C++项目中的extern "C" {}

2010-07-10 19:45 by 吴秦, 24935 阅读, 15 评论, 收藏, 编辑

引言

在用C++的项目源码中，经常会不可避免的会看到下面的代码：

```
#ifdef __cplusplus
extern "C" {
#endif

/*...*/

#ifdef __cplusplus
}
#endif
```

它到底有什么用呢，你知道吗？而且这样的问题经常会出现在面试or笔试中。下面我就从以下几个方面来介绍它：

- 1、#ifdef _cplusplus/#endif _cplusplus及发散
- 2、extern "C"
  - 2.1、extern关键字
  - 2.2、"C"

1、#ifdef _cplusplus/#endif _cplusplus及发散

在介绍extern "C"之前，我们来看下#ifdef _cplusplus/#endif _cplusplus的作用。很明显#ifdef/#endif、#ifndef/#endif用于条件编译，#ifdef _cplusplus/#endif _cplusplus——表示如果定义了宏_cplusplus，就执行#ifdef/#endif之间的语句，否则就不执行。

在这里为什么需要#ifdef _cplusplus/#endif _cplusplus呢？因为C语言中不支持extern "C"声明，如果你明白extern "C"的作用就知道在C中也没有必要这样做，这就是条件编译的作用！在.c文件中包含了extern "C"时会出现编译时错误。

既然说到了条件编译，我就介绍它的一个重要应用——**避免重复包含头文件**。还记得腾讯笔试就考过这个题目，给出类似下面的代码（下面是我最近在研究的一个开源web服务器——Mongoose的头文件mongoose.h中的一段代码）：

```
1 #ifndef MONGOOSE_HEADER_INCLUDED
2 #define    MONGOOSE_HEADER_INCLUDED
3
4 #ifdef __cplusplus
5 extern "C" {
```


C++静态库与动态库

C++对象模型

Python应用与实践

PureMVC（AS3）剖析：设计模式（二）

PureMVC（AS3）剖析：设计模式（一）

基于AIR Android应用开发1：环境搭建

最新评论

Re:C++项目中的extern "C" {}

最简单的一个用法是其他文件调用时候用 -- C++专家

Re:PyQt5应用与实践

顶一下，顺便占个沙发^_^ -- love _荣

Re:PureMVC（AS3）剖析：开篇

@梦の见可以这么用... -- 吴秦

Re:PureMVC（AS3）剖析：实例

@绝对刚把源码看了看，发现这是以前的版本，我修改了启动项是可以跑的，很不错~！！！！后面需要的人，可以建立新工程和当前项目合并，并修改启动项内容（删除原来的LinkUpGame.as）mxml内容如下..... -- love _荣

Re:PureMVC（AS3）剖析：实例

刚把源码看了看，发现这是以前的版本，我修改了启动项是可以跑的，很不错~！！！！后面需要的人，可以建立新工程和当前项目合并，并修改启动项内容（删除原来的LinkUpGame.as）mxml内容如下： ..... -- love _荣

日历

< 2010年7月 >

日	一	二	三	四	五	六
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17

随笔档案

2015年1月(1)

2014年12月(3)

2014年11月(1)

2014年2月(3)

```
#endif /* __cplusplus */

/*.............................
 * do something here
 *.............................
 */

#ifdef __cplusplus
}
#endif /* __cplusplus */

#endif /* MONGOOSE_HEADER_INCLUDED */
```

然后叫你说明上面宏#ifndef/#endif的作用？为了解释一个问题，我们先来看两个事实：

- 这个头文件mongoose.h可能在项目中被多个源文件包含（#include "mongoose.h"），而对于一个大型项目来说，这些冗余可能导致错误，因为一个头文件包含类定义或inline函数，在一个源文件中mongoose.h可能会被#include两次（如，a.h头文件包含了mongoose.h，而在b.c文件中#include a.h和mongoose.h）——这就会出错（在同一个源文件中一个结构体、类等被定义了两次）。
- 从逻辑观点和减少编译时间上，都要求去除这些冗余。然而让程序员去分析和去掉这些冗余，不仅枯燥且不太实际，**最重要的是有时候又需要这种冗余来保证各个模块的独立**。

