[1. Solution 说明 2](#_Toc432515827)

[2. 动态链接/静态链接， 隐式链接/显示链接 2](#_Toc432515828)

[3. 链接DLL时，必须链接.lib文件而不能是.dll文件（其实是隐式链接） 2](#_Toc432515829)

[4. Project添加Reference的方式 4](#_Toc432515830)

# Solution 说明

一个是生成DLL的工程CppDLL，一个是调用DLL工程 DemoConsole。

注意，CppDLL编译出来的时候会生成两个文件，一个是CppDLL.lib， 一个是CppDll.dll 两个文件都有各自的用处。

运行的时候，只需要CppDll.dll, 和DemoConsole.exe, CppDll.lib 可以删除的。

编译DemoConsole的时候，可以没有CppDll.dll， 必须有CppDll.lib

# 动态链接/静态链接， 隐式链接/显示链接

动态链接就是用dll库， 静态链接就是用lib库

隐式链接和显示链接都是针对动态链接库dll来说的。当主程序要使用dll时，可以用loadlibrary函数显示链接dll，此时只要有dll文件就可以了，称为显示链接。还有一种形式是主程序编译时，添加对dll的lib文件的引用，这样就可以直接使用dll中到处的函数名而不用使用loadlibrary函数，称为隐式链接。（程序其实链接dll的lib库的时候，还是静态链接，只不过lib库里只有函数的名称和地址，没有具体的实现代码和数据而已）

程序员在建立一个 DLL文件时，链接程序会 自动生成一个与之对应的LIB导入文件。该文件包含了每一个DLL导出 函数的符号名和可选的标识号，但是并不含有实际的 代码。LIB文件作为DLL的替代文件被编译到 应用程序 项目中。

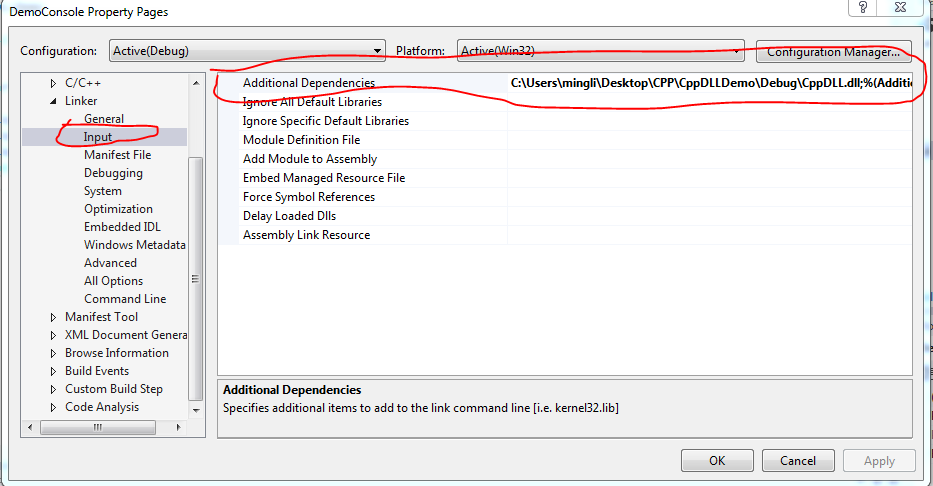
当程序员通过静态链接方式编译生成应用程序时，应用程序中的调用函数与LIB文件中导出符号相匹配，这些符号或标识号进入到生成的EXE文件中。LIB文件中也包含了对应的DLL文件名（但不是完全的路径名），链接程序将其存储在EXE文件内部。当应用程序运行过程中需要加载DLL文件时，Windows根据这些信息发现并加载DLL，然后通过符号名或标识号实现对DLL函数的动态链接。所有被应用程序调用的DLL文件都会在应用程序EXE文件加载时被加载在到内存中。可执行程序链接到一个包含DLL输出函数信息的输入库文件(.LIB文件)。操作 系统在加载使用可执行程序时加载DLL。可执行程序直接通过函数名调用DLL的输出函数，调用方法和程序内部其他的函数是一样的。 调用dll的export函数是不需要lib的， dll本身就是和 lib同一个级别的东西，也提供类似的功能，调用dll函数只需要loadlibrary就行，但是你也发现，用loadlibrary很麻烦，要取出函数的地址，然后还要定义一个调用的原型。用dll比用静态链接库lib要麻烦得多。 微软为了 解决这个问题，在编译dll的时候，顺便生成一个引入库lib文件，这个lib不包含实际功能代码，只不过用来帮你做那些loadlibrary的麻烦工作，让你用 dll和用一般的lib一样方便。  dll改过之后，只要export函数的声明没有修改，也就是 参数、函数名、返回值没有发生变化，调用代码不需要重新编译。不过export函数声明修改过就必须重编译。 使用动态dll有两种方法，一种是隐式链接，一种显示链接，如果用loadlibrary就是显示链接，用lib就属于隐式链接

