

CMake使用总结

总结CMake的常用命令，并介绍有用的CMake资源。

CMake意为cross-platform make，可用于管理c/c++工程。CMake解析配置文件CMakeLists.txt生成Makefile，相比直接用Makefile管理工程，CMake更灵活和简单。

简单的例子

一个完整的Demo可参考这里 (<https://github.com/mawenbao/protobuf-demo>)。

假设当前目录的结构为

```
./a.cpp
./b.cpp

./include/common.h
./include/defines.h

./other/c.cpp
./other/d.cpp

./lib/libB.a
./lib/libBd.a
./lib/libA.so
./lib/libAd.so
./lib/libB.so
./lib/libBd.so
./lib/libC.so
./lib/libCd.so
```

使用下面的CMakeLists.txt文件，目标是编译当前目录和./other目录下的所有源文件，并链接./lib目录下的相应库文件到最终的可执行文件./bin/hello(或./bin/hellod)。

```
cmake_minimum_required(VERSION 2.8)
project(helloworld)

set(CMAKE_VERBOSE_MAKEFILE on)
set(CMAKE_CXX_COMPILER "g++")
set(CMAKE_CXX_FLAGS "-Wall")
set(CMAKE_CXX_FLAGS_DEBUG "-g3")
set(CMAKE_CXX_FLAGS_RELEASE "-O2")
set(EXECUTABLE_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/bin)

aux_source_directory(. SRC_LIST)
aux_source_directory(.other OTHER_SRC_LIST)
list(APPEND SRC_LIST ${OTHER_SRC_LIST})

include_directories(${PROJECT_SOURCE_DIR}/include)
link_directories(${PROJECT_SOURCE_DIR}/lib)

if(${CMAKE_BUILD_TYPE} MATCHES "debug")
    add_executable(hellod ${SRC_LIST})
    target_link_libraries(hellod Ad Bd.a Cd.so)
else()
    add_executable(hello ${SRC_LIST})
    target_link_libraries(hello A B.a C.so)
endif()
```

执行命令 `cmake -DCMAKE_BUILD_TYPE=debug .` 生成Makefile，`make`之后生成`./bin/hellod`（调试版本），或执行 `cmake .` 最后生成`./bin/hello`。

常用的CMake变量

详细内容请参考CMake Useful Variables (http://www.cmake.org/Wiki/CMake_Useful_Variables)。

- PROJECT_SOURCE_DIR 工程的源文件目录，通常是包含CMakeLists.txt（有Project命令的）的目录。

自定义变量

可在命令行下向CMake传递自定义变量

```
cmake -DMY_VAR=hello .
```

然后在CMakeLists.txt中使用，注意大小写。

```
message(${MY_VAR})
```

构建类型

cmake默认支持多种构建类型(build type)，每种构建类型都有专门的编译参数变量，详情见下表^[1]:

CMAKE_BUILD_TYPE	对应的c编译选项变量	对应的c++编译选项变量
None	CMAKE_C_FLAGS	CMAKE_CXX_FLAGS
Debug	CMAKE_C_FLAGS_DEBUG	CMAKE_CXX_FLAGS_DEBUG
Release	CMAKE_C_FLAGS_RELEASE	CMAKE_CXX_FLAGS_RELEASE

RelWithDebInfo	CMAKE_C_FLAGS_RELWITHDEBINFO	CMAKE_CXX_FLAGS_RELWITHDEBINFO
MinSizeRel	CMAKE_C_FLAGS_MINSIZEREL	CMAKE_CXX_FLAGS_MINSIZEREL

在CMakeLists.txt中可以自定义编译选项变量

```
set(CMAKE_CXX_FLAGS_RELEASE "-O2")
```

然后运行cmake的时候，传入相应的构建类型即可

```
cmake -DCMAKE_BUILD_TYPE=Release
```

常用命令

详情可参考对应版本的CMake文档 (<http://www.cmake.org/cmake/help/documentation.html>)。

检查CMake版本

cmake版本至少为2.8

```
cmake_minimum_required(VERSION 2.8)
```

定义工程名称

工程名为helloworld

```
project(helloworld)
```

查找源文件

查找当前目录下所有的源文件并保存到SRC_LIST变量里

```
aux_source_directory(. SRC_LIST)
```

查找src目录下所有以cmake开头的文件并保存到CMAKE_FILES变量里

```
file(GLOB CMAKE_FILES "src/cmake*")
```

file 命令同时支持目录递归查找

```
file(GLOB_RECURSE CMAKE_FILES "src/cmake*")
```

按照官方文档的说法，不建议使用file的GLOB指令来收集工程的源文件，原文解释如下

We do not recommend using GLOB to collect a list of source files from your source tree. If no CMakeLists.txt file changes when a source is added or removed then the generated build system cannot know when to ask CMake to regenerate.

