动态链接库技术及其在Delphi中的使用

张仁平　周庆忠（重庆 后勤工程学院油料管理工程系 400016）

摘要：采用动态链接库 (dynamic-link library) 技术不仅可以实现代码共享，有利于程序的模块化，而且使应用程序动态地调用链接代码，大大减少了系统消耗，更重要的是隐藏了实现的细节，保护了开发者的利益，并且可以在其他支持动态链接库技术的开发工具中调用，如VB等。本文通过一个例子，力求能浅显易懂地介绍在Delphi中如何创建和使用该技术，让你轻松编写和使用自己的动态链接库程序。

关键词：动态链接库 (DLL)　Delphi

1　什么是DLL，为什么要使用DLL

动态链接是相对于静态链接而言的，我们通常讲的静态链接是指在应用程序的编译期就把需要的过程或函数链接到可执行文件中，成为可执行文件的一部分。当应用程序调入内存时，过程或函数也同时被调入了内存；而动态链接却在应用程序运行期动态地调入链接代码，当退出动态链接库时，动态链接代码同时也被释放。在32位的操作系统中，系统保证内存中只有一个DLL，当DLL被多个进程调用时，通过内存映射文件使得每一个进程得到一份DLL的映像，这样，每一个进程就把DLL看作是自己的代码。这在16位的操作系统中是不可能实现的。我们比较熟悉多个应用程序同时共享一个DLL的最典型例子是Windows95/98系统中的Kernel32.DLL、User32.DLL和Gdi32.DLL，用户的许多操作都用到以上三个基本DLL文件。

我们有太多的理由要使用DLL，其中最重要的理由有三个：一是实现代码和资源的共享，DLL代码不仅可以在所有支持动态链接库技术的Windows应用程序中共享，并且可以在其他支持动态链接库技术的开发工具中调用，如VB、VFP、PowerBuilder等；二是应用程序动态地调用链接代码，当退出动态链接时，其代码同时被释放，大大减少了系统消耗，提高了程序的执行速度；三是隐藏了动态链接代码的实现细节，有效地保护了程序某些关键技术的实现细节，有利于维护开发者的利益。

2　创建自己的DLL

在Delphi中创建自己的DLL是非常方便的，因为它提供了快捷的实现框架，下面介绍创建一个实现显示对话信息、时间转换和显示选择的日期等三个功能的DLL的全部步骤。

(1) 关闭Delphi中打开的所有文件，选择"File＼New"菜单，双击DLL图标，完成了DLL工程文件的基本框架，保存为DLLProject；

(2) 创建一个不含窗体的单元文件和一个含窗体的单元文件，后者给一般工程文件的单元文件没有区别，前者也可通过修改后者得到，但最好用Delphi提供的模板来获得，其方法是选中"File＼New"菜单，双击Unit图标。将它们分别保存为MyDLL.pas和MyDLLForm.pas。前者用于定义调用自己创建的动态链接库(例子中的DLLProject)的接口，后者用于实现DLLProject工程中的单元文件，例子中实现"显示选择的日期"的功能。

(3) 编写DLL代码。DLLProject实现的三种功能：显示对话信息、时间转换和显示选择的日期(含窗体)，分别由三种过程或函数(一般而言，需要输出结果时用函数，反之则用过程，当然所有过程均可用函数替代，只是不如过程方便)。以下分别介绍动态链接主库DLLProject的制作和调用DLLProject接口的定义。

(1) 先介绍接口单元MyDLL.pas的代码如下：

```
unit MyDLL;//
interface
type PMyTime = ^TMyTime; // 定义指针变量PMyTime
TMyTime = record// 定义TMyTime的三个word型数据项
Hour,
minute,
second: word;
end;
procedure MyDialog; StdCall; // 定义"显示对话信息"输出过程
function SecondToTime (MySecond:word;Time: PMyTime):Word; StdCall; //定义时间转换输出函数
implementation
//下面介绍两种引入DLL方式：名称引入和索引号引入。前者需要在DLL中查找该例程的名称，稍微增加了系统消耗，但不用担心DLL的修改；而后者正好相反，虽然减少了DLL的调入时间，却要担心DLL的修改导致了索引号的变化，且也不直观，不便于维护，加上计算机的运行速度已经非常快这一特点，故一般采用第一种方式。
function SecondToTime; external 'DLLProject.DLL' name 'SecondToTime'; //按名称引入DLL函数
procedure MyDialog; external 'DLLProject.DLL' index 2; // 按索引号引入DLL过程 end.
```

注：如果采用按索引号引入，则索引号应与动态链接库的输出部分的顺序号相同，如上面MyDialog的索引号为2，则在DLLProject中输出部分(exports)中，MyDialog排在第2位。

