# CMake链接DLL可行实践

2024-01-16 19:38 2113 0 1 2032 4:03 ~ 6:46

实践应用 C++ 实践应用 CMAKE

DLL作为Windows下的动态库格式，其同Linux等平台的动态库稍有不同，比如对MSVC编译器需要导出符号等。本文简述一种情况，即仅有dll文件及对应头文件时应当如何在CMake中完成链接。

首先必须提示的一点是，DLL文件是程序运行时会调用的文件。编译器在编译链接时并不关心DLL文件在哪，而是会去查找DLL文件的导出库。该导出库记载了DLL中所有暴露的导出符号（类、函数、变量等），以提供给其它程序调用使用。编译器会根据指令链接该导出库到程序，随后程序运行时方才会去查找DLL文件中对应符号所记载的内容（函数中的程序、类的内容、变量的值等）。

Windows下，这个导出库同静态库一样是以.lib作为后缀名的。MSVC在编译时会同时输出dll和这个.lib文件。一般动态库的开发者会同时发布DLL、LIB、头文件给用户使用。而有时我们可能只能获取到DLL和头文件。这种情况下，我们可以使用工具重新生成.lib链接库用于链接。可行实践是使用两个工具，

gendef和lib。前者在绝大多数MinGW64包中提供；后者在MSVC套件中提供，安装Visual Studio即可在开发者命令行中使用。

首先，我们使用gendef获取DLL文件中的所有导出符号，并以特定格式保存至一种.def文件中：

```cmd
gendef <DLL文件名.dll>
```

程序会输出一个与DLL同名的def文件。该文件记载了DLL文件中所有导出的符号。这些符号同样可以使用MSVC工具dumpbin列出。

接下来，我们根据def文件中所记载的导出符号生成lib导出库：

```cmd
lib /def:<DEF文件.def> /machine:x64 /out:<DLL文件名.lib>
```

machine即为希望编译的目标平台架构。

现在我们可以在CMakeLists.txt中尝试链接该库了。在本实践中，我们尝试将其注册为CMake对象：

```
add_library(库名 SHARED IMPORTED)
set_target_properties(库名 PROPERTIES
    IMPORTED_LOCATION DLL文件路径/DLL文件名.dll
    IMPORTED_IMPLIB LIB文件路径/LIB文件名.lib
)
target_link_libraries(程序名 库名)
```

提醒注意的是，`set_target_properties` 所设置的 `IMPORTED_LOCATION` 和 `IMPORTED_IMPLIB` 属性不支持相对路径，而必须使用绝对路径，否则可能导致Ninja等构建系统报错找不到链接的程序位置。但在

CMakeLists.txt中写绝对路径绝对不是一个好主意，毕竟您需要分发您的项目给别人使用。一个好办法是使用 `${CMAKE_SOURCE_DIR}` 宏。该宏记录了CMakeLists.txt所在的绝对路径，只需借此添加相对位置即可，例如：

```
add_library(库名 SHARED IMPORTED)
set_target_properties(库名 PROPERTIES
    IMPORTED_LOCATION ${CMAKE_SOURCE_DIR}/external/DLL文件名.dll
    IMPORTED_IMPLIB ${CMAKE_SOURCE_DIR}/external/LIB文件名.lib
)
target_link_libraries(程序名 库名)
```

这样我们便成功链接DLL了。在项目中include库的头文件，就可以使用了。其它的一些可能的情况如下：

1. 无法解析的外部符号。如果实在查不出问题来，需要怀疑您链接的DLL是用MinGW GCC编译的，而不是MSVC。这种情况下MSVC无法直接链接使用MinGW编译的DLL。可换用MinGW尝试解决或动用一些方法转为MSVC DLL。后者暂不在本文讨论范围内。
2. 务必注意，这样做程序会优先根据IMPORTED_LOCATION所给出的位置查找DLL。故在本机上运行，不把DLL复制到程序根目录也不会报错。但如果将程序放到其它计算机上使用，就会导致找不到DLL文件而报找不到DLL或0xc000007b。故此不要忘记将DLL复制到程序目录下。CMake的install指令可以协助完成这一工作。



posted @ 2024-01-16 19:38 IZWB-003 阅读(2113) 评论(0) 编辑 收藏 举报