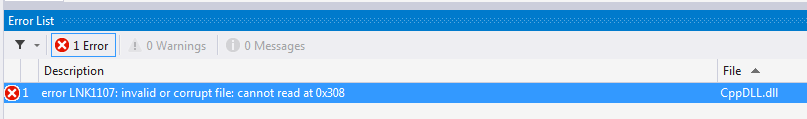
# 链接DLL时，必须链接.lib文件而不能是.dll文件（其实是隐式链接）

Dll是运行时需要的，lib是链接时需要的。

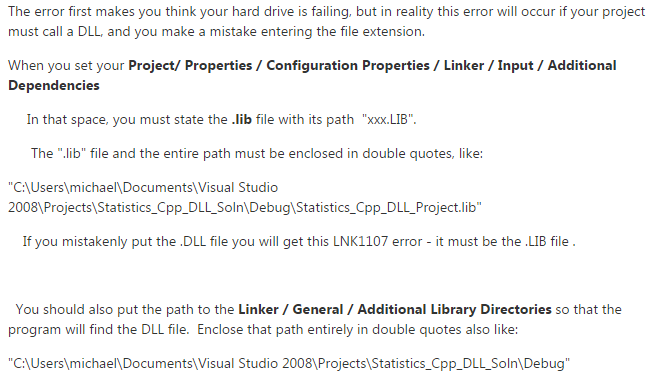
经测试，只Build（当然包含了链接）DemoConsole工程时，根本就不需要.dll文件，删了之后，build DemoConsole也不会出错，而且成功的生成了DemoConsole.exe

但是，如果提供的依赖是dll文件而不是lib文件，则build时会报错



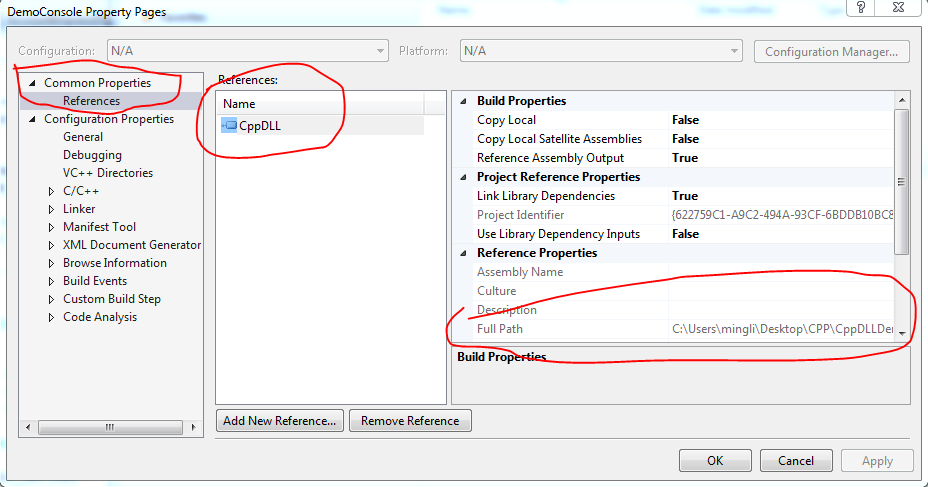


经查其原因就是



# Project添加Reference的方式

有两种，一种是Common Reference，可以直接Reference本solution的其他project



一种是在Additional Dependencies中去添加对应的lib文件