Windows 动态链接库原理分析及其应用

熊　华　刘凤新　潘小莉
(北京化工大学信息科学与技术学院,北京　100029)

摘　要: 深入分析了 Windows 动态链接库(DLL)的内部原理,阐述了 DLL 的动态链接、地址映射、引用计数等特性,介绍了创建和调用动态链接库的方法和技巧,为工程人员开发和使用动态链接库提供了一定的技术支持。

关键词: 动态链接库;高级 Windows 应用程序;地址映射;动态链接
中图分类号: TP313

引　言

在 Windows(98/2000/XP)操作系统的体系结构中,动态链接库占据重要地位。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。Windows 操作系统本身就由许多 DLL 组成,对于 Windows 应用程序开发者而言,其中最重要的 3 个 DLL 是 Kernal32.dll(用于管理内存、进程和线程),GDI32.dll(用于画图和显示文本)和 User32.dll(用于处理用户接口)[1-2]。Win32 应用程序接口(API)也是由 DLL 实现的。虽然 WDM 已经成为 Windows 的标准驱动程序模型(随着 Windows98 逐渐被 Windows2000/XP 取代,VxD 模式将被抛弃),仍然有许多驱动程序包含 DLL,尤其对于开发者自己研制的非标准设备(如高速数据采集卡)的驱动程序。所以深入理解 DLL 的原理并熟练掌握 DLL 的编程技术对开发高级 Windows 程序十分重要。

1　使用 DLL 的优点

开发者应根据系统功能的需要设计相应的 DLL,清楚 DLL 的优点可以作为选择 DLL 的一个标准,其优点主要有:

(1)节省内存空间。一个 DLL 在内存中只存在一个实例,即只存在一段可执行代码,多个应用程序可以同时共享这段代码。不需要代码的时候可以卸载 DLL 直到需要的时候再加载。

收稿日期: 2003-05-29
第一作者: 男,1978 年生,硕士生
E-mail: xionghua2008@yahoo.com.cn

(2)封装共享资源和代码。DLL 中可以包含字符串、位图等 Windows 资源,多个应用程序可以通过 DLL 共享这些资源。

(3)接口通用。只要遵守 DLL 的接口规范,DLL 可被众多编程语言调用。比如界面由 VB 实现,而核心逻辑由 VC 编写的 DLL 实现。

(4)定制和升级系统。选择不同的 DLL 可以实现不同的系统;当核心逻辑改变后,只需替换相应的 DLL,而系统不用重新编译。

(5)硬件访问功能。有些高级语言硬件访问功能较弱(如 VB),通过 DLL 不仅可以方便的读取内存、寄存器、端口等,还能保持硬件访问的独立性,减少异常,降低系统崩溃的可能。

2　DLL 内部原理分析

DLL 是包含资源、数据和函数的模块,虽然它包含可执行代码,但是它只能被 EXE 或者其它 DLL 调用,而不能独立运行。正如控制台程序的入口点为 Main(),Windows 应用程序的入口点为 WinMain(),驱动程序的入口点为 DriverEntry()一样,DLL 的入口点为 DllMain()。当被调用时,可执行代码被动态加载到内存中,并被映射到调用进程的地址空间中。DLL 中的数据和函数分为内部的和外部的。内部数据和函数只能在所在 DLL 中使用,而外部数据和函数可以被其它模块使用。DLL 文件包括一张导出表,其中记录了 DLL 向外部输出的数据和函数,外部调用模块正是通过这张表完成对 DLL 中的数据和函数的访问。

2.1　动态链接

当应用程序引用某个函数库时,静态链接是将调用的函数代码从库中提取出来,附加到应用程序

的可执行模块中,因而不同的应用程序引用同一函数时,每个应用程序的可执行模块中都存在这一函数的拷贝[3]。静态链接示意如图1所示。

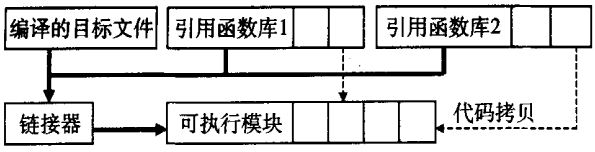

图1 静态链接

Fig. 1 Static link

动态链接并不复制代码到应用程序的可执行模块中,而是在应用程序运行期间加载DLL,并通过导出表重定位要调用的函数(这个过程就称为动态链接),此时多个应用程序共享内存中同一段函数代码[3]。动态链接示意如图2所示。

