C++ DLL的隐式与显式调用

个人理解

DLL分为显式与隐式调用，显式不需要.lib文件,而隐式需要.lib文件

但是这里有一个重要的问题，即visual stuido工具不同版本间的问题，目前vs主要三大版本vc2015, vc2017, vc2019□如果低版本vs使用高版本vc生成出来的.lib很有可能有问题（不知道是不是叫binary-compatibility restriction https://docs.microsoft.com/en-us/cpp/porting/binary-compat-2015-2017?view=vs-2019），， 所以从这点考虑突显了显式调用的优点，因为它不需要.lib文件。

目前我遇到的FT D2XX 驱动的问题有可能就是这个问题， 有可能是使用的vs2017做的，而我目前使用的是vc2015□还有一个原因是静态库.lib文件与动态库引入库.lib文件是不一样的，前者一般文件较大上M□而后者一般就只有几十KB□

Operating System	Release Date	x86 (32-bit)	x64 (64-bit)
Windows*	2017-08-30	2.12.28	2.12.28

网上参考文章

原文链接：https://www.cnblogs.com/johngu/p/7877939.html

C++ dll的隐式与显式调用　　应用程序使用DLL可以采用两种方式：一种是隐式链接，另一种是显式链接。在使用DLL之前首先要知道DLL中函数的结构信息□Visual C++6.0□或者更先进的版本）在VC\bin目录下提供了一个名为Dumpbin.exe的小程序（使用方法见VS自带工具□dumpbin的使用），用它可以查看DLL文件中的函数结构。另外□Windows系统将遵循下面的搜索顺序来定位DLL□ 1□包含EXE文件的目录，2．进程的当前工作目录□ 3□Windows系统目录□ 4□Windows目录，5．列在Path环境变量中的一系列目录。

VS下动态库dll的显式调用

动态库的加载分两种形式：分为静态加载和动态加载。静态加载时，对应的头文件□DLL□和LIB缺一不可，并且生产的EXE没有找到DLL文件就会导致“ 应用程序初始化失败 ”。动态加载只需要dll□通过LoadLibrary()函数进行加载，但该方式对生成的dll的规范有一定的要求否则容易出错。

1. 隐式链接

隐式链接就是在程序开始执行时就将DLL文件加载到应用程序当中。实现隐式链接很容易，只要将导入函数关键字_declspec(dllimport)函数名等写到应用程序相应的头文件中就可以了。下面的例子通过隐式链接调用MyDll.dll库中的Min函数。首先生成一个项目为TestDll□在DllTest.h□DllTest.cpp文件中分别输入如下代码：

复制代码

```
//Dlltest.h
```

```
#include"MyDll.h"
#pragma comment(lib,"MyDll.lib")
extern "C"_declspec(dllimport) int Max(int a,int b);
extern "C"_declspec(dllimport) int Min(int a,int b);
//TestDll.cpp
#include"Dlltest.h"
void main()
{int a;
a=min(8,10)
printf("比较的结果为%d\n",a);
}
```

复制代码

在创建DllTest.exe文件之前，要先将MyDll.dll和MyDll.lib拷贝到当前工程所在的目录下面，也可以拷贝到windows的System目录下。如果DLL使用的是def文件，要删除TestDll.h文件中关键字extern “C”，TestDll.h文件中的关键字Progam commit是要Visual C+的编译器在link时，链接到MyDll.lib文件，当然，开发人员也可以不使用#pragma comment(lib，“MyDll.lib”)语句，而直接在工程的Setting->Link页的Object/Moduls栏填入MyDll.lib既可。

2. 显式链接

显式链接是应用程序在执行过程中随时可以加载DLL文件，也可以随时卸载DLL文件，这是隐式链接所无法作到的，所以显式链接具有更好的灵活性，对于解释性语言更为合适。不过实现显式链接要麻烦一些。在应用程序中用LoadLibrary或MFC提供的AfxLoadLibrary显式的将自己所做的动态链接库调进来，动态链接库的文件名即是上述两个函数的参数，此后再用GetProcAddress()获取想要引入的函数。自此，你就可以象使用如同在应用程序自定义的函数一样来调用此引入函数了。在应用程序退出之前，应该用FreeLibrary或MFC提供的AfxFreeLibrary释放动态链接库。下面是通过显式链接调用DLL中的Max函数的例子。

```
#include <windows.h>
#include <cstdio>
void main(void)
{
typedef int(*pMax)(int a,int b);
typedef int(*pMin)(int a,int b);
HINSTANCE hDLL;
PMax Max
HDLL=LoadLibrary("MyDll.dll");//加载动态链接库MyDll.dll文件；
Max=(pMax)GetProcAddress(hDLL,"Max");
A=Max(5,8);
Printf("比较的结果为%d\n",a);
FreeLibrary(hDLL);//卸载MyDll.dll文件；
}
```

□□

在上例中使用类型定义关键字typedef，定义指向和DLL中相同的函数原型指针，然后通过LoadLibray()将DLL加载到当前的应用程序中并返回当前DLL文件的句柄，然后通过GetProcAddress()函数获取导入到应用程序中的函数指针，函数调用完毕后，使用FreeLibrary()卸载DLL文件。在编译程序之前，首先要将DLL文件拷贝到工程所在的目录或Windows系统目录下。

□□使用显式链接应用程序编译时不需要使用相应的Lib文件。另外，使用GetProcAddress()函数时，可以利用MAKEINTRESOURCE()函数直接使用DLL中函数出现的顺序号，如将GetProcAddress(hDLL,“Min”)改为GetProcAddress(hDLL, MAKEINTRESOURCE(2))□函数Min()在DLL中的顺序号是2），这样调用DLL中的函数速度很快，但是要记住函数的使用序号，否则会发生错误。

注意两点：

1.上面中提到extern “C”□表明函数使用c语言的风格编译函数，这样后面显式调用时□GetProcAddress第二个参数就是原来的函数名，否则如果使用C++编译的话支持函数重载，则第二个参数会发生变化，想要知道发生了什么变化，可以使用前面提到的Dumpbin工具查看dll的生成函数。具体参考文章：http://www.cnblogs.com/laogao/archive/2012/12/07/2806528.html

2.MAKEINTRESOURCE()函数可以代替复制dll生成的函数的一大串字符串名，但是必须知道函数生成的顺序，所以你也必须打开dll查看，才能知道。