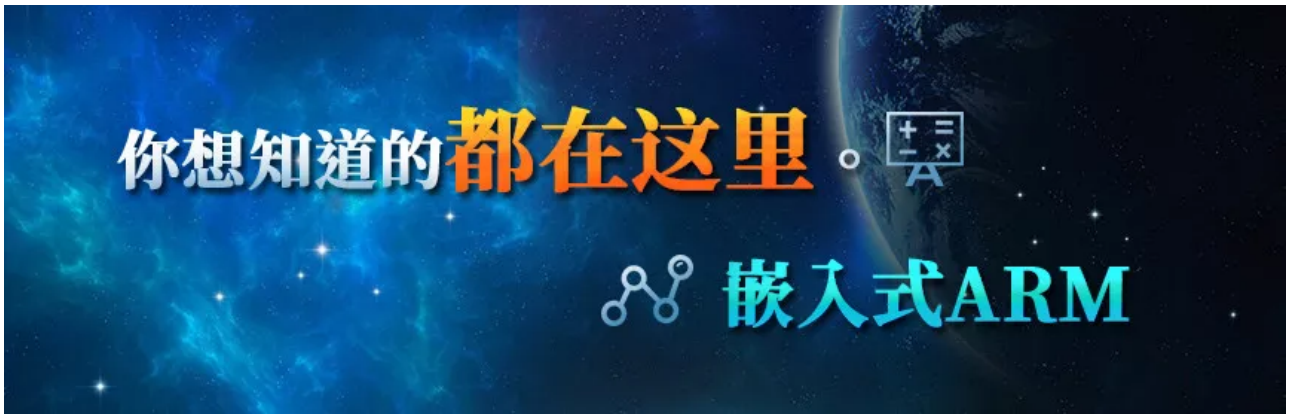
长见识：你真的知道C语言里extern "C" 的作用吗?

嵌入式ARM 2020-08-22



经常在C语言的头文件中看到下面的代码：

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

// all of your legacy C code here

#ifdef __cplusplus
}
#endif
```

这通常用于 C++ 和 C 混合编程的时候，为了防止 C++ 的编译器在编译 C 文件的时候出现错误；
众所周知， C++ 可以进行函数名重载，但是 C 则没有这种功能，那这和 extern "C" 又有什么关系呢?
先看下面这个表格，如下所示；

语言	描述
C	函数名可以作为唯一**ID**和代码段的程序建立联系

语言	描述
C++	因为重载的关系，函数名符号会被破坏，从而会根据函数的参数不同而重新生成函数符号

未添加 **extern "C"**

test.h

```
#ifndef TEST_H
#define TEST_H

void foo1(void);
void foo2(void);
void foo3(int i);

#endif
```

test.c

```
void foo1(void){}
void foo2(void) {}
void foo3(int i){}

int main(int argc,char** argv){

 foo1();
 foo2();
 foo3(1);
 return 0;
}
```

编译这两个文件，生成 `test.o` 文件，通过 `objdump` 查看函数符号；

```
g++ -c test.c test.h
objdump -t test.o
```

可以看到函数符号已经被编译器修改了；

添加**extern "C"**

`test.h`

```
#ifndef TEST_H
#define TEST_H

#ifdef __cplusplus
extern "C" {
#endif
void foo1(void);
void foo2(void);
void foo3(int i);

#ifdef __cplusplus
}
#endif

#endif
```

test.c

```
#ifdef __cplusplus
extern "C" {
#endif
void foo1(void){}
void foo2(void) {}
void foo3(int i){}
#ifdef __cplusplus
}
#endif

int main(int argc,char** argv){

 foo1();
 foo2();
 foo3(1);
 return 0;
}
```

编译这两个文件，生成 `test.o` 文件，通过 `objdump` 查看函数符号；

```
g++ -c test.c test.h
objdump -t test.o
```

这时候函数符号是正确的；

`extern "C"` 是告诉 `C++` 的编译器不要打我这些C函数的主意。

-END-

来源 ： 小麦大叔

推荐阅读

【01】国内为什么写不出操作系统和编程语言?

【02】史上最全Linux/C/C++思维导图，理清思路全靠它了！

【03】如何在Linux上恢复误删除的文件或目录

【04】超实用！分享5个基本Linux命令行工具的现代化替代品

【05】从零基础到Linux开发，我是这样一步步过来的

嵌入式ARM

撒贝宁怼“95后”女生上热搜：你以为的耿直，其实就是情商低

行动派DreamList

女子扔下一坨“烂拖布”走了，老板救助后发现它美炸了！

狗与爱的世界