DLL

WINDOWS操作系统中，几乎所有的内容都是一DLL的形式存在的。

静态链接

链接程序首先对库文件(.lib)进行搜索，直到在某个库中，找到包含函数的对象模块为止。然后，链接程序把这个对象模块拷贝到可执行文件中。

缺点是：当多个程序都运行这个DLL，那么内存就会有这个函数的多个副本，造成资源的浪费。

动态链接

区别于静态链接的是这次查找的是函数的输入记录，链接程序把输入记录拷贝到可执行文件中。

* Regular DLL
* Extension DLL

由MFC编写，且只能由MFC编写的应用程序调用。

访问DLL就是将应用程序中的导入函数与DLL文件中的导出函数进行链接。

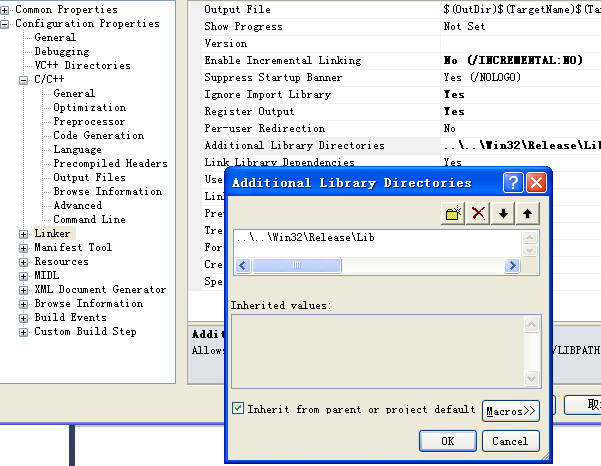
两种链接方式：

显示链接

用Win32的LoadLibary函数加载DLL，函数返回DLL的句柄。

用GetProcAddress函数，根据函数名获取DLL函数的入口指针。

隐式链接



#pragma comment(lib,”mylib”)

在使用隐式链接的进程启动的时候，如果没有找到所需要的DLL就会系统终止。

如果应用程序使用隐式链接的方式链接了很多DLL，那么在应用程序启动的时候，系统要装载所有的DLL，变的很慢，并且占用很多内存，所以对于那些很少用到的DLL最好采用显示链接的方式链接

* 在动态库的情况下，有两个文件，一个是引入库（.LIB）文件，一个是（.DLL）文件，引入库文件包含被DLL导出的函数的名称和位置，DLL包含实际的函数和数据，此时的.LIB很小
* 在静态链接库的情况下，只有一个文件（.LIB）

，此时的（.LIB）很大，是库所有的实现过程。

COM技术及组件开发

COM定义了客户与构件之间互操作的标准，包括规约与实现两个部分

规约部分定义了构件之间的通信机制，这些规约不依赖于任何特定的语言和操作系统

实现部分就是COM库，为COM规约的具体实现提供了一些核心服务

COM（Component Object Model），组件对象模型，是一种以组件为发布单元的对象模型，这种模型使得各个软件组件可以用一种统一的方式进行交互

COM是一种协议，它建立了一个软件模块同另一个软件模块之间的连接，提供了组件之间进行交互的规范，也提供了实现交互的环境

当这种连接建立起来后，两个模块之间就可以通过“接口”的机制来进行通讯，这些接口既不需要静态的或动态的连接入口点，也不需要编码很复杂的地址

* **COM接口是一个二进制标准，客户同接口的连接并不是通过其名称或其成员函数的名称来实现的，而是通过其在内存块中的位置完成的**
* **COM接口是一个包含一个函数指针数组的内存结构，每一个数组元素包含的是一个由组件所实现的函数的地址**
* **C++通过纯抽象基类实现COM接口，实际上是定义了一个内存块的结构，但此内存只是在派生类中实现此抽象基类时才会被分配，派生类继承此抽象基类时将继承此内存结构**
* **COM构件可以和客户机程序在同一个进程中，也可以在不同的进程中**
  + **前者称为进程内构件(INPROC\_SERVER**)
  + **后者称为进程外构件(LOCAL\_SERVER** **)**
* **如果是DCOM，进程外构件还包括**
  + **CLSCTX\_REMOTE\_SERVER**
* **对于进程外构件，由于进程之间相互屏蔽，客户机进程不能直接调用构件进程，需要遵循操作系统对进程间通信所做的规定，在Windows平台上，进程外构件以EXE实现，COM采用本地过程调用（LPC）作为进程间通信机制**
* **COM不仅定义了客户与构件之间交互的规约，而且提供了全面的实现细节，COM的实现与操作系统平台密切相关，包括如下内容**
* COM库（COM Library）
* 系统注册表（windows registry）
* 类工厂
* COM服务器

