50 Using Dynamic Libraries in C++

C++中使用动态库

1. 动态链接

动态链接发生在*runtime*（运行时），而静态链接是在编译时发生的。
当你编译一个静态库的时候，将其链接到可执行文件，也就是应用程序，或者链接到一个动态库。就像你取出了那个静态库的内容，然后你把那些内容放入到其它的二进制数据中，实际在你的动态库中或者在你的可执行文件中。

有很多地方可以优化，因为编译器和链接器现在完全知道静态链接时实际进入应用程序的代码（静态链接允许更多的优化发生）。
而动态链接发生在运行时，所以只有你真正启动你的可执行文件时，你的动态链接库才会被加在，所以它实际上不是可执行文件的一部分（运行时将一个额外的文件加载到内存中）。

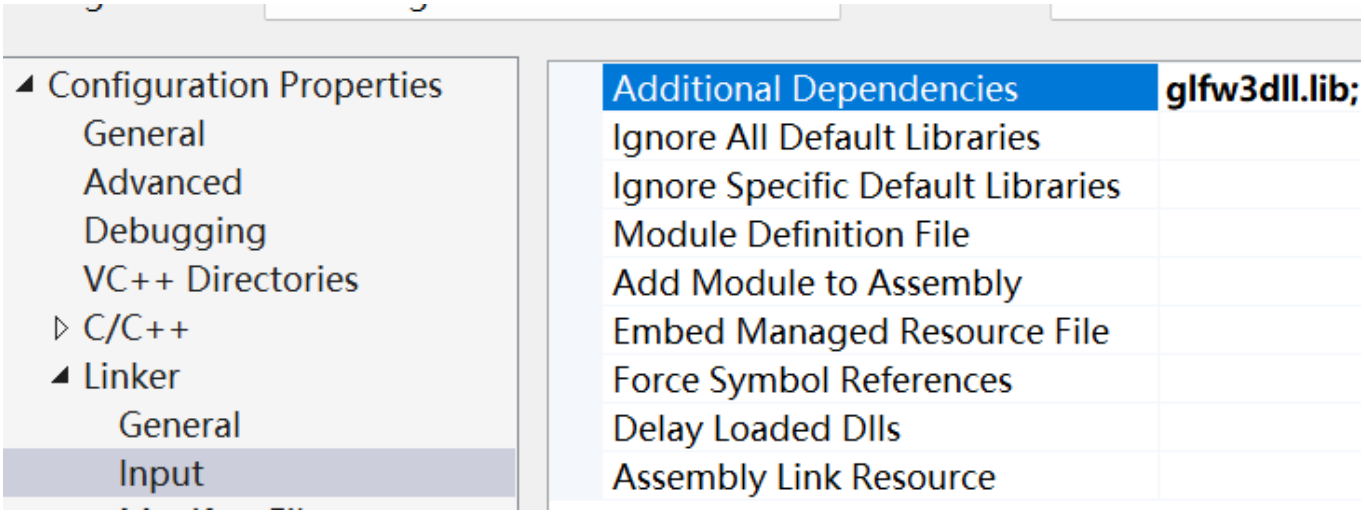
现在可执行文件在实际运行前就需要具备某些库、某些动态库、某些外部文件，这就是为什么你在Windows上启动一个应用程序时，可能看到一个错误消息弹出：需要dll、没有找到dll.....
这是动态链接的一种形式，可执行文件知道动态链接库的存在，把动态库作为一项需要，虽然动态库仍然是一个单独的文件，一个单独的模块，并在运行时加载。你也可以完全动态地加载动态库，这样可执行文件就与动态库完全没有任何关系了，但是在你的可执行文件中，你可以查找并在运行时加载某些动态库，然后获得某些函数指针或者动态库里你想要的东西，然后使用那个动态库。

对于动态库，请记住两个版本。
第一个是"静态的"动态库的版本，我的应用程序现场需要这个动态链接库，我已经知道里面有什么函数，我可以用什么。
第二个版本是我想任意加载这个动态库，我甚至不需要知道里面有什么，但我想取出一些东西或者做很多事。
这两种动态库都有很好的用途，先专注看第一种：我知道我的应用程序需要这个库，但我要动态地链接它。

