VC++动态链接库（DLL）的开发与应用

倪步喜[1]，章丽芙[2]
(1.温州职业技术学院 计算机系，浙江 温州 325035；
2.浙江工贸职业技术学院 电子系，浙江 温州 325003)

[摘 要] 动态链接库（DLL）封装了共享资源和代码，在开发以Windows为平台的应用程序时，使用DLL技术将大大节约内存。使用DLL开发项目，可简化项目管理，提高开发速度。通过实例介绍了隐式和显式链接动态链接库的过程，为工程人员开发和使用DLL提供了一定的技术支持。

[关键词] 动态链接库（DLL）；隐式链接；显式链接
[中图分类号] TP311 [文献标识码] A [文章编号] 1671-4326(2006)01-0034-03

Development and Application of Dynamic Link Library in VC++

NI Bu-xi[1], ZHANG Li-fu[2]
*(1. Computer Science Department, Wenzhoun Vocational & Technical College, Wenzhou, 325035, China
2. Electronic Department, Zhejiang Vocational and Technical College of Industry and Trade, Wenzhou, 325003, China)*

Abstract: Dynamic Link Library (DLL) encapsulates the shared resource and code. In exploring the application based on Windows, the use of DLL technology can greatly save the memory. Using DLL developing project can simplify the project management and speed up the exploration. This paper, introducing the implicit and explicit linking of DLL with instances, provides technical support to technicians in exploring and using DLL.

Key words: Dynamic Link Library (DLL); Implicit linking; Explicit linking

0 引 言

自从微软推出Windows操作系统的第一个版本以来，动态链接库（DLL）一直是Windows操作系统的基础，它既是多个进程共享资源的主要方式，也是操作系统向应用程序提供系统服务的重要手段。DLL通常是包含了若干函数、类和资源的库文件，它不能直接运行，也不能接收消息，但可以被其它执行文件和其它DLL文件动态调用。使用DLL有许多优点，首先最主要的是当多个应用程序通过动态链接的方式共享一个DLL文件时，实现了资源共享，大大缩小了应用程序的执行代码，能更加有效地利用内存。其次是DLL作为应用程序的独立模块，在修改了DLL后，应用程序本身可不作重新编译，许多应用软件可以通过改写DLL文件来增强应用程序的功能。使用DLL开发项目，可简化项目管理，提高开发速度。

1 动态链接库（DLL）的创建

在VC++开发环境中，可以在新建对话框的工程选项卡中选“Win32 Dynamic-Link Library”命令建立DLL，这种DLL中不能使用MFC基础类库。如果在DLL中需要使用MFC基础类库，则可以选择“MFC AppWizard(dll)”命令创建DLL。这两种方法创建DLL有许多相似之处，本文以前者的方法为例来创建DLL。

创建一个DLL，库中包含做加法和减法运算的两个函数，并使用一个基于对话框的应用程序去调用这些函数。应用程序在链接DLL中的函数信息之前，需要从DLL文件中导出函数。导出DLL中的函数有两种方法：一是在定义函数时使用导出函数关键字“_declspec(dllexport)”；二是使用模块定义文件(.def)。

1.1 使用_declspec(dllexport)关键字创建DLL

(1) 头文件（DLL.H)。DLL中函数的申明一般在头文件中进行，这不仅仅是良好的编程习惯问题，在项目的其他开发人员也需要调用DLL中的函数时，尤其需要提供头文件对于给定的例子。其头文件（DLL.H）

[收稿日期] 2005-09-19
[作者简介] 倪步喜(1970—)，男，浙江平阳人，温州职业技术学院计算机系讲师；
章丽芙(1968—)，女，浙江瑞安人，浙江工贸职业技术学院电子系高级讲师.

