chapter1.4 cmake包含动态库

1. 学习目标：自己创建动态库

2. 文件构成：

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

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 "shared/Hello.h"

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

3.3 main.cpp

```
#include "shared/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) 根据Hello.cpp生成动态库
add_library(hello_library SHARED
    src/Hello.cpp
)

# (1.2) 给动态库hello_library起一个别的名字hello::library [其实这一步本不必要，这里只是为了炫技hhh]
add_library(hello::library ALIAS hello_library)

# (1.3) 为这个库目标，添加头文件路径，PUBLIC表示包含了这个库的目标也会包含这个路径
target_include_directories(hello_library
    PUBLIC
        ${PROJECT_SOURCE_DIR}/include
)

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

# (2.1) 根据main.cpp生成可执行文件
add_executable(hello_binary
    src/main.cpp
)

# (2.2) 链接库和可执行文件，使用的是这个库的别名。PRIVATE 表示
target_link_libraries( hello_binary
    PRIVATE
        hello::library
)
```

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(改名为hello::library)
> [2] 为hello_library(即：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. 文件解析：

4.1 总体逻辑跟chapter1.3一样，本文从略

4.2 new：使用Alias别名，下面介绍：

顾名思义，别名目标是在只读上下文中可以代替真实目标名称的替代名称。

```
add_library(hello::library ALIAS hello_library)
```

如下所示，当您将目标链接到其他目标时，使用别名可以引用目标。

链接共享库与链接静态库相同。 创建可执行文件时，请使用target_link_library（）函数指向您的库 。

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

target_link_libraries(hello_binary
    PRIVATE
        hello::library
)
```

这告诉CMake使用别名目标名称将hello_library（hello::library）链接到hello_binary可执行文件

5. 总览：

```
huluobo@huluobodeMacBook-Pro ▶ ~/cmake-examples/myCmake/chapter1.4/build ▶ ᛘ main ± ▶ cmake ..
-- Configuring done (0.0s)
-- Generating done (0.0s)
-- Build files have been written to: /Users/huluobo/cmake-examples/myCmake/chapter1.4/build
 huluobo@huluobodeMacBook-Pro ▶ ~/cmake-examples/myCmake/chapter1.4/build ▶ ᛘ main ± ▶ make
[ 25%] Building CXX object CMakeFiles/hello_library.dir/src/Hello.cpp.o
[ 50%] Linking CXX shared library libhello_library.dylib
[ 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/chapter1.4/build ▶ ᛘ main ± ▶ ls
CMakeCache.txt         CMakeFiles             Makefile               cmake_install.cmake    hello_binary
libhello_library.dylib
 huluobo@huluobodeMacBook-Pro ▶ ~/cmake-examples/myCmake/chapter1.4/build ▶ ᛘ main ± ▶ ./hello_binary
Hello Shared Library!
 huluobo@huluobodeMacBook-Pro ▶ ~/cmake-examples/myCmake/chapter1.4/build ▶ ᛘ main ± ▶
```