

高手讲解：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_ti2020/206/2021_2022__E9_AB_98_E6_89_8B_E8_AE_B2_E8_c103_206582.htm 时常在cpp的代码之中看到这样的代码:以下是引用片段：`#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:以下是引用片段：`int f(void) { return 1. }` 在加入extern "C"的时候产生的汇编代码是:以下是引用片段：`.file "test.cxx" .text .align 2 .globl _f .def _f .scl 2 .type 32 .endef _f: pushl euml.p movl $1, %eax popl euml.p movl %esp, %eax 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。 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