可定义如下：

```
#ifdef DLL_API
#else
#define DLL_API extern "C" _declspec (dllimport)
#endif
DLL_API int  Add(int a,int b);
DLL_API int  Subtract(int a,int b);
```

其中，extern “C”的作用是为使函数在导出后，函数名保持不变。这是因为C++ 编译器为了支持函数的重载，在编译时会改变函数的名字的缘故。

（2）函数的实现文件（DLL.CPP)。在函数的实现文件中，为了指明DLL 中的导出函数信息，需要事先定义DLL_API，具体如下：

```
#define DLL_API extern "C" _declspec (dllexport)
#include "DLL.h"
int Add(int a,int b)
{return a+b;}
int Subtract(int a,int b)
{return a-b;}
```

定义了DLL_API 后，在动态链接库的实现文件中，加法和减法运算的两个函数就被申明为：

```
extern "C" _declspec(dllexport) int  Add(int a,int b);
extern "C" _declspec(dllexport) int  Subtract (int a,int b);
```

而其他编程人员的程序不用关心DLL_API，不用事先定义它，因此，在包含头文件后，在编译预处理后，函数申明为：

```
extern "C" _declspec(dllimport) int  Add(int a,int b);
extern "C" _declspec(dllimport) int  Subtract (int a,int b);
```

其中关键字_declspec(dllimport)指出用户程序需要的函数在DLL 中需要导入的信息，而关键字_declspec(dllexport)指出DLL 中需要导出的信息。

对函数实现文件进行编译链接，会在工程的Debug 文件夹下会生成DLL 文件（DLL.DLL）和导入库文件(DLL.LIB)。导入库文件较小，其中并不包含实际代码，只是记录了相应的DLL 文件中的信息，以便在链接时提供DLL 中的相关函数等信息。

（3）信息查看。生成了DLL 文件后，有时需要查看DLL 中的导出信息，这可以通过命令Dumpbin 来实现。Dumpbin 实际上是VC++ 安装目录下的一个文件，该安装目录通常为：C:\Program Files\Microsoft Visual Studio\VC98\Bin。有时为了更方便地使用Dumpbin 命令，需要事先运行相同目录下的VCVARS32.BAT 批处理文件，该批处理文件是用来建立使用VC 工具的环境。在查看前面建立的DLL 文件时，需要键入如下命令来查看导出信息：

```
D:\temp\DLL\Debug>dumpbin /exports dll.dll
```

该命令列出的主要信息如下：

ordinal	hint	RVA	name
1	0	00001005	Add
2	1	0000100A	Subtract

在导出的信息中，可以看到函数名没有发生改变，但在申明函数中去掉extern “C”后，查看生成的DLL 文件，就会发现导出函数名发生了变化。

此外，还可以通过图形工具Depends 来查看DLL 信息，该工具位于开始菜单中的Visual studio 工具集中。

1.2 使用模块定义文件(.def)创建DLL

模块定义文件是文本文件，它包含DLL 的声明信息和DLL 文件中导出的函数。本例中可建立一模块定义文件DLL1.DEF，文件内容如下：

```
LIBRARY DLL1
EXPORTS
Add
Subtract
```

其中，LIBRARY 语句的作用是指出DLL 的名称，EXPORTS 指出要导出的函数名称。查阅MSDN 可以知道，导出函数时还可以在函数后加@n 表示要导出的函数顺序号，如Add @1、Subtract @2 等。

在DLL 的实现文件(DLL1.CPP)中写上Add 和Subtract 函数的定义，在编译链接后生成DLL1.DLL。通过命令dumpbin 查看动态链接库的信息时，也可以看到函数的导出名称没有改变。

2　动态链接库(DLL)的调用

调用DLL 分为隐式链接和显式链接，使用方式和条件都不相同。

2.1 隐式链接

隐式链接通常需要三个文件，即DLL 文件（DLL.DLL)、导入库文件(DLL.LIB)和头文件（DLL.H)，而且要把这三个文件复制到应用程序工程目录下。其实Windows 操作系统的加载程序查找应用程序所需的全部DLL，并将它们映射到进程的地址空间中时，其搜索的先后顺序如下：包含可执行映像文件的目录；当前目录；Windows 系统目录； Windows 目录； PATH 环境变量中列出的各个目录。

