chapter1.3 cmake包含静态库

1. 学习目标：学习自己创建库的操作

2. 文件构成：

```
unix
├── CMakeLists.txt
├── include
│   └── static
│       └── Hello.h
└── src
    ├── Hello.cpp
    └── main.cpp

|--- build (empty at first)
```

3. 文件填充：

3.1 Hello.h

```
/*声明了Hello类，Hello的方法是print(),*/

#ifndef __HELLO_H__
#define __HELLO_H__

class Hello
{
public:
    void print();
};

#endif
```

3.2 Hello.cpp

```
/*实现了Hello::print()*/
#include <iostream>

#include "static/Hello.h"

void Hello::print()
{
    std::cout << "Hello Static Library!" << std::endl;
}
```

3.3 main.cpp

```
#include "static/Hello.h"

int main(int argc, char *argv[])
{
    Hello hi;
    hi.print();
```

```
    return 0;
}
```

3.4 CMakeLists.txt

```
cmake_minimum_required(VERSION 3.5)
project(hello_library)
#############################################################
# (1) Create a library
#############################################################

# (1.1) 库的源文件Hello.cpp [生成静态库] hello_library
add_library(hello_library STATIC
    src/Hello.cpp
)

# (1.2) target_include_directories为一个目标（可能是一个库library也可能是可执行文件）[添加头文件路径]
target_include_directories(hello_library
    PUBLIC
        ${PROJECT_SOURCE_DIR}/include
)

#############################################################
# (2) Create an executable
#############################################################

# Add an executable with the above sources
# (2.1) 指定用哪个源文件生成可执行文件
add_executable(hello_binary
    src/main.cpp
)

# (2.2) 链接可执行文件和静态库
target_link_libraries( hello_binary
    PRIVATE
        hello_library
)
#链接库和包含头文件都有关于scope这三个关键字的用法。
```

> 总体解析：
> 这一节的主要目的是：将/src/Hello.cpp制作成库函数，main.cpp作为主程序调用该库函数
> [1] 将Hello.cpp制作成库hello_library
> [2] 为hello_library这一库函数添加调用路径
> [3] 将main.cpp制作成可执行文件hello_binary
> [4] 链接 可执行文件hello_binary 和 库函数hello_library

4. CMakeList解析：

4.1 创建静态库（static）

add_library（）函数用于从某些源文件创建一个库，**默认生成在构建文件夹（build）**。 写法如下：

```
add_library(hello_library STATIC
    src/Hello.cpp
)
```

在add_library调用中包含了源文件，用于创建名称为libhello_library.a的静态库。

> 源文件：Hello.cpp
> 构建static库：hello_library

4.2 添加头文件所在的目录

使用target_include_directories（）添加了一个目录，这个目录是库所包含的头文件的目录，并设置库属性为PUBLIC。

```
target_include_directories(hello_library
    PUBLIC
        ${PROJECT_SOURCE_DIR}/include
)
```

使用这个函数后，这个目录会在以下情况被调用：

- 编译这个库的时候 因为这个库hello_library由Hello.cpp生成，Hello.cpp中函数的定义在Hello.h中，Hello.h在这个include目录下，所以显然编译这个库的时候，这个目录会用到
- 编译链接到这个库hello_library的任何其他目标（库或者可执行文件）

4.3 private/public/interface详解

略，仅介绍要点！

- **对于公共的头文件，最好在include文件夹下建立子目录。**
- **传递给函数target_include_directories()的目录，应该是所有包含目录的根目录，然后在这个根目录下建立不同的文件夹，分别写头文件。**
- **这样使用的时候，不需要写${PROJECT_SOURCE_DIR}/include，而是直接选择对应的文件夹里对应头文件。下面是例子：`#include "static/Hello.h"` 而不是 `#include "Hello.h"` 使用此方法意味着在项目中使用多个库时，头文件名冲突的可能性较小**

4.4 链接库

创建将使用这个库的可执行文件时，必须告知编译器需要用到这个库。 可以使用target_link_library（）函数完成此操作。add_executable()连接源文件，target_link_libraries()连接库文件。

```
add_executable(hello_binary
    src/main.cpp
)

target_link_libraries( hello_binary
    PRIVATE
        hello_library
)
```

这告诉CMake在链接期间将hello_library链接到hello_binary可执行文件。 同时，这个被链接的库如果有INTERFACE或者PUBLIC属性的包含目录，那么，这个包含目录也会被传递（ propagate ）给这个可执行文件

5. 总览：

```
huluobo@huluobodeMacBook-Pro ▶ ~/cmake-examples/myCmake/chapter1.3/build ▶ ᛘ main ± ▶ cmake ..
-- The C compiler identification is AppleClang 15.0.0.15000040
-- The CXX compiler identification is AppleClang 15.0.0.15000040
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /Library/Developer/CommandLineTools/usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /Library/Developer/CommandLineTools/usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done (0.8s)
-- Generating done (0.0s)
-- Build files have been written to: /Users/huluobo/cmake-examples/myCmake/chapter1.3/build
```

```
 huluobo@huluobodeMacBook-Pro ▶ ~/cmake-examples/myCmake/chapter1.3/build ▶ ᚴ main ± ▶ make
[ 25%] Building CXX object CMakeFiles/hello_library.dir/src/Hello.cpp.o
[ 50%] Linking CXX static library libhello_library.a
[ 50%] Built target hello_library
[ 75%] Building CXX object CMakeFiles/hello_binary.dir/src/main.cpp.o
[100%] Linking CXX executable hello_binary
[100%] Built target hello_binary
 huluobo@huluobodeMacBook-Pro ▶ ~/cmake-examples/myCmake/chapter1.3/build ▶ ᚴ main ± ▶ ls
CMakeCache.txt      CMakeFiles          Makefile            cmake_install.cmake hello_binary
libhello_library.a
 huluobo@huluobodeMacBook-Pro ▶ ~/cmake-examples/myCmake/chapter1.3/build ▶ ᚴ main ± ▶ ./hello_library
zsh: no such file or directory: ./hello_library
 ✘ huluobo@huluobodeMacBook-Pro ▶ ~/cmake-examples/myCmake/chapter1.3/build ▶ ᚴ main ± ▶ ./hello_binary
Hello Static Library!
 huluobo@huluobodeMacBook-Pro ▶ ~/cmake-examples/myCmake/chapter1.3/build ▶ ᚴ main ± ▶
```