大意就是，GLOB收集的源文件增加或删除，而CMakeLists.txt没有发生修改时，CMake不能识别这些文件。其实，当CMakeLists.txt使用aux_source_directory和file glob查找工程源文件时，如果添加或删除源文件，都需要重新运行CMake。

set命令

经常配合set命令使用的CMake变量，使用 `set(variable value)` 进行设置。

- CMAKE_VERBOSE_MAKEFILE on 输出详细的编译和链接信息
- CMAKE_CXX_COMPILER "g++" c++编译器
- CMAKE_CXX_FLAGS "-Wall" c++编译器参数
 - CMAKE_CXX_FLAGS_DEBUG 除 CMAKE_CXX_FLAGS 外，debug版本的额外编译器参数
 - CMAKE_CXX_FLAGS_RELEASE 除 CMAKE_CXX_FLAGS 外，release版本的额外编译器参数
- EXECUTABLE_OUTPUT_PATH \${PROJECT_SOURCE_DIR}/bin 可执行文件的输出目录
- LIBRARY_OUTPUT_PATH \${PROJECT_SOURCE_DIR}/lib 链接库的输出目录

set命令还可以设置自定义变量，比如

```
set(MY_GREETINGS "hello world")
```

包含目录和链接目录

将 ./include 和 ./abc 加入包含目录列表

```
include_directories(  
    ./include  
    ./abc  
)
```

将 ./lib 加入编译器链接阶段的搜索目录列表

```
link_directories(  
    ./lib  
)
```

添加生成目标

使用SRC_LIST源文件列表里的文件生成一个可执行文件hello

```
add_executable(hello ${SRC_LIST})
```

使用SRC_LIST源文件列表里的文件生成一个静态链接库libhello.a

```
add_library(hello STATIC ${SRC_LIST})
```

使用SRC_LIST源文件列表里的文件生成一个动态链接库libhello.so

```
add_library(hello SHARED ${SRC_LIST})
```

将若干库文件链接到生成的目标hello(libhello.a或libhello.so)

```
target_link_libraries(hello A B.a C.so)
```

需要注意的是，`target_link_libraries`里库文件的顺序符合gcc链接顺序的规则，即被依赖的库放在依赖它的库的后面，比如上面的命令里，`libA.so`可能依赖于`libB.a`和`libC.so`，如果顺序有错，链接时会报错。还有一点，`B.a`会告诉CMake优先使用静态链接库`libB.a`，`C.so`会告诉CMake优先使用动态链接库`libC.so`，也可直接使用库文件的相对路径或绝对路径。使用绝对路径的好处在于，当依赖的库被更新时，`make`的时候也会重新链接。

自定义链接选项

有时候需要自定义链接选项，比如需要单独对`B.a`使用 `--whole-archive` 选项，可以

```
target_link_libraries(hello A -Wl,--whole-archive B.a -Wl,--no-whole-archive C.so)
```

自定义Makefile目标

运行下面的`whatever`目标 `make whatever`，会先创建一个目录 `./hello`，然后将当前目录的 `a.txt` 拷贝到新建的 `./hello` 目录里。

```
add_custom_command(  
    OUTPUT ./hello/a.txt  
    COMMAND mkdir -p ./hello  
            cp a.txt ./hello  
    DEPENDS a.txt  
)  
add_custom_target(whatever DEPENDS ./hello/a.txt)
```

自定义目标还可以使用 `add_dependencies` 命令加入到其他目标的依赖列表里，当执行 `make demo` 时，`whatever` 目标会被自动调用。

```
add_executable(demo ${SRC_LIST})  
add_dependencies(demo whatever)
```

其他常用命令

包含其他目录的CMakeLists.txt

```
include(/path/to/another/CMakeLists.txt)
```

if命令

```
if(${CMAKE_BUILD_TYPE} MATCHES "debug")  
    ...  
else()  
    ...  
endif()
```

list命令

```
list(APPEND SRC_LIST
  a.cpp
  b.cpp
)

list(REMOVE_ITEM SRC_LIST
  a.cpp
)
```