**COM库**

* 是COM标准的系统级实现，为COM对象的标识和创建、内存管理、构件程序的卸载等提供了一组标准接口和辅助函数
* 在Windows系统中，COM库主要包含在OLE32.dll、OLE32.lib和RPCSS.dll文件中
* COM库很多地方直接用到了Windows系统的一些特性，例如系统注册表和动态链接库，COM通过注册表查找本地服务器(即EXE程序) 以及程序名与CLSID的转换

**类工厂**

* 类工厂是能够创建其它COM对象的特殊COM对象，类工厂支持一个特殊的接口IClassFactory，为例化CoClass构件类提供了一种标准机制
* 每一个COM对象类应该有一个相应的类工厂对象
* IClassFactory接口中定义了两个重要的操作
* CreateInstance:生成COM对象并返回所请求接口的指针
* LockServer：负责在内存中保持COM构件

消息和消息处理

1. 获取此时点击的按钮id

调用CWnd的方法，GetFocus()->GetDlgCtrlID();

1. [MFC \_窗口ID，句柄，指针三者相互转换函数](http://m.blog.csdn.net/blog/aqaya/8206898)

ID--HANDLE--HWND三者之间的互相转换  
id->句柄        hWnd = ::GetDlgItem(hParentWnd,id);  
id->指针        CWnd::GetDlgItem();  
句柄->id        id = GetWindowLong(hWnd,GWL\_ID);  
句柄->指针    CWnd \*pWnd=CWnd::FromHandle(hWnd);  
指针->ID       id = GetWindowLong(pWnd->GetSafeHwnd,GWL\_ID);  
                                            GetDlgCtrlID();  
指针->句柄     hWnd=cWnd.GetSafeHandle() or mywnd->m\_hWnd;

1. cstring int 相互转换

CString str = \_T("123");  
int i = \_ttoi(str);  
===============  
int i = 123;  
CString str ;  
str.Format(\_T("%d"), i);

1、CString转char \*

CString cstr;

char \*p =(LPSTR)(LPCTSTR)cstr;

2、string转 CString  
CString.format(”%s”, string.c\_str());

用c\_str()确实比data()要好.

3、char转 CString  
CString.format(”%s”, char\*);

4、char转 string  
string s(char \*);

只能初始化，在不是初始化的地方最好还是用assign().

5、string转 char \*  
char \*p = string.c\_str();

6、CString转 string  
string s(CString.GetBuffer());

1、char \* 转成 CString

[复制代码](javascript:void(0);)

char \*str="abcd1234";  
//预转换，得到所需空间的大小  
int wcsLen = ::MultiByteToWideChar(CP\_ACP, NULL, inputbuffer, strlen(inputbuffer), NULL, 0);  
//分配空间要给'\0'留个空间，MultiByteToWideChar不会给'\0'空间  
wchar\_t\* wszString = new wchar\_t[wcsLen + 1];  
//转换  
::MultiByteToWideChar(CP\_ACP, NULL, inputbuffer, strlen(inputbuffer), wszString, wcsLen);  
//最后加上'\0'  
wszString[wcsLen] = '\0';  
//附加到CString对象上  
CString content;  
content.Append(wszString);

[复制代码](javascript:void(0);)

1、CString 转成 char \*

[复制代码](javascript:void(0);)

CString str=\_T("hello kitty 12345 hello jackson");  
int len=str.GetLength();  
   
char \*buffer=new char[len+1];  
memset(buffer,0,len+1);  
  
WideCharToMultiByte(CP\_OEMCP, NULL, (LPCWSTR)str, -1,NULL, 0, NULL, FALSE);  
WideCharToMultiByte(CP\_OEMCP, NULL, (LPCWSTR)str, -1,(LPSTR)buffer, len, NULL, FALSE);  
   
buffer[len]='\0';

1. 在SystemMenu里添加菜单

GetSystemMenu(FALSE);

OnSysCommand

1. 画布cdc

Cdc是一个画板，为GDI对象提供显示的空间

静态控件CStatic

编辑框控件CEdit

按钮控件CButton

列表框控件CListbox

组合框控件CCombobox

1. [CStdioFile的Writestring无法写入中文的问题](http://blog.csdn.net/ljz888666555/article/details/5511247)

char\* old\_locale = \_strdup( setlocale(LC\_CTYPE,NULL) );

setlocale( LC\_CTYPE, "chs" );//设定

file.WriteString(\_T("abc你好"));//正常写入

setlocale( LC\_CTYPE, old\_locale );

free( old\_locale );//还原区域设定

简化处理可以仅使用语句setlocale( LC\_CTYPE, "chs" )。

setlocale:  
函数原形为：char \*setlocale( int category, const char \*locale );  
头文件：<locale.h>  
所支持的操作系统为:ANSI, Win 95, Win NT  
对于简体中文可以使用如下设置：setlocale( LC\_ALL, "chs" );