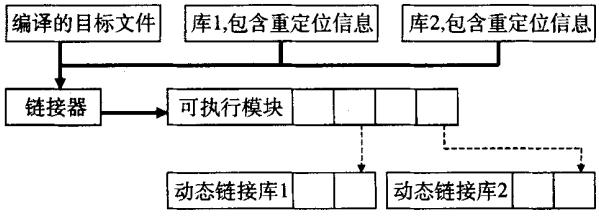

图2 动态链接

Fig. 2 Dynamic link

2.2 地址映射

应用程序在运行期间加载DLL时有2种方式:1)在应用程序启动时加载DLL(又称为隐式动态链接)。这种方式是应用程序预先知道要引用哪些DLL,然后一次性加载所有DLL,在应用程序可执行模块中遇到DLL中的函数标识符时直接定位到对应DLL中的代码。在应用程序被编译链接时,问题是如何解析出现应用程序代码中而实际定义在DLL中的函数标识符(称为外部引用)。解决外部引用的方法是在应用程序编译链接时包含相应DLL的LIB文件,LIB文件中包含DLL的输出列表(LIB不同于导出列表,导出列表存在DLL文件中,包含重定位信息,而LIB包含的输出列表仅相当于函数标识符的占位符,让编译链接器知道这个函数标识符是合法的,其定义在指定的DLL中)。包含LIB的同时链接器会在应用程序的EXE文件中加入相关的DLL信息,使得操作系统在启动应用程序时能根据这些信息查找所需的DLL并将它们映射到应用程序进程的虚拟地址空间。2)在应用程序运行过程中加载DLL(又称为显式动态链接)。由于功能逻辑的需要,调用哪个DLL中的哪个函数可能要在应用程序运行过程中才能确定,此时地址空间的映射(加载DLL、卸载DLL以及重定位调用的函数)是在应用程序运行过程中动态完成的。

如果一个进程在运行过程中多次加载同一个DLL,只要其间DLL的卸载次数始终小于加载次数(使用显式动态链接可能存在这种情况),则DLL始终存在于虚拟地址空间,且每次加载返回的DLL在虚拟地址空间的地址都相同。如果多个进程都加载同一个DLL文件(不是文件名相同的DLL,而是同一个磁盘文件),虽然此DLL在各个进程的虚拟地址空间的地址可能不同,但实际上这些虚拟地址都映射到一个相同的物理地址,因为物理内存中只存在DLL的一个映像。

2.3 引用计数

DLL动态存在于内存空间,能被多个应用程序或者一个应用程序的多个进程共享,DLL中和调用DLL的进程中都存在引用计数器:1)DLL中的引用计数器决定DLL是否从物理内存空间中清除,此引用计数器为正整数,表明引用该DLL的进程数,但物理内存中只存在该DLL的一个映像;当此引用计数器为0时,DLL从物理内存中彻底清除。2)调用DLL的进程中的引用计数器决定该进程能否继续访问DLL,当此引用计数器为0时,DLL从该进程的虚拟地址空间中清除,该进程不能再访问此DLL,尽管此时物理内存中可能还存在此DLL的映像。

进程每加载一次DLL,这2种引用计数器分别加1;每卸载一次DLL,这2种引用计数器分别减1。隐式动态链接在启动应用程序时加载DLL,退出时卸载DLL,并改变计数器的值。显式动态链接在应用程序运行过程中动态加载和卸载DLL,并改变计数器的值。

2.4 数据共享

多个进程对DLL可执行代码和DLL公共数据的引用对应到DLL物理空间的相同地址,对其私有数据的引用对应到DLL物理空间的不同地址[3]。示意图如图3所示。

3 DLL的创建和调用

许多编程工具(如TC,BC,BCB,VC)都可以创建DLL,其中VC提供的功能最为强大,配合其丰富的MFC(Microsoft Foundation Classes),使得VC成为许多开发者开发DLL的首选。VC支持的动态

