利用 Delphi 的动态链接库访问 NetWare 资源的方法和注意事项，并给出了一个实例进行说明。

关键词：NetWare　Delphi　DLL

在 Windows 环境下工作的应用程序常常需要访问 NetWare 资源，例如向服务器注册、读取用户权限、存储用户文件等。

Novell 公司为了增强 NetWare 操作系统的竞争力，开发了 NetWare Client SDK 作为 NetWare 工作站程序开发工具，从而使在 Windows 环境下访问 NetWare 资源得以简化。NetWare Client SDK 提供了在 DOS/Windows/OS/2 环境下利用 C 语言进行网络开发所需的各种头文件、库函数以及实例。此外在 Windows 环境中，NetWare Client SDK 还提供了动态链接库（.DLL）供应用程序调用。而在 Windows 程序开发工具中，Delphi 以其编程方便、功能强大的优点而成为 96 年度最受用户欢迎的工具之一，同时它可以非常方便地调用动态链接库。这样，只要知道了 Novell 公司提供的 SDK 中的 API 函数和该函数所在的动态链接库，就可以很方便地使用 Delphi 与 NetWare 进行接口，进而访问 NetWare 资源。在使用 Delphi 调用 DLL 时，应熟悉以下几方面的内容。

1. Delphi 中的动态链接库

动态链接库（DLL）是允许数个 Windows 应用程序分享的程序代码和资源。DLL 是一个可执行模块，它包含了能被共享的程序代码和资源。在观念上一个 DLL 如同一个模块，这二者都可以用过程和函数方式为程序提供服务。其根本区别在于：使用一个来自程序单元中的过程或函数时，代码是静态地链接（拷贝）到这个程序的执行文件中的；而 DLL 则是程序运行过程中动态地链入的，效率明显优于前者（在有重复使用情形下）。

一个 DLL 的程序代码和资源存在于以 .DLL 为扩展文件名的独立可执行文件中，当客户端程序运行时，这个文件必须存在。在程序中对过程和函数的调用会动态地链接到它们的 DLL 中的入口。注意 DLL 只能输出过程和函数。由于被 Object Pascal 应用程序使用的 DLL 并不需要用类 Pascal 语言编写，故而 DLL 在使用多种语言混合编程是尤为方便。

要让自己编写的 Delphi 模块使用 DLL 中的过程和函数，这个模块必须用一个外部说明来输入过程和函数。如下面的说明从一个名为 MY-DLL. DLL（自编的 DLL）中输入一个称为 HI-PRI 的函数：

```
Funtion HI-PRI(Context:String):Integer;far;
external'MY-DLL'Index 1;
```

在输入输出的过程和函数中，external 命令取代了原先会出现的说明和语句部分。输入的过程和函数必须由 far 过程命令或一个{$F+}编译器命令来使用远地址调用模型，除此之外 DLL 和其他正常的过程和函数是相同的。

在 delphi 中可以使用三种方法输入 DLL 中的过程和函数：

(1)凭借函数（或过程）的名字输入；

(2)凭借函数（或过程）新的名字输入；

(3)凭借函数（或过程）的序号输入。

(2)的实现由关键字 name 完成；(3)的实现由关键字 index 完成；name 和 index 均未出现时，即为(1)中的实现方式。这三种方法的 external 命令格式分别如下例所示：

```
Funtion HI-PRI(context:String):Integer;
    external'MY-DLL';
{用过程或函数的名字 Improt}
Funtion HI-PRI(context:String):Integer;
    external'MY-DLL'name 'HE-PRI';
{用过程或函数的新的名字 Import}
Funtion HI-PRI(context:String):Integer;
    external'MY-DLL';index 3;
{用过程或函数的在 DLL 中的序号 Import}
```

{用序号输入能有效地减少模块的装载时间,因为无需在 DLL 的检索表中搜索}

上述关于过程和函数的说明可以直接放在使用它们的程序中。但 Delphi 的模块化形式提供了另一更为规范的方法:使用输入程序单元。这个输入程序单元说明一个 DLL 中所有的过程和函数,以及任何需要和这个 DLL 做接口的常数和数据类型。如同 Delphi 自身提供的 WInType 和 WinCrt 程序单元一样,这些输入程序单元并不是与 DLL 接口的必要对象,但是它们可以简化使用多个 DLL 的情况。

假如以上用到的 MY-DLL.DLL 中有两个函数 HI-PRI 和 HE—PRI,其中 HE-PRI 需要传递一个日期型的参数。则可以建立如下的一个.DCU 文件。

```
Unit DLL-EXAM;
interface
type
  TDateRec = record
  Day: Integer;
  Month: Integer;
  Year: Integer;
end;
Function HI-PRI(var context: Integer): Integer;
Function HE-PRI(var context: Integer): Integer;
implementation
Function HI-PRI; external 'MY-DLL' index 1;
Function HE-PRI; external 'MY-DLL' index 2;
end.
```