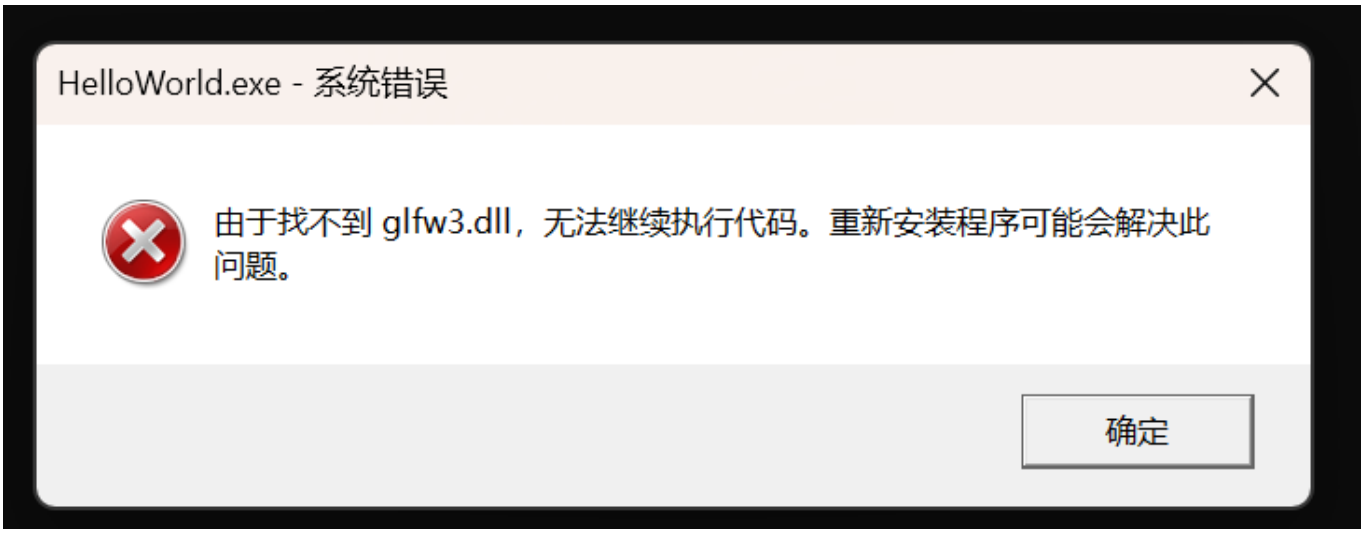
如果你要对比静态和动态链接的话，对于函数之类的声明，动态链接时实际有些不同。但GLFW像大多数库一样，同时支持静态和动态链接，使用相同的头文件。

见上节课，`.dll`和`dll.lib`同时编译是非常重要的，因为如果你尝试使用不同的静态库，在运行时链接到dll，你可能会得到不匹配的函数和错误类型的内存地址，函数指针不会正常工作。

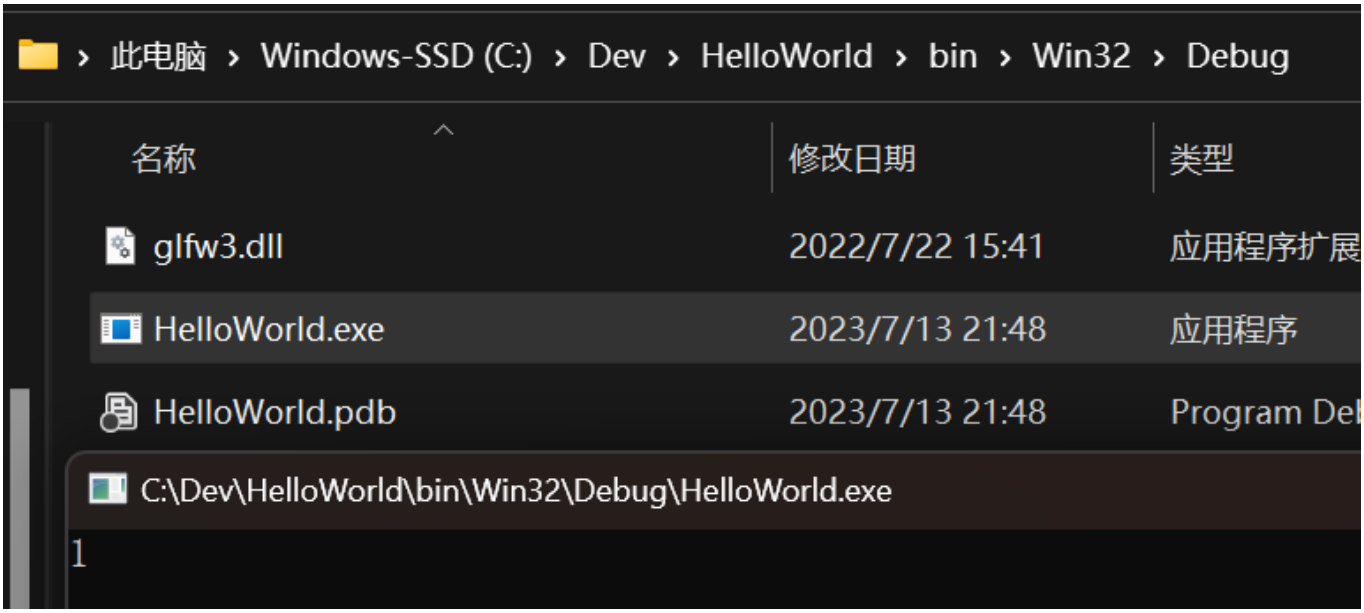
载入`dll.lib`



build并运行程序：



代码不能继续，因为glfw3.dll没有被找到，这时候我们就得告诉程序这个dll文件在哪儿了。
简单的做法是将dll文件和可执行文件放一起，同样的位置：



运行成功。