CMake 教程第四集：更好一点的 Hello World

一、前言

在之前的课程中，我们已经学会了如何使用 CMake 构建简单的项目。本节课，我们将让 Hello World 项目更加完善，通过引入项目目录结构、使用外部构建（Out-of-Source Build）以及构建过程中的文件管理，打造一个更像真正工程的 Hello World。

二、项目改进

1. 外部构建介绍

在接下来的所有构建中，我们将采用 **Out-of-Source 外部构建**，即构建文件会被放置到源代码目录之外的 `build` 目录中，以保持源代码目录的整洁。

2. 项目目录结构调整

为了让项目更规范化，我们需要对项目进行如下改动：

1. 添加一个 `src` 子目录，用于存放项目源代码。
2. 添加一个 `doc` 子目录，用于存放项目文档，如 `hello.txt` 。
3. 添加一个 `runhello.sh` 脚本文件，用于执行生成的 `hello` 二进制文件。
4. 将构建后的目标文件放入构建目录中的 `bin` 子目录。
5. 最终安装这些文件：将 `hello` 二进制文件和 `runhello.sh` 安装到 `/usr/local/bin`

3. 项目初始化

步骤1：准备工作

- 准备 `main.c` 和 `CMakeLists.txt` 拷贝到目录中。

步骤2：添加子目录 `src`

在项目目录下执行以下命令，创建 `src` 子目录并将 `main.c` 移动到该目录中：

```
mkdir src && cd src
touch main.cc
```

步骤3：创建 `runhello.sh` 脚本

在**项目根目录下**创建 `runhello.sh` 文件，内容如下：

```
#!/bin/bash
# 运行生成的 hello 二进制文件
./bin/hello
```

确保该脚本具有可执行权限：

```
chmod +x runhello.sh
```

步骤4：修改工程结构

目前项目结构如下：

- 一个 `src` 子目录
- 一个顶层的 `CMakeLists.txt`
- 一个 `runhello.sh` 脚本文件

我们需要为 `src` 子目录添加 `CMakeLists.txt`，内容如下：

```
# ./src/CMakeLists.txt
ADD_EXECUTABLE(hello main.c)
```

然后修改项目根目录下的 `CMakeLists.txt`，使其支持子目录构建，并安装生成的二进制文件及脚本文件：

```
# ./CMakeLists.txt
PROJECT(HELLO)

# 添加子目录
ADD_SUBDIRECTORY(src bin)

# 安装二进制文件和脚本
INSTALL(TARGETS hello DESTINATION /usr/local/bin)
INSTALL(PROGRAMS runhello.sh DESTINATION /usr/local/bin)
```

这样，`src` 子目录会被加入到项目中，并且构建结果会被存放到 `bin` 目录。生成的 `hello` 二进制文件和 `runhello.sh` 将会安装到 `/usr/bin`。

三、外部构建

现在，我们可以进行外部构建了。首先，在项目根目录下创建 `build` 目录，并在该目录中执行构建命令：

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

构建完成后，生成的目标文件 `hello` 位于 `build/bin` 目录中。

四、命令语法解释

`ADD_SUBDIRECTORY` 指令

```
ADD_SUBDIRECTORY(source_dir [binary_dir] [EXCLUDE_FROM_ALL])
```

- **source_dir**: 指定子目录存放源文件的位置。
- **binary_dir**: 可选，指定编译生成文件（包括中间结果和最终二进制文件）的存放位置。

- **EXCLUDE_FROM_ALL**: 可选，表示从编译过程中排除该子目录，通常用于不需要立即编译的示例文件。

在我们的项目中，`src` 是存放源文件的目录，`bin` 是存放编译结果的目录。如果没有指定 `bin` 目录，编译结果会默认存放在 `build/src` 目录中。

`SET` 指令

为了指定可执行文件和库文件的最终输出路径，我们可以使用 `SET` 指令：

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

这些指令将可执行文件的输出路径定义为 `build/bin`，库文件的输出路径定义为 `build/lib`。

五、改进后的Hello World项目

在本节的示例中，我们将Hello World项目进行了目录调整，并使用外部构建保持源代码目录的清洁。通过 `ADD_SUBDIRECTORY` 和 `SET` 指令，我们还可以控制构建文件的输出位置，使项目更加规范化。此外，通过创建 `runhello.sh` 脚本文件，我们还可以自动化执行构建后的二进制文件，并设置安装路径，将二进制文件和脚本文件放入系统目录。

六、作业

1. 将你自己的项目按照本节内容进行改造：
    - 添加 `src` 子目录并调整项目结构。
    - 使用 `ADD_SUBDIRECTORY` 和 `SET` 指令控制编译结果的输出位置。
    - 进行外部构建并查看生成的结果。
    - 创建 `runhello.sh` 脚本并确保其能够正常执行生成的二进制文件。
2. 尝试通过 CMake 安装构建的二进制文件和文档到系统的 `/usr/bin` 和 `/usr/share/doc` 目录。

在下节课中，我们将继续讨论如何构建和管理库文件，包括静态库和共享库的创建与使用。