参考原文： https://www.cnblogs.com/wangqiguo/p/4348818.html

最近将两个开源C++项目编译成windows版本的时候遇到很多问题，关键是两个项目经过同事的修改之后，一个项目引用了另一个项目，两个项目的头文件中都有一些跨平台的关于数据类型，以及一些通用函数的定义，所以导致有冲突，编译的时候总是报错，报的最多的是“ 无法解析的外部符号 ”，经过近3天的折腾总算都通过了，这里是一些总结。

首先，关于VC中的lib□与linux下的静态库是不同的，在VC中编译动态库的时候会生成一个lib和一个对应的dll□使用者在使用的时候需要包含头文件以及连接到该lib□在发布最终程序的时候则需要将对应的dll拷贝到发布目录。当然也可以使用LoadLibrary的方式在程序中动态加载dll而不需要使用这个动态库生成的lib了。

如果是静态库，编译之后只会生成一个lib文件，该lib文件非常大，可能有几十M的大小，(而编译动态库的时候生成的lib可能只有几十KB或者几百KB)在使用这个静态库的lib的时候，也需要指定头文件，与对应的lib库文件，编译成功之后就可以直接运行，不需要拷贝额外的文件了。

另外如果A是静态库□B是静态库，并且B使用了A的接口，这个时候在编译B的时候只需要指定A的头文件就可以了，不需要指定A的库文件。如果有一个项目C编译成可执行文件□C使用了B中的接口，这个时候在编译C的时候，需要同时指定B的头文件(如果该头文件中又引用了A的头文件那可能也要同时指定A的头文件)，与B的lib库文件，以及A的lib库文件。 也就是说编译C的时候要指定之前所有依赖的lib文件。

在windows中编译动态库的时候，如果动态库中的函数需要给别人使用，那么这些函数或者类则需要被导出，具体如下，假设库的头文件为A.h□

复制代码

```
#    if defined LIB_A   //这个宏为这个A特有的宏
#        define DLLEXP __declspec(dllexport)
#    else
#        define DLLEXP __declspec(dllimport)
#    endif

class DLLEXP ExportClass{
//......
};
```

复制代码

如果在项目B中使用A库，那么项目B在引用A.h的时候，由于项目B没有定义LIB_A这个宏，所以实际上使用的是#define DLLEXP declspec(dllimport)这个定义，也就是说在B项目中，这个ExportClass类的声明变成导入了，表示该类是从外部库导入的类。 而在项目A中由于定义了LIB_A这个项目特有的宏，所以使用的是#define DLLEXP declspec(dllexport)这个定义，说明需要编译成导出给别人用的类。

如果是C语言的库给C++使用或者C++的库封装给C使用则除了要添加declspec(dllexport)导出声明之外，还需要添加 external “C” 的声明，该声明主要告诉编译器，编译的时候生成的函数的符号表按照C的规则来生成。 因为C编译器与C++编译器生成符号表的时候规则是不一样的。 那么编译的时候报告LINK错误，无法解析的外部符号，一般是下面几种原因造成的： 1. 最常见的情况是要么没有指定引用库的路径，或者没有指定所以依赖的库文件名字。 2. 如果正确指定了lib库路径，以及lib库名，那检查一下该lib中是否有该符号的实现，也就是说头文件中声明了该符号，但是该库文件中却没有具体的实现。 3. 如果库文件中确实实现了符号的定义，那么检查一下lib库的版本是否与正确(32位或者64位)。还有如果报告的是某一个函数无法解析，则要对比一下该函数在库中的实现与在头文件中的声明是否一致(特别是函数的参数个数与参数类型是否完全一致)。 4. 有一种情况就是在编译lib的时候，该lib是动态库，但是没有添加导出声明，导致该库中的函数并不对外导出(静态库不需要导出声明，加了反而会有问题)，那么使用者在链接的时候也会

报无法解析的符号。 5. 还有一种非常隐蔽的情况，这也是我遇到的情况，在项目A中将一些基本的数据类型做了typedef，例如类似下面的定义： typedef unsigned char uint8_t; typedef unsigned short int uint16_t; 然后A项目的导出函数 FUNC(uint8_t); 使用了该uint8_t，但是在B项目中对上述的 uint8_t 又做了另外一套定义 (如果A和B是两个开源项目则很有可能出现这种冲突)，如下： typedef unsigned int8 uint8_t;

那么在B中使用A的时候，使用的uint8_t是B的定义，实际上函数的声明变成了 FUNC(unsigned __int8) 但是在A的lib库文件的实现里面使用的是FUNC(unsigned char)也就是说该函数FUNC的声明与定义并不匹配，那么当然也会报告找不到符号了，这种情况一般是在两个开源项目混合使用的时候就会出现冲突。

6编译静态库的时候，如果静态库B引用了静态库A中的内容，此时在B的项目里面都不需要指定A的库路径，只需要指定A的相关头文件，就可以编译通过，如果里面有什么问题，那么会在最终使用B的项目的时候，链接的时候报出来。例如C项目使用了B，那么在编译C的时候需要同时添加A和B两个库，如果之前B使用A的过程中有问题的话，那么在编译C的时候就会报告LINK错误，而不是在编译B的时候报告(除非是语法错误)。

7. 如果项目C使用了B库与A库，但是B与A是有依赖关系的，那么在C的工程设置中，也要指定B和A的先后关系，否则也可能会报错。

8. 如果在连接项目的时候报告下面的错误：

无法找到外部符号 _CrtDbgReportW 或者是 error LNK2038: 检测到“_ITERATOR_DEBUG_LEVEL”的不匹配项: 值“2”不匹配值“0”(有可能是值“0”不匹配“2”)

这种错误一般是在Release版本的项目中使用了Debug的库，但是有时候明明看到我们编译的库都是Release版本的，使用那个库的时候却还是报告这个问题，这个现象可能是，编译那个库的时候，虽然选择的是Release方式编译，但是在项目的宏定义中却定义了_DEBUG宏，导致该还是会被认为是Debug的版本。