为了解决这个问题，上面代码中的

#ifndef MONGOOSE_HEADER_INCLUDED

#define MONGOOSE_HEADER_INCLUDED

/*.....*/

18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

随笔分类

.NET 2.0配置解谜系列(9)
.NET(C#) Internals (10)
【日常小记】(3)
【转载】(2)
Android开发之旅(18)
as3(1)
C/C++ Internals(15)
cocos2d-x(1)
JavaScript(1)
nginx(2)
PureMVC（AS3）剖析(5)
Python(5)
Unix/Linux下编程(8)
服务器开发(3)
基于AIR Android应用开发(1)
客户端开发(1)
数据库(4)
网页游戏开发(23)
源码剖析：DotText源码学习(2)
源码剖析：Mongoose(5)

推荐排行榜

2013年11月(1)
2013年10月(1)
2013年9月(1)
2013年5月(1)
2013年3月(2)
2013年2月(2)
2013年1月(2)
2012年12月(4)
2012年11月(1)
2012年8月(1)
2012年4月(1)
2012年3月(2)
2012年1月(1)
2011年7月(1)
2011年6月(5)
2011年5月(3)
2011年3月(2)
2011年2月(1)
2011年1月(2)
2010年12月(6)
2010年10月(1)
2010年9月(4)
2010年7月(12)
2010年6月(4)
2010年5月(14)
2010年4月(12)
2010年3月(10)

#endif /* MONGOOSE_HEADER_INCLUDED */

就起作用了。如果定义了MONGOOSE_HEADER_INCLUDED，#ifndef/#endif之间的内容就被忽略掉。因此，编译时第一次看到mongoose.h头文件，它的内容会被读取且给定MONGOOSE_HEADER_INCLUDED一个值。之后再次看到mongoose.h头文件时，MONGOOSE_HEADER_INCLUDED就已经定义了，mongoose.h的内容就不会再次被读取了。

2、extern "C"

首先从字面上分析extern "C"，它由两部分组成——extern关键字、"C"。下面我就从这两个方面来解读extern "C"的含义。

2.1、extern关键字

在一个项目中必须保证函数、变量、枚举等在所有的源文件中保持一致，除非你指定定义为局部的。首先来一个例子：

```
1 //file1.c:
2     int x=1;
3     int f(){do something here}
4 //file2.c:
5     extern int x;
6     int f();
7     void g(){x=f();}
```

在file2.c中g()使用的x和f()是定义在file1.c中的。extern关键字表明file2.c中x，仅仅是一个变量的声明，其并不是在定义变量x，并未为x分配内存空间。变量x在所有模块中作为一种全局变量只能被定义一次，否则会出现连接错误。但是可以声明多次，且声明必须保证类型一致，如：

1. Android开发之旅：环境搭建及HelloWorld(135)
2. HTTP协议及其POST与GET操作差异 & C#中如何使用POST、GET等(133)
3. 字符集和字符编码（Charset & Encoding）(122)
4. Linux Socket编程（不限Linux）(81)
5. 浏览器缓存机制(58)
6. HTTP Keep-Alive模式(51)
7. Linux多线程编程（不限Linux）(49)
8. Android 开发之旅：view的几种布局方式及实践(42)
9. C++项目中的extern "C" {}(31)
10. Android开发之旅：应用程序基础及组件(31)

阅读排行榜

1. Android开发之旅：环境搭建及HelloWorld(950840)
2. Linux Socket编程（不限Linux）(163201)
3. 字符集和字符编码（Charset & Encoding）(119634)
4. Android 开发之旅：view的几种布局方式及实践(83355)
5. Android开发之旅：android架构(79338)
6. Android开发之旅：HelloWorld项目的目录结构(63710)

```
//file1.c:
    int x=1;
    int b=1;
    extern c;
//file2.c:
    int x;// x equals to default of int type 0
    int f();
    extern double b;
    extern int c;
```