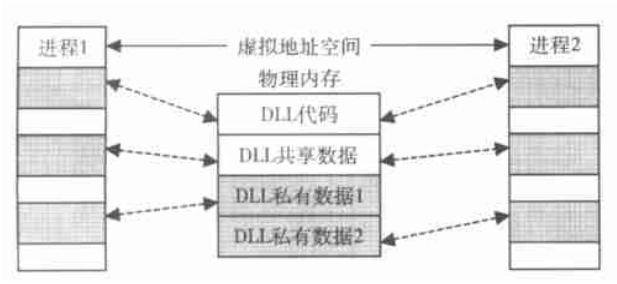

图 3　DLL 与进程的地址空间映射

Fig. 3　Address space mapping between DLL and process

链接库包括 4 种:非 MFC DLL,常规静态 MFC 链接 DLL,常规动态 MFC 链接 DLL,扩展 MFC DLL。

3.1　非 MFC DLL

非 MFC DLL 中不能包含 MFC,如果 DLL 中不需使用 MFC,则创建这种类型的 DLL 可以节省链接到 MFC 所需的磁盘和内存空间。这种类型的 DLL 在工程中应用得最为广泛,它常常包含各种算法的实现、共享常量和变量等。首先创建工程,选择工程类型为"Win32 Dynamic-Link Library"。接着选择工程的属性:"An empty DLL project"不创建任何文件,开发者可以将已经存在的 *.cpp, *.h, *.def 等文件引入工程;"A simple DLL project"仅包含空的入口函数 DllMain,但没导出和导入宏定义;"A DLL that exports some symbols"包含简单的入口函数 DllMain,导出和导入宏定义,还包含导出变量、导出函数和导出类的模型。

3.2　常规静态 MFC 链接 DLL

如果在 DLL 中要使用 MFC,则必须要创建 MFC 链接 DLL。常规静态 MFC 链接 DLL 使用 MFC 的静态链接库,即在编译链接时就将 MFC 从静态链接库中提取出来,并包含在 DLL 文件中,而不是在程序运行时再链接到 MFC 库,因此生成的 DLL 文件也比较大。同一般 MFC 应用程序相同,常规静态 MFC 链接 DLL 中封装了一个 CWinApp 继承类的对象,但这个对象中并没有消息循环,如果需要消息循环,只能让调用 DLL 的应用程序调用 DLL 的输出例程,在这个例程中调用 CWinApp::PreTranslateMessage()。在 DLL 中分配的内存空间限于在 DLL 中使用,禁止在 DLL 和调用程序之间传递在 DLL 中分配的内存指针和 MFC 对象指针。

3.3　常规动态 MFC 链接 DLL

常规动态 MFC 链接 DLL 与常规静态 MFC 链接 DLL 大致相似,不同的是它动态链接到 MFC 的函数库中(各种 MFC *.dll)。值得注意的是,由于 MFC *.dll 也可能被调用 DLL 的应用程序引用,则进程中可能存在多个相同的在 MFC *.dll 中定义的共享全局变量。解决机制是赋予模块一个模块状态信息,使用 AFX_MANAGE_STATE 宏可令此模块状态信息自动切换,这样就把不同的共享全局变量限制在自己的模块中共享。

3.4　扩展 MFC DLL

扩展 MFC DLL 实现 MFC 的继承类的定义,采用动态链接 MFC 技术,要引用 MFC *.dll 的应用程序和 DLL 才能调用扩展 MFC DLL。

3.5　导出方法

DLL 中包含共享的数据、函数、类和其他资源,它们的导出方法并不一致。一种导出方法是使用 *.def 文件(其详细编写规则可参考 MSDN),在其中声明要导出的成员,*.h 和 *.cpp 文件的编写规则同一般 EXE 程序源文件的编写规则一样。另一种导出方法是使用导出符号(或其宏定义),直接在导出成员定义前加导出符号。

(1)导出数据。使用 *.def 文件,例:"GlbVarName @1 DATA";使用导出符号,例:"_declspec(dllexport) int GlbVarName";