(2) DLL工程文件代码如下：

```
library DLLProject;// 动态链接库工程名
uses SysUtils, MyDLL (接口单元), Classes, Dialogs,
MyDLLForm in 'MyDLLForm.pas' {DLLForm};
procedure MyDialog; StdCall;// 实现显示成功对话信息 begin
```

```
ShowMessage ('Very Good! You Are Successful!')
end;
function SecondToTime (MySecond:word;
MyTime:pMyTime) :Word; StdCall; // 实现时间转换功能函数
begin
With MyTime^ Do// PMyTime是指针型变量，在MyDLL.pas中定义，存放转换后的时间
begin
Hour:=MySecond div 3600;// MySecond为输入的总时间(以秒为单位)，获得小时数
MySecond:=MySecond-Hour*3600;// 除去小时后的秒数
minute:=MySecond div 60;// 获得分钟数
MySecond:=MySecond-minute*60;// 获得秒数
second:=MySecond;//
end;
end;
exports// 输出的过程或函数名，如果按照索引号引入，则注意其先后顺序
SecondToTime, // 实现时间转换函数
MyDialog,// 实现显示"成功"对话信息
ShowCalendar;// 显示今天的日期(含窗体)
end.
```

(3) 在MyDLLForm的窗体上放置一个MonthCalendar1组件，其单元文件的主要代码如下

```
function ShowCalendar (ACaption: String):
Longint; //
begin Application.Handle := AHandle;// 得到调用DLL应用程序的句柄
DLLForm := TDLLForm.Create (Application); // 创建窗体文件
try
DLLForm.Caption := ACaption;//获得窗体文件的标题
DLLForm.ShowModal;
Result := DLLForm.MonthCalendar1.Date; // 选择返回的日期
finally
DLLForm.Free; // 释放DLL
end;
```

(4) 选中"Project\Build DLLProject"菜单，生成了DLLProject.dll，从而完成DLL文件的制作。

3 使用自己的DLL

在Delphi应用程序中使用DLL通常有隐式调用和显式调用两种方式(显示对话信息和时间转换采用隐式调用方式，显示选择的日期采用显式调用方式)。下面分别介绍这两种调用方式。

3.1 调用DLLProject.dll的窗体文件

中放置四个按扭用于完成上面四种功能

四个TEdit组件，用于显示总时间数和经过转换后的时间数；五个TLabel组件，其中Label5用于显示你所选择的日期。如图1所示。

3.2 单元文件的主要代码

(1) 使用DLLProject.DLL接口单元MyDLL.pas，则在implementation部分进行声明：

```
uses MyDLL;
```

在MyDLL.pas中声明了动态链接库中输出的过程和函数。当然，接口单元是可选择的，当用到DLL中的少数例程时，常常不用接口单元，此时需要在implementation部分声明DLL所用到的所有过程和函数，如下所示：

```
function SecondToTime (MySecond:word;Time:
PMyTime):Word; StdCall; external 'DLLProject.dll';
procedure MyDialog; StdCall; external 'DLLProject.dll';
```

由于调用SecondToTime函数时用到一个指针类型数据PMyTime，因此还要声明这个数据

```
type
PMyTime = ^TMyTime;
TMyTime = record
```