在这段代码中存在着这样的三个错误：

1. x被定义了两次
2. b两次被声明为不同的类型
3. c被声明了两次，但却没有定义

回到extern关键字，extern是C/C++语言中表明**函数**和**全局变量**作用范围（可见性）的关键字，该关键字告诉编译器，其声明的函数和变量可以在本模块或其它模块中使用。通常，在模块的头文件中对本模块提供给其它模块引用的函数和全局变量以关键字extern声明。例如，如果模块B欲引用该模块A中定义的全局变量和函数时只需包含模块A的头文件即可。这样，模块B中调用模块A中的函数时，在编译阶段，模块B虽然找不到该函数，但是并不会报错；它会在连接阶段中从模块A编译生成的目标代码中找到此函数。

与extern对应的关键字是 static，被它修饰的全局变量和函数只能在本模块中使用。因此，一个函数或变量只可能被本模块使用时，其不可能被extern “C”修饰。

2.2、"C"

7. HTTP协议及其POST与GET操作差异 & C#中如何使用POST、GET等(55767)
8. Linux多线程编程（不限Linux）(52777)
9. Android开发之旅: Intents和Intent Filters（理论部分）(37769)
10. Android开发之旅：应用程序基础及组件(37225)

系列索引帖

.NET 2.0配置解谜系列索引（完结）

典型的，一个C++程序包含其它语言编写的部分代码。类似的，C++编写的代码片段可能被使用在其它语言编写的代码中。不同语言编写的代码互相调用是困难的，甚至是同一种编写的代码但不同的编译器编译的代码。例如，不同语言和同种语言的不同实现可能会在注册变量保持参数和参数在栈上的布局，这个方面不一样。

为了使它们遵守统一规则，可以使用extern指定一个编译和连接规约。例如，声明C和C++标准库函数strcyp()，并指定它应该根据C的编译和连接规约来链接：

```
1 extern "C" char* strcpy(char*,const char*);
```

注意它与下面的声明的不同之处：

```
1 extern char* strcpy(char*,const char*);
```

下面的这个声明仅表示在连接的时候调用strcpy()。

extern "C"指令非常有用，因为C和C++的近亲关系。**注意：extern "C"指令中的C，表示的一种编译和连接规约，而不是一种语言。C表示符合C语言的编译和连接规约的任何语言，如Fortran、assembler等。**

还有要说明的是，extern "C"指令仅指定编译和连接规约，但不影响语义。例如在函数声明中，指定了extern "C"，仍然要遵守C++的类型检测、参数转换规则。

再看下面的一个例子，为了声明一个变量而不是定义一个变量，你必须在声明时指定extern关键字，但是当你又加上了"C"，它不会改变语义，但是会改变它的编译和连接方式。

如果你有很多语言要加上extern "C"，你可以将它们放到extern "C"{ }中。

2.3、小结extern "C"

通过上面两节的分析，我们知道extern "C"的真实目的是实现**类C和C++的混合编程**。在C++源文件中的语句前面加上extern "C"，表明它按照类C的编译和连接规约来编译和连接，而不是C++的编译的连接规约。这样在类C的代码中就可以调用C++的函数or变量等。（注：我在这里所说的类C，代表的是跟C语言的编译和连接方式一致的所有语言）

3、C和C++互相调用

我们既然知道extern "C"是实现的类C和C++的混合编程。下面我们就分别介绍如何在C++中调用C的代码、C中调用C++的代码。首先要明白C和C++互相调用，你得知道它们之间的编译和连接差异，及如何利用extern "C"来实现相互调用。

3.1、C++的编译和连接

C++是一个面向对象语言（虽不是纯粹的面向对象语言），它支持函数的重载，重载这个特性给我们带来了很大的便利。为了支持函数重载的这个特性，C++编译器实际上将下面这些重载函数：

```
1 void print(int i);
2 void print(char c);
3 void print(float f);
4 void print(char* s);
```

