原文：https://www.cnblogs.com/qihuanye-229110/p/11215620.html

extern 作用1：声明外部变量

例1：

创建一个工程，里面含有A.cpp和B.cpp两个简单的C++源文件：

//A.cpp:

int iRI;

int main()

{

    //.....

}

//B.cpp

int iRI;

在链接阶段，会出现重复定义的错误。

extern的原理很简单，就是告诉编译器：“你现在编译的文件中，有一个标识符虽然没有在本文件中定义，但是它是在别的文件中定义的全局变量”

//A.cpp:

extern int iRI;

int main()

{

    iRI = 64;

    //.....

}

//B.cpp

int iRI;

这样编译就能够通过。

extern int iRI; //并未分配空间，只是通知编译器，在其它文件定义过iRI。

extern 作用2：在C++文件中调用C方式编译的函数

例3：C++调用C定义的全局变量

//A.cpp:

extern int iRI;

int main()

{

    iRI = 64;

    //.....

}

//B.c

int iRI;

在链接时则会报iRI没有定义。为什么呢？

因为gcc看到A.cpp，就使用C++方式编译，看到B.c，就使用C方式编译。

所以在A.cpp中的

iRI编译为XXX@XXX\_iRI;

而B.c中

iRI编译为\_iRI;

所以在链接时，A.cpp想找到XXX@XXX\_iRI，当然找不到。

所以就需要告诉编译器，iRI是使用C方式编译的。

//A.cpp:

extern "C"

{

    int iRI;

}

int main()

{

    iRI = 64;

    //.....

}

//B.c

int iRI;

例4：C++调用C定义的function

同理

//A.cpp

extern "C"

{

    int functionA();

}

int main()

{

    functionA();

}

//B.c

int functionA()

{

    //....

}