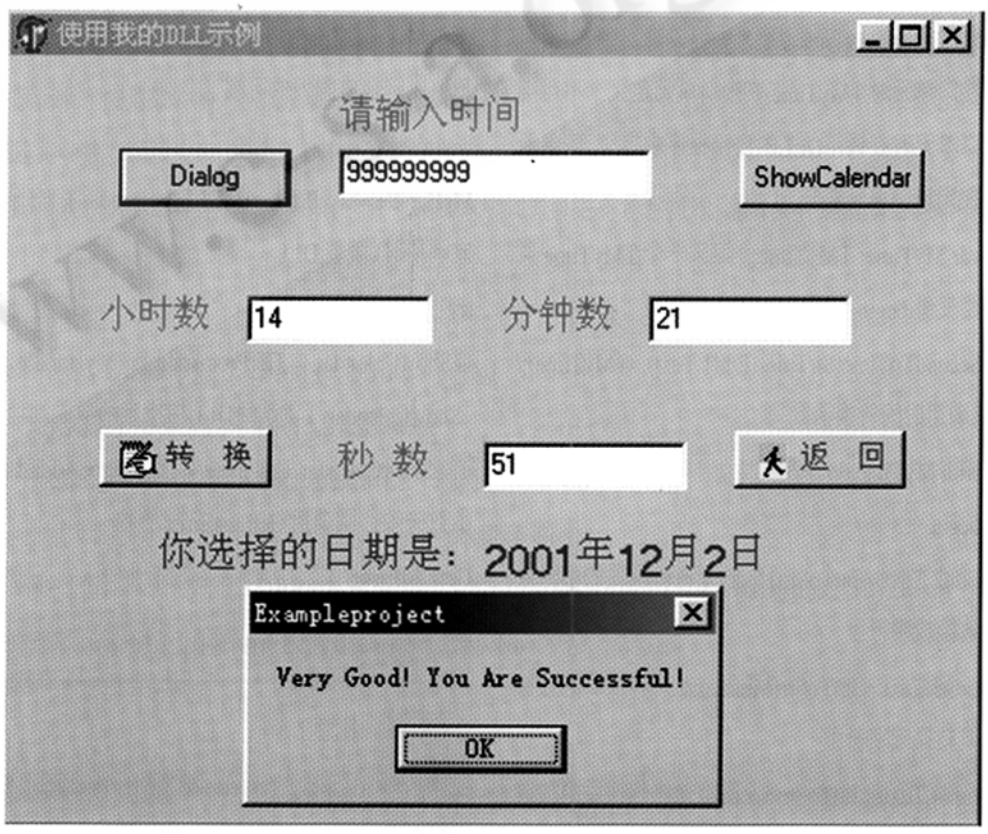

图1 "我的DLL"运行图

```
Hour,
minute,
second: word;
end;
```

(2) 显式调用的声明部分，由上面可知，隐式调用DLL是比较方便和容易的，但当DLL中包含许多例程时，并且大部分例程可能在这个应用程序中用不上，把整个DLL调入内存显然是很浪费的，因而应该采用显式调用方式，采用显式调用方式时需要在Type部分声明调用的函数和过程，声明如下所示：

```
type
TShowCalendar = function (AHandle: THandle;
ACaption: String): TDateTime; StdCall;
```

(3) "Dialog"按扭的代码非常简单，直接调用DLLProject.dll的MyDialog过程，因而其代码仅有一句：

```
MyDialog;// 调用显示 "成功" 信息过程
```

(4) "转 换"按扭主要完成将输入的总时间(以秒为单位)转化为时、分、秒，其代码如下：

```
var
MyTime: TMyTime;//定义函数的输出参数，当用接口单元时TMyTime在接口单元定义，不用接口单元时在调用DLL的应用程序中定义，可参看使用接口单元部分。
var MyTime: TMyTime;// 定义一个TMyTime类型变量 begin
SecondToTime(StrToInt(Edit1.Text), @MyTime);// 调用时间转换函数
with MyTime do
begin
Edit2.Text:= IntToStr(Hour);// 将转换后的小时数转换成字符型
Edit3.Text:= IntToStr(Minute);// 将转换后的分钟数转换成字符型
Edit4.Text:= IntToStr(Second);// 将转换后的秒钟数转换成字符型
end;
```

图2 "选择日期"的MyDLLForm窗体

```
end;
```

(5) "ShowCalendar"按扭主要完成日期的选择和返回所选择的日期，触发该按扭的OnClick事件，将弹出DLLProject工程文件的MyDLLForm窗体，如图2所示，选择完日期后，关闭该窗体，返回选择的日期。该方法采用了DLL的显式调用方式，请注意显式调用方式与隐式调用方式的区别，其代码如下：

```
var
MyHandle:THandle;// 定义应用程序的句柄
ShowCalendar:TShowCalendar;//定义装载的函数或过程 Year,Month,Day:word;//分别存放年、月、日，以符合中国人的习惯
begin
MyHandle:=LoadLibrary ('DLLProject.dll'); //(显式调用时) 装载 DLL
try
@ShowCalendar := GetProcAddress (MyHandle, 'ShowCalendar'); //返回DLL中指定函数的地址
if @ShowCalendar = nil then//如果GetProcAddress函数失败，则调用 GetLastError 函数
GetLastError
else
begin// DecodeDate 将 ShowCalendar 函数返回的日期分解为年、月、日
DecodeDate (ShowCalendar (Application.Handle, 'Hello,This is Today Date'), Year,Month,Day);
Label5.Caption:= '你选择的日期是：' +Concat (IntToStr (Year), '年', IntToStr (Month), '月', IntToStr (Day), '日');
end;
finally FreeLibrary (MyHandle); // 释放 DLLProject.dll
end;
end;
```

以上程序是在Delphi 6.0中调试通过。

4 应注意的几个问题

(1) DLL是可执行文件，但不能被单独运行，只能被别的应用程序调用；

(2) 动态链接库的文件扩展名一般为.dll，也可能是系统文件(.sys)和设备驱动程序(*.drv)等；

(3) 最好避免多线程访问DLL中的全局变量，虽然在32位的操作系统中，一个线程修改了全局变量不会影响其他的线程(因为每个线程都拥有全局变量的一份副本)，但处理不当往往会造成意想不到的结果；

(4) 如果要分发如上所示的DLL文件，则应该同时分发DLLProject.dll和接口单元文件MyDLL.pas，以便该DLL文件能在其他开发环境中使用，当然要对MyDLL.pas转换为相应的开发语言。

参考文献

1 Steve Teixeira & Xavier Pacheco，Delphi4开发大全，人民邮电出版社，1999年8月。

2 Calvert，Delphi4编程技术内幕，机械工业出版社，1999年6月。

3 刘臣勇等，精通Delphi 4.X，清华大学出版社，1999年5月。