编译为：

```
1 _print_int
2 _print_char
```

```
3 _print_float
4 _pirnt_string
```

这样的函数名，来唯一标识每个函数。注：不同的编译器实现可能不一样，但是都是利用这种机制。所以当连接是调用print(3)时，它会去查找_print_int(3)这样的函数。下面说个题外话，正是因为这点，重载被认为不是多态，多态是运行时动态绑定（“一种接口多种实现”），如果硬要认为重载是多态，它顶多是编译时“多态”。

C++中的变量，编译也类似，如全局变量可能编译g_xx，类变量编译为c_xx等。连接是也是按照这种机制去查找相应的变量。

3.2、C的编译和连接

C语言中并没有重载和类这些特性，故并不像C++那样print(int i)，会被编译为_print_int，而是直接编译为_print等。因此如果直接在C++中调用C的函数会失败，因为连接是调用C中的print(3)时，它会去找_print_int(3)。因此extern "C"的作用就体现出来了。

3.3、C++中调用C的代码

假设一个C的头文件cHeader.h中包含一个函数print(int i)，为了在C++中能够调用它，必须要加上extern关键字（原因在extern关键字那节已经介绍）。它的代码如下：

```
1 #ifndef C_HEADER
2 #define C_HEADER
3
4 extern void print(int i);
5
```

```
#endif C_HEADER
```

相对应的实现文件为cHeader.c的代码为：

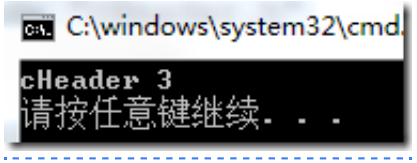
```
#include <stdio.h>
#include "cHeader.h"
void print(int i)
{
    printf("cHeader %d\n",i);
}
```

现在C++的代码文件C++.cpp中引用C中的print(int i)函数：

```
extern "C"{
#include "cHeader.h"
}

int main(int argc,char** argv)
{
    print(3);
    return 0;
}
```

执行程序输出：



3.4、C中调用C++的代码

现在换成在C中调用C++的代码，这与在C++中调用C的代码有所不同。如下在

cppHeader.h头文件中定义了下面的代码：

```
#ifndef CPP_HEADER
#define CPP_HEADER

extern "C" void print(int i);

#endif CPP_HEADER
```

相应的实现文件cppHeader.cpp文件中代码如下：

```
#include "cppHeader.h"

#include <iostream>
using namespace std;
void print(int i)
{
    cout<<"cppHeader "<<i<<endl;
}
```

在C的代码文件c.c中调用print函数：

```
extern void print(int i);
int main(int argc,char** argv)
{
    print(3);
    return 0;
}
```

注意在C的代码文件中直接#include "cppHeader.h"头文件，编译出错。而且如果不加extern int print(int i)编译也会出错。

4、C和C++混合调用特别之处函数指针

当我们C和C++混合编程时，有时候会用一种语言定义函数指针，而在应用中将函数指针指向另一中语言定义的函数。如果C和C++共享同一中编译和连接、函数调用机制，这样做是可以的。然而，这样的通用机制，通常不然假定它存在，因此我们必须小心地确保函数以期望的方式调用。

而且当指定一个函数指针的编译和连接方式时，函数的所有类型，包括函数名、函数引入的变量也按照指定的方式编译和连接。如下例：

```
typedef int (*FT) (const void* ,const void*);//style of C++

extern "C"{
    typedef int (*CFT) (const void*,const void*);//style of
C
    void qsort(void* p,size_t n,size_t sz,CFT cmp);//style
of C
}

void isort(void* p,size_t n,size_t sz,FT cmp);//style of
C++
void xsort(void* p,size_t n,size_t sz,CFT cmp);//style of C

//style of C
extern "C" void ysort(void* p,size_t n,size_t sz,FT cmp);

int compare(const void*,const void*);//style of C++
extern "C" ccomp(const void*,const void*);//style of C

void f(char* v,int sz)
{
    //error,as qsort is style of C
    //but compare is style of C++
```

```
    qsort(v,sz,1,&compare);
    qsort(v,sz,1,&ccomp);//ok

    isort(v,sz,1,&compare);//ok
    //error,as isort is style of C++
    //but ccomp is style of C
    isort(v,sz,1,&ccopm);
}
```

