我发现了这样的一个宏：

#define XXXX extern "C" \_\_declspec(dllexport)

#define XXXX extern "C" \_\_declspec(dllimport)

DLL的export是指将DLL中的函数和数据输出到其它程式中，以供其使用。

DLL的import是指使用DLL的程式引入DLL中的函数和数据。

DLL的export

DLL 中包含有一个表，称为export table（以下简称ET），其中包含了DLL中可以被外部程式使用的所有函数和数据的名字。

只有记录在ET中的函数和数据才可以被外部程式所使用（如果 没有.DEF文件的话），其它所有没有记录在ET中的函数和数据

都被视为是DLL私有的。因此，要将DLL中的函数和数据export只有两个方法：

1. 为DLL创建一个.DEF文件（模块定义文件），并在build该DLL时使用这个.DEF文件。使用这种方法使你可以将函数按序号export。
2. 在DLL中想要export的函数和数据定义前添加\_declspec（dllexport）关键字（对于函数和变量定义，加在最前面；对于class定义，加在class关键字后），这样该函数和数据就会被添加到ET中。使用这种方法函数将按名字export。

在WINDOWS下，无论使用上述的哪一种方法，都必须要将export函数声明为\_stdcall。

**关于C和C++的兼容问题**

如果要写C和C++兼容的DLL，因为在C和C++下使用了不同的名字修饰规则以及不同的调用约定，所以，如果DLL是用C编写和编译的，则在用于C++模块时，函数的声明前应加上**extern “C”**关键字，以告诉LINKER使用C外部连接（即按照C名字修饰规则在外部模块中寻找函数）；反之，如果DLL是用C++编写和编译的，则在用于C模块时，函数的声明前要加上**extern “C++”**关键字。VC++通过\_cplusplus宏来标识C++程式。如果是C++程式，VC编译器就会为你定义\_cplusplus宏。所以在DLL中可以使用如下的技术来解决兼容问题：

**DLL的import**

　　　 外部程式的一个源文件要使用DLL中的函数和数据，就像要使用外部模块中的函数和数据一样，必须首先给出函数和数据的声明；对于class则要给出类的定义，这就称为import。对于VC编译器，Import DLL的函数和数据的语法与一般的声明类似，但要在前面加上\_declspec（dllimport）关键字（对于函数和变量声明，加在最前面；对于 class定义，加在class关键字后）。如果是函数，则该关键字是可选的，但使用该关键字有可能会导致编译器产生较高效的代码。但对于变量和 class，则必须使用该关键字。

　　　 通过使用以下的技术，可以编写在.LIB文件和外部程序源文件通用的头文件：

        #ifdef \_EXPORTING

        #define CLASS\_DECLSPEC    \_\_declspec(dllexport)

     #else

        #define CLASS\_DECLSPEC    \_\_declspec(dllimport)

        #endif

编译器提供的\_EXPORTING宏可以用于标式该源文件来自DLL文件还是外部程式

# [extern "C" \_declspec(dllexport)浅析](http://blog.csdn.net/ghevinn/article/details/17297685)

extern "C"     extern "C"使得在C++中使用C编译方式成为可能。在“C++”下定义“C”函数，需要加extern “C”关键词。用extern "C"来指明该函数使用C编译方式。输出的“C”函数可以从“C”代码里调用

使用微软专用的\_declspec (dllexport)

cpp文件在编译为OBJ文件时要对函数进行重新命名，C语言会把函数name重新命名为\_name,而C++会重新命名为\_name@@decoration，

extern "C"表示用C语言的格式将函数重命名

要输出整个的类，对类使用\_declspec(\_dllexpot)；要输出类的成员函数，则对该函数使用\_declspec(\_dllexport)

像如下代码

int main()

{

   func()

   return 0;

}

编译器生成像下面的代码

call func1

连接器传入像下面的代码

call 0x4000000         ; 函数func的地址

如果func在DLL中，连接器就无法得到func的直接地址；在32位系统下，连接器将生成一个thunk包含func的调用地址

0x40000000:    jmp DWORD PTR \_\_imp\_func

\_\_imp\_func事func在.exe导入函数表中的地址,这个地址传入连接器进行连接

这样生成的代码比较大而且速度慢；

而使用\_\_declspec(dllimport) 则直接告诉连接器直接调用DLL中func的地址，不要使用thunk

  \_\_declspec(dllimport) void func(void);

   void main(void)

   {

       func1();

   }

生成如下代码

   call DWORD PTR \_\_imp\_fu