除了要把相关的文件复制到指定的目录下以外，还要在工程的设置对话框中的Link 选项卡上，在Li-

brary Modules框中要写入导入库文件的名称。

隐式链接有其缺点，即当DLL中包含较多的函数时，装载应用程序时会将全部的DLL中的函数装入内存，而不管是否用到该函数。

例如，创建一个基于对话框的MFC AppWizard(.exe)应用程序TestDLL.exe的过程。

(1) 用MFC AppWizard(.exe)生成一个基于对话框的应用程序TestDLL；

(2) 把前面生成的三个文件：DLL.H、DLL.DLL、DLL.LIB复制到工程目录下；

(3) 工程设置对话中的Link选项卡上写入DLL.LIB；

(4) 在TestDLLDlg.CPP的首部添加代码：#include "DLL.H";

(5) 在对话框资源上添加两个命令按钮，其控件ID分别为：IDC_BTN_ADD和IDC_BTN_SUBTRACT，Caption分别为Add和Subtract，代码如下：

```
    void CTestDLLDlg::OnBtnAdd()
{
// TODO: Add your control notification handler code here
    CString str;
    str.Format("4+7=%d",Add(4,7));
    MessageBox(str);
}
void CTestDLLDlg::OnBtnSubtract()
{
    // TODO: Add your control notification handler code here
    CString str;
    str.Format("4-7=%d",Subtract(4,7));
    MessageBox(str);
}
```

最后，应用程序的运行结果如图1所示。

图1 应用程序运行结果

2.2 显式链接

显式链接是指应用程序执行过程中调用某函数时，才把该函数从DLL中装入内存，也可以随时卸载DLL文件。这是一种比较节约内存的调用方式，与隐式链接相比，它不需要头文件和导入库文件，具有更好的灵活性。

在显式链接中常常要用到以下几个函数：LoadLibrary()和FreeLibrary()函数用来加载和卸载一个DLL文件映像；GetModuleHandle()函数可以检查DLL是否已经被映射到进程的地址空间中；GetProcAddress()函数用来得到已加载的DLL中某个函数的地址。

为了显式链接已经创建的DLL1.DLL，可以再次创建一个基于对话框的MFC AppWizard(.exe)应用程序TestDLL1.exe，也在对话框上添加命令按钮，其中，做加法运算的按钮代码如下：

```
    void CTestDLL1Dlg::OnBtnAdd()
{
    // TODO: Add your control notification handler code here
    HMODULE hInst=GetModuleHandle("dll1.dll");
    if(hInst==NULL)
    {hInst=LoadLibrary("dll1.dll");}
    typedef int (*AddProc)(int a,int b);// 定义函数指针类型
    AddProc ad=(AddProc)GetProcAddress(hInst,"Add");
    if(!ad)
    {MessageBox(" 获取函数地址失败!");
    return;
}
    CString str;
    str.Format("4+7=%d",ad(4,7));
    MessageBox(str);
}
```

3 结束语

动态链接库（DLL）封装了共享资源和代码，在开发以Windows为平台的应用程序时，使用DLL技术将大大节约内存。同时，在应用系统定制和升级时，只需替换相应的DLL，而系统不用重新编译。所以，掌握好DLL技术对程序开发很有好处。

[参 考 文 献]

[1]周金萍，徐丙立.Windows系统编程[M].北京：人民邮电出版社,2002.
[2]王建华，张焕生，等(译).MFC Visual C++6编程技术内幕[M].北京：机械工业出版社,2002.
[3]北京博彦科技发展有限公司(译).Charles Petzold.Windows程序设计(第5版)[M].北京:北京大学出版社,1999.
[4]北京博彦科技发展有限公司(译).Microsoft Corporation.Visual C++6.0 MFC应用程序开发[M].北京:清华大学出版社,2002.

[责任编辑：冯云华]