cmake学习资料汇总

https://cgold.readthedocs.io/en/latest/overview.html

Learning CMake: A beginner's guide

tuannguyen68

Published with GitBook

Table of Contents

Introduction

Recently, I've created a simple library in C++. I want to use CMake as the building system so it’s the time to learn a new tool.

Summary

Chapter 1: Let's start with CMake

1. What is CMake?

From the CMake's website:

> CMake is an extensible, open-source system that manages the build process in an operating system and in a compiler-independent manner. Unlike many cross-platform systems, CMake is designed to be used in conjunction with the native build environment. Simple configuration files placed in each source directory (called CMakeLists.txt files) are used to generate standard build files (e.g., makefiles on Unix and projects/workspaces in Windows MSVC) which are used in the usual way. CMake can generate a native build environment that will compile source code, create libraries, generate wrappers and build executables in arbitrary combinations. CMake supports in-place and out-of-place builds, and can therefore support multiple builds from a single source tree. CMake also supports static and dynamic library builds. Another nice feature of CMake is that it generates a cache file that is designed to be used with a graphical editor. For example, when CMake runs, it locates include files, libraries, and executables, and may encounter optional build directives. This information is gathered into the cache, which may be changed by the user prior to the generation of the native build files.

In a short conclusion, CMake help you to manage and build your source codes effectively. If you have some troubles with gcc and Makefile, just move out to CMake.

2. Installation

To install CMake in Linux, just simply do on your terminal

```
# For Ubuntu

$ sudo apt-get install cmake

# For Redhat

$ yum install cmake

# For Mac OS X with Macports

$ sudo port install cmake
```

3. Quick start

So I assume that you know C++ and what the Makefile is. CMake will do the job of Makefile from now. Let start with a simple C++ program.

```
// test.cpp

#include <iostream>

using namespace std;

int main(void) {

     cout << "Hello World" << endl;

     return(0);

}
```

And you saved it as `test.cpp`, then to compile it in CMake you should create a txt file named `CMakeLists.txt`

```
# Specify the minimum version for CMake

cmake_minimum_required(VERSION 2.8)

# Project's name

project(hello)
# Set the output folder where your program will be created
set(CMAKE_BINARY_DIR ${CMAKE_SOURCE_DIR}/bin)
set(EXECUTABLE_OUTPUT_PATH ${CMAKE_BINARY_DIR})
set(LIBRARY_OUTPUT_PATH ${CMAKE_BINARY_DIR})

# The following folder will be included
include_directories("${PROJECT_SOURCE_DIR}")
```

There is a list of CMake's global varibales. You should know them.

> CMAKE_BINARY_DIR

if you are building in-source, this is the same as `CMAKE_SOURCE_DIR`, otherwise this is the top level directory of your build tree

这个难道在整个编译的流程中永远保持不变吗

> CMAKE_SOURCE_DIR

this is the directory, from which cmake was started, i.e. the top level source directory

> EXECUTABLE_OUTPUT_PATH

set this variable to specify a common place where CMake should put all executable files (instead of `CMAKE_CURRENT_BINARY_DIR`)

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

> LIBRARY_OUTPUT_PATH

set this variable to specify a common place where CMake should put all libraries (instead of `CMAKE_CURRENT_BINARY_DIR`)

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

> PROJECT_NAME

the name of the project set by `PROJECT()` command.

> PROJECT_SOURCE_DIR

所以PROJECT目录的作用应该就是确定PROJECT_SOURCE_DIR 目录

包含了PROJECT命令的CMakeLists.txt文件所在的目录是什么。。。。。？

所以应该是PROJECT_SOURCE_DIR 目录

contains the full path to the root of your project source directory, i.e. to the nearest directory where `CMakeLists.txt` contains the PROJECT() command Now, you have to compile the `test.cpp`. The way to do this task is too simple. Add the following line into your `CMakeLists.txt`:

```
add_executable(hello ${PROJECT_SOURCE_DIR}/test.cpp)
```

Now, let build the source code with CMake. At this point, you will have the folder with the following files:

```
$ ls
test.cpp   CMakeLists.txt
```

To build your project `hello`, just do

```
$ cmake -H. -Bbuild
$ cmake --build build -- -j3
```

The first command will creates CMake configuration files inside folder `build` and the second one will generate the output program hello in `bin` folder. You should test your output:

```
$ ./bin/hello
Hello World
```

Too simple, right?

> The source code of this sample project canbe found at CMakeLists.txt and test.cpp

Chapter 2: CMake's build system