注意：typedef int (*FT) (const void* ,const void*)，表示定义了一个函数指针的别名FT，这种函数指针指向的函数有这样的特征：返回值为int型、有两个参数，参数类型可以为任意类型的指针（因为为void*）。

最典型的函数指针的别名的例子是，信号处理函数signal，它的定义如下：

```
typedef void (*HANDLER)(int);
HANDLER signal(int ,HANDLER);
```

上面的代码定义了信函处理函数signal，它的返回值类型为HANDLER，有两个参数分别为int、HANDLER。 这样避免了要这样定义signal函数：

```
void (*signal (int ,void(*)(int) ))(int)
```

比较之后可以明显的体会到typedef的好处。

作者：吴秦
出处：http://www.cnblogs.com/skynet/

吴秦
关注 - 18
粉丝 - 2236
31 1
荣誉：推荐博客
+加关注

(请您对文章做出评价)

« 上一篇：STL之Map
» 下一篇：汇编中参数的传递和堆栈修正【转载】

分类: C/C++ Internals

ADD YOUR COMMENT

#1楼 kwjlk
2010-07-10 20:11
飘过~~
支持(0) 反对(0)

#2楼 真名士
2010-07-10 20:48
文章不错，支持一下！
支持(0) 反对(0)

#3楼 SmartJJ
2010-07-10 22:33
文章不错，支持一下！
支持(0) 反对(0)

#4楼 zhaobo
2010-07-10 22:36
文章不错，支持一下！

支持(0) 反对(0)

#5楼 cpp
2010-07-10 23:59

写的不错，顶！
对c调用c++代码说的不够详细。

支持(0) 反对(0)

#6楼 chinese_submarine
2010-07-14 14:05

文章不错，不过内容比较杂，最后关于指针的部分有些乱。。。

支持(0) 反对(0)

#7楼 hahahuang[未注册用户]
2010-10-26 21:05

好文章，羡慕楼主的实力啊！

#8楼 hahahuang[未注册用户]
2010-10-26 21:06

好文章！羡慕楼主的实力

#9楼 fishnet [未注册用户]
2011-02-22 11:41

extern int print(int i);
你上面定义的方法和C文件中的方法声明不一致。
如果要是加上编译说明就更好了

#10楼 fishnet [未注册用户]
2011-02-22 12:01

还有在c文件中不加extern是不会报错的

#11楼 sudo

2011-08-18 11:31

总算开明白了 extern “C"，谢谢。

支持(0) 反对(0)

#12楼 Ranger98

2012-03-07 16:27

看面试题看到了这个问题，以前一直不知道，谢谢

支持(0) 反对(0)

#13楼 Aoysme

2012-03-27 15:02

3.3、C++中调用C的代码

假设一个C的头文件cHeader.h中包含一个函数print(int i)，为了在C++中能够调用它，必须要加上extern关键字（原因在extern关键字那节已经介绍）。它的代码如下：

"必须要加上extern关键字",不需要加吧

支持(0) 反对(0)

#14楼 东南小石头

2014-09-24 21:14

好赞！

支持(0) 反对(0)

#15楼 C++专家

2015-02-12 12:31

最简单的一个用法是其他文件调用时候用

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

最新IT新闻:

· 滴滴快的将在数周内完成合并
· 窝窝团称上市仍在走流程
· 三星声明准备直接使用人民币进行贸易结算
· LG寻求合作伙伴推广OLED电视
· 苹果iTunes被判专利侵权 需赔5.329亿美元
» 更多新闻...

最新知识库文章:

· 在线数据迁移经验：如何为正在飞行的飞机更换引擎
· HHVM 是如何提升 PHP 性能的？
· Web API设计方法论
· Bitmap的秘密
· 我该如何向非技术人解释SQL注入？
» 更多知识库文章...