在C语言的头文件中经常发现extern "C"声明，像下面这样：

/\* DO NOT EDIT THIS FILE - it is machine generated \*/

#include <jni.h>

/\* Header for class com\_feifei\_hello\_Hello \*/

#ifndef \_Included\_com\_feifei\_hello\_Hello

#define \_Included\_com\_feifei\_hello\_Hello

#ifdef \_\_cplusplus

extern "C" {

#endif

/\*

\* Class: com\_feifei\_hello\_Hello

\* Method: getHello

\* Signature: ()Ljava/lang/String;

\*/

JNIEXPORT jstring JNICALL Java\_com\_feifei\_hello\_Hello\_getHello

(JNIEnv \*, jobject);

#ifdef \_\_cplusplus

}

#endif

#endif

此处的extern "C"声明告诉编译器“{ }”包围的代码应该按照C语言的语法进行编译。   
这是因为C++相比C多了很多语言上的特性，比如C++支持函数重载，但是C语言不支持，所以编译器对于 C++函数和C函数的签名形式不一样，编译器在编译C++函数的时候，会用函数名和函数形参结合的形式作   
为函数签名，编译C函数时只会使用函数名作为函数签名，那么如果存在 C++和C之间的函数调用，链接   
器在查找调用的函数时就找不到相同函数签名的函数。

extern "C"声明就告诉编译器它声明的函数按照C语言的风格进行函数签名，那么不管C调用C++函数， 还是C++调用C函数，被extern "C"声明的函数具有相同的函数签名，链接器就能够找到所调用的函数。

一、extern "C"的使用形式：

1、声明一个代码块

extern "C" {

int add( int , int);

int max(int , int);

}

2、只声明一个函数

extern "C" int add( int, int);

二、extern "C"的使用场景

1、需要实现一个C模块，如果该模块有可能被C++调用，那么在头文件中使用extern "C"声明：

#ifndef \_Included\_com\_feifei\_hello\_Hello

#define \_Included\_com\_feifei\_hello\_Hello

#ifdef \_\_cplusplus

extern "C" {

#endif

int max( int , int);

int min( int , int);

#ifdef \_\_cplusplus

}

#endif

#endif

2、需要在C++中调用一个已有的C模块，但是该C模块没有做extern "C"声明，在包含该 C模块头文件的时候做声明：

extern "C" {

# include "c\_modull.h"

}