chapter1.11 cmake中设置c++标准

这一节相当于拓展资料，看看就行，实际意义不大

以下示例显示了设置C ++标准的不同方法，以及提供哪些版本的CMake。
The examples include:

- common-method. 可以与大多数版本的CMake一起使用的简单方法。
- cxx-standard. 使用CMake v3.1中引入的CMAKE_CXX_STANDARD变量。
- compile-features. 使用CMake v3.1中引入的target_compile_features函数。

C++ Standard Common Method

Introduction

This example shows a common method to set the C Standard. This can be used with most versions of CMake. However, if you are targeting a recent version of CMake there are more convenient methods available.此示例显示了设置C 标准的常用方法。 可以与大多数版本的CMake一起使用。 但是，如果有CMake的最新版本建议使用其他更便捷的方法。

The files in this tutorial are below:

```
A-hello-cmake$ tree
.
├── CMakeLists.txt
├── main.cpp
```

- CMakeLists.txt - Contains the CMake commands you wish to run
- main.cpp - A simple "Hello World" cpp file targeting C++11.

Concepts

Checking Compile flags

CMake has support for attempting to compile a program with any flags you pass into the function `CMAKE_CXX_COMPILER_FLAG`. The result is then stored in a variable that you pass in.CMake支持传递一个变量给函数CMAKE_CXX_COMPILER_FLAG来编译程序。 然后将结果存储在您传递的变量中。

For example:

```
include(CheckCXXCompilerFlag)
CHECK_CXX_COMPILER_FLAG("-std=c++11" COMPILER_SUPPORTS_CXX11)
```

This example will attempt to compile a program with the flag `-std=c11` and store the result in the variable `COMPILER_SUPPORTS_CXX11`.这个例子将flag “-std=c11”传递给变量COMPILER_SUPPORTS_CXX11

The line `include(CheckCXXCompilerFlag)` tells CMake to include this function to make it available for use. 想使用这个函数，必须使用include(CheckCXXCompilerFlag)包含这个函数

Adding the flag

Once you have determined if the compile supports a flag, you can then use the standard cmake methods to add this flag to a target. In this example we use the `CMAKE_CXX_FLAGS` to propegate the flag to all targets .确定编译器是否支持标志后，即可使用标准cmake方法将此标志添加到目标。 在此示例中，我们使用CMAKE_CXX_FLAGS将标志（c++标准）传播给所有目标。

```
if(COMPILER_SUPPORTS_CXX11)#
    set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++11")
elseif(COMPILER_SUPPORTS_CXX0X)#
    set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++0x")
else()
    message(STATUS "The compiler ${CMAKE_CXX_COMPILER} has no C++11 support. Please use a different C++
compiler.")
endif()
```

The above example only checks for the gcc version of the compile flags and supports fallback from C++11 to the pre-standardisation C+\0x flag. In real usage you may want to check for C14, or add support for different methods of setting the compile, e.g. `-std=gnu11` 上面的示例仅检查编译标志的gcc版本，并支持从C + 11到预标准化C + \ + 0x标志的回退。 在实际使用中，您可能需要检查C14，或添加对设置编译方法的不同支持，例如 `-std = gnu11`