CMake简易教程

更新日期 : 2015-12-04

CMake简易教程

使用的原因

- CMake 拥有比直接编写 Makefile 更直观的语法，更易于编写和维护。
- 伟大的 jetbrains 公司的CLion编译器依赖于 CMake 。

简介

CMake 根据内置的规则和语法来自动生成相关的makefile 文件进行编译，同时还支持静态库和动态库的构建。 具体 CMake 的介绍和详细语法可以参考官方文档（http://www.cmake.org/）。这里简单对项目中常用的内容 进行一些说明。

工程中的 CMakeLists.txt

通过 CMake 管理的项目主目录下会有一个 CMakeLists.txt，其中会包括工程包含哪些子目录等内容。 而在每个子目录下，也会包含一个 CMakeLists.txt， 用来管理该子目录中相关内容的构建。

简单的 CMakeLists.txt

下面给出一个简单的 CMakeLists.txt：

```
cmake_minimum_required(VERSION 3.1)  # CMake 版本要求
PROJECT(hello)                       # 项目名称

aux_source_directory(. PROGRAM_SOURCE)  # 将当前目录所有文件添加到变量 PROGRAM_SOURCE 中

add_executable(hello ${PROGRAM_SOURCE}) # 指定目标可执行文件 hello 的源代码文件为 PROGRAM_SOURC
E
```

要编译的时候，可以建立单独的文件夹，让编译过程文件和源代码区分出来，以下是一种编译的方式, 当前目录为项目源代码目录：

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

这样操作之后，编译的目标会位于 build 中，不会和源代码混在一起。

添加外部头文件查找目录

当我们用到外部的库的时候，我们便需要添加外部库的头文件所在目录作为头文件查找目录。在 CMakeLists 中添加以下代码即可：

```
include_directories(${CMAKE_CURRENT_SOURCE_DIR}/../
    ${CMAKE_CURRENT_SOURCE_DIR}../include)
```

CMake 中通过空格或者换行区分多个变量，上面的示例便是添加了两个目录到头文件查找路径中。

添加外部链接库

通过以下代码可以添加外部链接库查找目录, 其中 CMAKE_CURRENT_SOURCE_DIR 是内置宏， 表示当前 CMakeLists.txt 所在目录：

```
link_directories(
    ${CMAKE_CURRENT_SOURCE_DIR}/../../lib
    ${CMAKE_CURRENT_SOURCE_DIR}/../lib)
```

通过以下代码可以添加链接的外部库，这里链接 libmylib1 和 libmylib2， 这里链接的库可以是静态库也可以是动态库：

```
link_libraries(mylib1
    mylib2)
```

如果是链接指定目录指定某个库，则可以用以下方式：

```
target_link_libraries(hello ../mylib1.a
    hello ../mylib2.so)
```

对于同一个工程中构建的库，则可以用以下方式，无需指定具体的库的位置：

```
target_link_libraries(hello mylib1 mylib2)
```

自定义编译选项

编译选项的内置宏为 CMAKE_CXX_FLAGS ，只要将此宏设置为自定义的编译选项即可：

```
set(CMAKE_CXX_FLAGS “-std=c++11 -O2 -g”)
```

创建工程内的库

创建库和创建可执行文件的 CMakeLists.txt 区别不大，只是讲 add_executable 替换为：

```
add_library(mylib STATIC ${SRC})
```

STATIC表示创建静态库，目标文件为 libmylib.a，如果是 SHARED， 则为创建动态库.

CMake 模块

如果工程的很多 CMakeLists.txt 都有共同的部分，则可以通过 CMake 模块来构建更容易维护的 CMakeLists.txt 代码。 CMake模块为以 .cmake 为文件后缀的文件，其中的语法与 CMakeLists.txt 保持一致，可以直接 讲 CMakeLists.txt 中的公共内容抽取出来，放到 .cmake 文件中。然后通过以下方式包含进来， 比如说公共模块的文件名为 common.cmake ：

```
set(CMAKE_MODULE_PATH "${CMAKE_CURRENT_SOURCE_DIR}/../cmake") #设置模块放置的目录，此处为上层目录的 cmake 文件夹中
include(common) # 包含 common 模块，无需文件后缀名
```

执行外部命令

当我们的代码在编译之前需要先执行一些外部命令，比如说使用 thrift 接口需要先执行 thrift 的代码生成器生成代码时，我们可以在 CMakeLists.txt 中添加以下代码， 当执行 cmake 命令 生成 Makefile 的时候，该命令就会被执行：

```
set(THRIFT_FILE ${CMAKE_CURRENT_SOURCE_DIR}/mythrift.thrift)
exec_program("thrift --gen cpp -o ${CMAKE_CURRENT_SOURCE_DIR} ${THRIFT_FILE}")
```

总结

以上介绍了 CMake 在一般工程中所需的基本功能，更详细的需要请查阅官方文档。希望此文对大家有所帮助。