|  |  |
| --- | --- |
| 1. 声明一个非c++函数 2. 链接指示与头文件 3. 指向extern c函数的指针 4. 链接指示对整个声明都有效 5. 导出C++函数到其它语言 6. 重载函数与链接指示 | 链接指示  单语句链接指示  复合语句链接指示  Extern “c” size\_t strlen(const char \*)  Extern “C”{  #include <string.h>  }  Extern “c” void (\*Pf)(int);  Extern “c” void f1(void(\*)(int));  Extern “c” typedef void Fc(int); |
| 开头：  C+程序有时需要调用其它语言编写的函数。  编译器检查其调用方式与处理普通C+函数方式相同，但是生成的代码有所区别，c+使用链接指示指出任意非c+函数所用的语言。  Note：需要c+有权访问该语言的编译器并与c+编译器兼容   1. 声明一个非c++函数 2. 单个的或复合的 3. 链接指示与头文件 4. 复合的花括号是将适用于该链接指示的多个声明聚合在一起，否则花括号就会被忽略，花括号中声明的函数名字就是可见的，就好像在花括号之外声明一样 5. 所以头文件中所有普通函数声明都被认为是链接指示的语言编写的 6. 链接指示可以嵌套，所以头文件中包含带自带链接指示的函数，则该函数的链接不受影响   Note：c+可以将从c语言继承而来的函数定义成c函数   1. 指向extern c函数的指针 2. Extern “c” void (\*Pf)(int); 3. 与指向c++函数的指针是不同的类型，不能互相赋值初始化 4. 链接指示对整个声明都有效 5. Extern “c” void f1(void(\*)(int));   此时形参也是c函数了  2）若只希望形参是c函数，而函数本身是c+函数，则需使用类型别名  Extern “c” typedef void Fc(int);  Void f2(Fc \*) f2是c+，fc是指向c函数的指针   1. 导出C++函数到其它语言 2. 通过链接指示可以让c+函数在其它语言编写的程序中可用   可被多种语言共享的函数的返回类型或形参类型收到很多限制，不能把c+类的对象给c程序，因为c无法理解构造函数等   1. 重载函数与链接指示 2. 链接指示与重载函数的相互作用依赖于目标语言，若目标语言支持重载函数，则为该语言实现链接指示的编译器很支持重载这些c+函数 3. C语言不支持重载，所以一个c链接指示只能用于一组重载函数中的一个 | |