

探索C 的秘密之详解extern PDF转换可能丢失图片或格式，建议阅读原文

[https://www.100test.com/kao\\_ti2020/245/2021\\_2022\\_\\_E6\\_8E\\_A2\\_E7\\_B4\\_A2C\\_\\_\\_E7\\_c97\\_245966.htm](https://www.100test.com/kao_ti2020/245/2021_2022__E6_8E_A2_E7_B4_A2C___E7_c97_245966.htm) 时常在cpp的代码之中看到

这样的代码: `#ifdef __cplusplus extern "C" { #endif //一段代码 #ifdef __cplusplus } #endif` 这样的代码到底是什么意思呢？首先，`__cplusplus`是cpp中的自定义宏，那么定义了这个宏的话表示这是一段cpp的代码，也就是说，上面的代码的含义是:如果这是一段cpp的代码，那么加入`extern "C" {和}`处理其中的代码。要明白为何使用`extern "C"`，还得从cpp中对函数的重载处理开始说起。在c中，为了支持重载机制，在编译生成的汇编码中，要对函数的名字进行一些处理，加入比如函数的返回类型等等.而在C中，只是简单的函数名字而已，不会加入其他的信息.也就是说:C和C对产生的函数名字的处理是不一样的.比如下面的一段简单的函数，我们看看加入和不加入`extern "C"`产生的汇编代码都有哪些变化: `int f(void) { return 1. }` 在加入`extern "C"`的时候产生的汇编代码是: `.file "test.cxx" .text .align 2 .globl _f .def _f .scl 2 .type 32 .endif _f: pushl movl %esp , movl $1 , popl ret` 但是不加入了`extern "C"`之后 `.file "test.cxx" .text .align 2 .globl __Z1fv .def __Z1fv .scl 2 .type 32 .endif __Z1fv: pushl movl %esp , movl $1 , popl ret` 两段汇编代码同样都是使用`gcc -S`命令产生的，所有的地方都是一样的，唯独是产生的函数名，一个是`_f`，一个是`__Z1fv`。明白了加入与不加入`extern "C"`之后对函数名称产生的影响，我们继续我们的讨论:为什么需要使用`extern "C"`呢？C之父在设计C之时，考虑到当时已经存在了大量的C代码，为了支持原来的C代码和已

经写好C库，需要在C中尽可能的支持C，而extern "C"就是其中的一个策略。试想这样的情况：一个库文件已经用C写好了而且运行得很良好，这个时候我们需要使用这个库文件，但是我们需要使用C来写这个新的代码。如果这个代码使用的是C的方式链接这个C库文件的话，那么就会出现链接错误。我们来看一段代码：首先，我们使用C的处理方式来写一个函数，也就是说假设这个函数当时是用C写成的：`//f1.c extern "C" { void f1() { return. } }`编译命令是：`gcc -c f1.c -o f1.o`产生了一个叫f1.o的库文件。再写一段代码调用这个f1函数：`// test.cxx //这个extern表示f1函数在别的地方定义，这样可以通过//编译，但是链接的时候还是需要//链接上原来的库文件. extern void f1(). int main() { f1(). return 0. }`通过`gcc -c test.cxx -o test.o`产生一个叫test.o的文件。然后，我们使用`gcc test.o f1.o`来链接两个文件，可是出错了，错误的提示是：`test.o(.text 0x1f):test.cxx: undefine reference to f1() 也就是说，在编译test.cxx的时候编译器是使用C的方式来处理f1()函数的，但是实际上链接的库文件却是用C的方式来处理函数的，所以就会出现链接过不去的错误：因为链接器找不到函数。因此，为了在C代码中调用用C写成的库文件，就需要用extern "C"来告诉编译器：这是一个用C写成的库文件，请用C的方式来链接它们。比如，现在我们有了一个C库文件，它的头文件是f.h，产生的lib文件是f.lib，那么我们如果要在C中使用这个库文件，我们需要这样写：extern "C" { #include "f.h" }回到上面的问题，如果要改正链接错误，我们需要这样子改写test.cxx：extern "C" { extern void f1(). } int main() { f1(). return 0. }重新编译并且链接就可以过去了。总结 C和C对函数的处理方式是不同的.extern "C"是使C能`

够调用C写作的库文件的一个手段，如果要对编译器提示使用C的方式来处理函数的话，那么就要使用extern "C"来说明。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 [www.100test.com](http://www.100test.com)