这样,在任何一个要使用 MY-DLL.DLL 的程序中只需加入一个 uses 语句即可:

```
uses DLL-EXAM;
```

2. 调用动态链接库时的参数传递

大家已经看到,在函数声明的参数列表中需要将原函数体中所有的参数填入小括号中,而且要将原来的参数类型一一对应到 Delphi 的数据类型中。在类 Pascal 语言中,函数参数可以是数值、常数、变量或无类型数、指针等。无论采用何种参数形式,最终函数体中建议尽量采用数值参数方式(即最简方式),这样程序的执行效率高且不易出错。当然,运用指针传递参数是很方便的(如果运用得当)而且有时是不可避免的(在 DLL 中参数必须传递指针时)。由于 C 语言和 Pascal 语言的差异,在传递参数时应该做适当的类型转换,否则在 Delphi 中将会出现类型不匹配的错误。

在将 SDK 中的 C 函数中的参数转换为 Delphi 中的类型时,必须确定其参数类型在类 Pascal 语言中的对应关系。

(1)字符串。无论是在 C 语言中还是在 Pascal 语言中。字符串都有定长和不定长之分(在 Pascal 中的开放型数据即对应这里的不定长类型)。两种类型的字符串在传递时都要注意保证串的空间足够,否则在调用 DLL 时将会出现保护性错误。

(2)无符号参数。大多数 NetWare DLL 要求无符号参数,如 BYTE、DWORD、WORD 在 C 中定义为 unsigned char、unsigned int、unsigned long,而对应到 Pascal 中则分别为 Byte、Pointer、Word 类型。

(3)结构变量。C 和 Pascal 中都允许用户自定义结构,由于结构由用户定义,因势而异,必须以指针传递。大多数 NetWare DLL 中含有指向结构的指针参数。

(4)数组变量。C 里数组的大小是固定的,而 Pascal 中允许存在开放型数组(其实际大小由实际的元素数决定),在调用 DLLJ 时,应采用固定长度的 Array 类型。

(5)空指针。有时 DLL 要求传递空指针,此时参数在 Pascal 中应该是 Pointer 类型。

表 1 列出了 C 语言编制的 DLL 中常见数据类型与类 Pascal 语言中数据类型的对应关系。

表 1

Hwindows	C	Delphi
BOOL	int	Boolean
BYTE	unsigned char	Byte
WORD	unsigned int	Word
DWORD	unsigned long	Pointer
LPSTR	char far *	Pstr
ATOM	WORD	Word
HANDLE	WORD	THandle
HWND	HANDLE	THandle
COLORREF	DWORD	Pointer

在 Delphi 中,所有类型的句柄都是 THandle 类型,其种类如表 2 所示。

表 2

HBitMap	BitMap Resouce
HBrush	Brush Drawing tools
HCursor	Cursor Resouce
HDC	Device control (Includeing display control)
HFont	Font Drawing Tools
HIcon	Icon Resouce
HMenu	Menu Resouce
HPalette	Palette Tools
HPen	Pen Drawing Tools
HRgn	Region

3. 程序实例

下面以得到本地工作站网卡的物理地址为例，具体说明如何通过 Delphi 来访问 NetWare SDK 中的动态链接库，访问 NetWare 资源。为了使程序能够正常运行，工作站上应装有 NetWare 3.12 以上版本提供的 NetWare DOS Requster 软件，它提供了我们将要用到的 NWCALLS.DLL 文件。

```
{输出程序单元}
Unit DLL-DECLARE;
interface
Function NWCallsInit(var in, out):Integer;
Function NWGetConnectionHandle(var ServerName: Stuing;
        var Reserved1: Integer; var ConnectionHandle: THandle;
    var ReServed2:Integer):THandle;
Function NWGetConectNumber(var ConnectionHandle: THandle;
    var ConnectionNumber:THandle;):Word;
Function NWGetInternetAddress(var ConnectionHandle: Thandle;
    var ConnectionNumber:Thandle;
    var NetWorkAddress):Pointer;
end;
implementation
Function NWCallsInit;external'MY-DLL;
Function NWGetConnectionHandle:external'MY-DLL';
Function NWGetConectNumber;external'MY-DLL';
Function NWGetInternetAddress;external'MY-DLL;
end.

{调用执行部分代码}
Unit My-Unit
interface
uses MY-DECLARE
implementation
{$R*.DFM}
procedure TButton1.Click(Sender:TObject);
var
  RInt:Integer;
  Rlong:Pointer;
  ConnectionHandle:THandle;
  ConnectionNumber:Pointer;
  ServerName:String;
  Addr:Address;
  {Address 为网际地址结构}
begin
  RInt:=NWCallsInit(0,0);
  {调用 DLL 初始化}
  RInt:=NWGetConnectionHandle("MYSERVER",0,
ConnectionHandle,0);
  {得到"MYSERVER"服务器的连接句柄}
  Rling:=NWGetConnectionNumber(ConnectionHandle,
ConnectionNumber);
  {得到本地工作站的连接号}
  NWGetInternetAddress(ConnectionHandle, Connection-
Number,Addr);
  {得到本地工作站的网际地址}
  {输出所得到的工作站网际地址}
  Labell.Caption:='工作站的网际地址为:'
  Label2.Caption:=Str(Addr.Addr(3));
  Label3.Caption:=Str(Addr.Addr(4));
  Label4.Caption:=Str(Addr.Addr(5));
end.
```

(来稿时间:1997 年 4 月)