CMake 教程第五集：构建 Hello World 的共享库与静态库

一、前言

在前几节课中，我们已经构建了一个基本的 **Hello World** 项目。这次，我们将进一步深入，构建 **Hello World** 的共享库和静态库，并通过函数 **HelloFunc()** 实现向终端输出字符串 **Hello World**。这一节将不再纠结于简单的打印语句，而是将重点放在共享库和静态库的构建及安装上。

二、本节任务

1. 创建一个静态库和动态库，提供 **HelloFunc()** 函数，供其他程序使用。该函数将在终端输出 **Hello World** 字符串。
2. 安装头文件与共享库。

三、准备工作

1. 建立 `episode5` 目录，用于存放本节涉及到的工程文件。
2. 在 `episode5` 目录下创建 `CMakeLists.txt`，内容如下：

```
PROJECT(HELLOLIB)
ADD_SUBDIRECTORY(lib)
```

四、构建共享库

1. 在 `episode5/lib` 目录下创建两个源文件 `hello.cc` 与 `hello.hpp`。
    - **hello.cc** 文件内容：

```
#include "hello.hpp"
void HelloFunc() {
    std::cout << "Hello World\n";
}
```

- **hello.hpp** 文件内容：

```
#ifndef __HELLO_HPP
#define __HELLO_HPP
#include <iostream>
void HelloFunc();
#endif
```

2. 在 `lib` 目录下创建 `CMakeLists.txt`，内容如下：

```
SET(LIBHELLO_SRC hello.cc)
ADD_LIBRARY(hello SHARED ${LIBHELLO_SRC})
```

五、编译共享库

1. 在 `episode5` 目录下创建 `build` 目录进行 **Out-of-Source** 编译。

```
mkdir build && cd build
cmake ..
make
```

2. 运行完上述命令后，生成的共享库文件 `libhello.so` 会位于 `lib` 目录中。
3. 如果需要指定生成库文件的位置，你可以在 `CMakeLists.txt` 中添加以下内容来设定输出路径：

```
SET(LIBRARY_OUTPUT_PATH ${PROJECT_BINARY_DIR}/lib)
```

然后，重新编译即可将共享库输出到指定的 `lib` 文件夹中。

六、添加静态库

1. 使用与共享库相同的源文件，添加静态库：

```
ADD_LIBRARY(hello_static STATIC ${LIBHELLO_SRC})
```

2. 再次在 `build` 目录中进行编译：

```
cd ../build
cmake ..
make
```

编译完成后， `lib` 目录下将生成一个 `libhello_static.a` 静态库。

3. 为了确保共享库和静态库使用相同的名称，我们需要通过 `SET_TARGET_PROPERTIES` 指令来设置静态库的输出文件名：

```
SET_TARGET_PROPERTIES(hello_static PROPERTIES OUTPUT_NAME "hello")
```

4. 重新编译后，将同时得到名称为 `libhello.so` 和 `libhello.a` 的库文件。

七、动态库版本号设置

1. 为了符合标准，动态库需要包含版本号。可以通过 `SET_TARGET_PROPERTIES` 指令设置动态库的版本号：

```
SET_TARGET_PROPERTIES(hello PROPERTIES VERSION 1.2 SOVERSION 1)
```

2. 重新编译后，将生成带有版本号的库文件，例如 `libhello.so.1.2` 。

八、安装共享库和头文件

1. 安装 `libhello.so` 、 `libhello_static.a` 以及 `hello.hpp` 到系统目录中。将头文件安装到 `<prefix>/include/hello` ，库文件安装到 `<prefix>/lib` ：

```
INSTALL(TARGETS hello hello_static DESTINATION <prefix>/lib)
INSTALL(FILES hello.hpp DESTINATION <prefix>/include/hello)
```

2. 最终安装的文件可以通过以下命令进行：

```
make install
```

九、总结

通过本节的学习，我们掌握了如何通过 CMake 构建静态库和共享库，并且学会了如何安装这些库和头文件到系统目录中。这为你将这些库集成到其他项目中奠定了基础，同时也让你能够通过设置库的版本号，更好地管理项目中的依赖库。

十、作业

1. 完成共享库和静态库的构建，并将库文件和头文件安装到系统目录。
2. 复习 `ADD_LIBRARY` 和 `SET_TARGET_PROPERTIES` 的使用，尝试为库设置不同的版本号。
3. 探索其他 CMake 指令，例如 `TARGET_LINK_LIBRARIES`，并尝试在项目中应用。