(2)导出函数。使用 *.def 文件,例:"GlbFunctionName @1";使用导出符号,例:"_declspec(dllexport) int GlbFunctionName(int par1,int par2)";

(3)导出类:使用 *.def 文件,例:"GlbClassName @1";使用导出符号,例:"class_declspec(dllexport) GlbClassName{…}"。

3.6　调用方法

调用 DLL 分为隐式调用和显式调用,使用方式和条件都不同[4]。

(1)隐式调用 DLL 需要三种文件:DLL 文件(包含具体的代码和重定位信息)、LIB 文件(包含链接信息)、头文件(包含各种标识符的外部声明)。将 DLL 拷贝到工程目录下,将 LIB 文件名添加到工程链接信息中,并将头文件包含在工程中。一般头文件在编译 DLL 的时候也需要,但声明方式不是外部声明。为了使 DLL 和调用 DLL 的应用程序都能使用这个头文件并满足各自要求,可以在头文件前编写条件编译语句声明一个宏 DLL_API,在编译 DLL 时定义 DLL_EXPORTS,而在调用 DLL 的应用程序中不定义 DLL_EXPORTS,并在各个函数定义前添加这个宏。代码如下

```
#ifdef DLL_EXPORTS        #define DLL_API
```

_declspec(dllexport)

else # define DLL_API _declspec(dllimport)

endif

(2) 显式调用使用 API 函数访问 DLL ,LoadLibrary() 加载 DLL ,FreeLibrary() 卸载 DLL ,GetProcAddress() 获得 DLL 中的函数指针[5]。代码如下

```
float result;
typedef float (CALLBACK * Function)(float, float);
HINSTANCE hDLL = LoadLibrary("DLL1.dll");
if(hDLL != NULL)
{
    Function fun = (Function) GetProcAddress(hDLL,"AddFunction");
    if(fun != NULL){result = fun(3.0, 4.5);}
    FreeLibrary(hDLL);
}
```

值得注意的是,DLL 输出函数默认按 C 语法规则(_cdecl 方式),而如果调用 DLL 的应用程序以 C++ 语法规则编写,则 DLL 的输出函数也要按 C++ 语法规则写,即在函数返回类型和函数名字间加_stdcall,代码如下

```
float_stdcall AddFunction(float x, float y){return (x + y);}
```

4 结束语

在多个实际工程中都运用了 DLL 技术,如在"实时检测航天器动力学特性软件系统"中将快速傅立叶算法包含在 DLL 中,在"基于神经网络的化工过程控制 CAI 多媒体课件"中将 BP 算法包含在 DLL 中,使系统结构更清晰,同时增强了系统的可扩展性。以上经验是笔者在工程中总结而成的,限于篇幅,还有些技术问题未能详细阐述,比如 VB 和 DLL 交互参数传递的问题、如何提取 DLL 导出信息等,希望本文能给工程技术人员提供一定的技术支持。

参 考 文 献

[1] 王建华,张焕生,侯丽坤,等译. Windows 核心编程[M]. 北京:机械工业出版社,2000

[2] Solomon D A, Russinovich M. Inside microsoft windows 2000[M]. Washington: Microsoft Press,2000

[3] 尤晋元,史美林,陈向群,等. Windows 操作系统原理[M]. 北京:机械工业出版社,2001

[4] 周振红,冯夏庭,杨国录,等. VB 调用 VC 创建的 Win32 DLL 函数[J]. 计算机工程,2002,28(4):285

[5] 唐丽丽. WINDOWS 系统中代码的动态嵌入执行技术[J]. 武汉理工大学学报(交通科学与工程版),2002,26(1):51 - 55

Theoretic analysis and application of windows dynamic link library

Xiong Hua Liu Feng-xin Pan Xiao-li

(College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China)

Abstract: The internal theory of Windows dynamic link library (DLL) was analyzed. Various characteristics of DLL such as dynamic linking, memory address mapping and reference counting were explained. The method and technique of designing and using DLL were proposed, prodiding a technical support for engineers in designing Windows dynamic link library.

Key words: dynamic link library; advanced Windows application; memory address mapping; dynamic linking

(责任编辑 刘同帅)