更多的例子

自定义Makefile目标的完整例子

下面的CMakeLists.txt添加一个自定义目标proto，该目标在编译工程前，会先调用protobuf程序编译先生成Google Protocol Buffers的消息解析器。

```
cmake_minimum_required(VERSION 2.6)
project(protobuf-demo)

# compile proto files
set(PROTO_IN news.proto)
set(PROTO_SRC news.pb.cc)
set(PROTO_OUT news.pb.h news.pb.cc proto/)

add_custom_command(
  OUTPUT ${PROTO_OUT}
  COMMAND protoc --cpp_out . --java_out . ${PROTO_IN}
  DEPENDS ${PROTO_IN}
)
add_custom_target(proto DEPENDS ${PROTO_OUT})

aux_source_directory(. SRC_LIST)
list(APPEND SRC_LIST
  ${PROTO_SRC}
)

set(CMAKE_CXX_COMPILER "g++")
set(CMAKE_CXX_FLAGS "-Wall")
set(CMAKE_VERBOSE_MAKEFILE on)

add_executable(demo ${SRC_LIST})
add_dependencies(demo proto)
target_link_libraries(demo protobuf)
```

阅读资料

1. CMake文档列表 (<http://www.cmake.org/cmake/help/documentation.html>)
2. CMake常用变量列表 (http://www.cmake.org/Wiki/CMake_Useful_Variables)
3. CMake入门教程 (http://www.cmake.org/cmake/help/cmake_tutorial.html)

脚注

1. ^ CMake Useful Variables: Compilers and Tools

(http://www.cmake.org/Wiki/CMake_Useful_Variables#Compilers_and_Tools), 引用于2014-05-06。

社交帐号登录: [微信](#) [微博](#) [QQ](#) [人人](#) [更多»](#)



说点什么吧...

发布

11 条评论

[最新](#) [最早](#) [最热](#)



[cosmo-xyh \(http://t.qq.com/xyh_physics\)](http://t.qq.com/xyh_physics)

好文啊！清晰明了！

(http://t.qq.com/xyh_physics)2014年9月5日 [回复](#) [顶](#) [转发](#)



[mawenbao \(http://www.douban.com/people/wishome/\)](http://www.douban.com/people/wishome/)

(<http://www.douban.com/people/wishome/>)

2014年9月7日 [回复](#) [顶](#) [转发](#)



[九月恒心 \(http://weibo.com/jinxueliu\)](http://weibo.com/jinxueliu)

帮助很大，谢谢啦

(<http://weibo.com/jinxueliu>)2015年7月24日 [回复](#) [顶](#) [转发](#)



[mawenbao \(http://blog.atime.me/\)](http://blog.atime.me/)

(<http://blog.atime.me/>)

2015年7月25日 [回复](#) [顶](#) [转发](#)



[itfanr \(http://weibo.com/itfanr\)](http://weibo.com/itfanr)

写的很好！

(<http://weibo.com/itfanr>)2015年7月26日 [回复](#) [顶](#) [转发](#)



[mawenbao \(http://blog.atime.me/\)](http://blog.atime.me/)

(<http://blog.atime.me/>)

2015年7月27日 [回复](#) [顶](#) [转发](#)



[维唯为为 \(http://weibo.com/rootls\)](http://weibo.com/rootls)

谢谢，总结不错。。


(<http://weibo.com/rootls>)2015年8月24日 [回复](#) [顶](#) [转发](#)




kyokyojiang

写的真好，帮助很大，谢谢啦～



2015年11月17日 [回复](#) [顶](#) [转发](#)

多说 (<http://duoshuo.com>) 发布时间:

2013-11-12 17:45

 最后修改:

2015-10-02 21:50

 分类:笔记 (<https://blog.atime.me/note/index.html>) 标签:c++ (<https://blog.atime.me/tag/c.html>)¹⁰ cmake (<https://blog.atime.me/tag/cmake.html>)¹note (<https://blog.atime.me/tag/note.html>)¹⁴ 总结 (<https://blog.atime.me/tag/zong-jie.html>)²²

Powered by Pelican (<https://github.com/getpelican/pelican>), theme (<https://github.com/mawenbao/niu-x2-sidebar>) built with Bootstrap3 (<https://github.com/twbs/bootstrap>) by Ma Wenbao (<https://blog.mawenbao.com>), icons by Font Awesome (<https://fontawesome.github.io/Font-Awesome>).

© 2009-2016 Ma Wenbao (<https://blog.atime.me>) / 关于 (</about.html>) / 使用协议 (</agreement.html>) / 沪ICP备14018579号-1 (<http://www.miitbeian.gov.cn/>)

 (/my_gnupg.html)  ([mailto: mwenbao@gmail.com](mailto:mwenbao@gmail.com))  (<https://github.com/mawenbao>)  (</feed.xml>)