**基本概念**

跨平台的安装（[编译](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91/1258343)）工具，可以用简单的语句来描述所有平台的安装(编译过程)。他能够输出各种各样的makefile或者project文件，能测试[编译器](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%99%A8/8853067)所支持的C++特性,类似UNIX下的automake。只是 CMake 的[组态档](https://baike.baidu.com/item/%E7%BB%84%E6%80%81%E6%A1%A3/4812025)取名为 CMakeLists.txt。Cmake 并不直接建构出最终的软件，而是产生标准的建构档（如 Unix 的 Makefile 或 Windows Visual C++ 的 projects/workspaces），然后再依一般的建构方式使用。

**CMake常用命令**

**基本语法**

COMMAND(args...)，多个参数用空白符分隔

**常用命令**

**1. cmake\_minimum\_required(VERSION 3.4.1)**

指定需要的最小的cmake版本

**2. aux\_source\_directory**

查找源文件并保存到相应的变量中:

#查找当前目录下所有源文件并保存至SRC\_LIST变量中

aux\_source\_directory(. SRC\_LIST)

**3. add\_library**

**3.1 添加一个库**

add\_library(<name> [STATIC | SHARED | MODULE] [EXCLUDE\_FROM\_ALL] source1 source2 ... sourceN)

* 添加一个名为<name>的库文件
* 指定STATIC, SHARED, MODULE参数来指定要创建的库的类型, STATIC对应的静态库(.a)，SHARED对应共享动态库(.so)
* [EXCLUDE\_FROM\_ALL], 如果指定了这一属性，对应的一些属性会在目标被创建时被设置(**指明此目录和子目录中所有的目标，是否应当从默认构建中排除, 子目录的IDE工程文件/Makefile将从顶级IDE工程文件/Makefile中排除**)
* source1 source2 ... sourceN用来指定源文件

**3.2 导入已有的库**

add\_library(<name> [STATIC | SHARED | MODULE | UNKNOWN] IMPORTED)

导入了一个已存在的<name>库文件，导入库一般配合set\_target\_properties使用，这个命令用来指定导入库的路径,比如：

add\_library(test SHARED IMPORTED)

set\_target\_properties( test #指定目标库名称

PROPERTIES IMPORTED\_LOCATION #指明要设置的参数

libs/src/${ANDROID\_ABI}/libtest.so #设定导入库的路径)

**4. set**

设置CMake变量

**例子：**

# 设置可执行文件的输出路径(EXCUTABLE\_OUTPUT\_PATH是全局变量)

set(EXECUTABLE\_OUTPUT\_PATH [output\_path])

# 设置库文件的输出路径(LIBRARY\_OUTPUT\_PATH是全局变量)

set(LIBRARY\_OUTPUT\_PATH [output\_path])

# 设置C++编译参数(CMAKE\_CXX\_FLAGS是全局变量)

set(CMAKE\_CXX\_FLAGS "-Wall std=c++11")

# 设置源文件集合(SOURCE\_FILES是本地变量即自定义变量)

set(SOURCE\_FILES main.cpp test.cpp ...)

**5. include\_directories**

设置头文件位置

# 可以用相对货绝对路径，也可以用自定义的变量值

include\_directories(./include ${MY\_INCLUDE})

**6. add\_executable**

添加可执行文件

add\_executable(<name> ${SRC\_LIST})

**7. target\_link\_libraries**

将若干库链接到目标库文件

target\_link\_libraries(<name> lib1 lib2 lib3)

将lib1, lib2, lib3链接到<name>上

NOTE: 链接的顺序应当符合gcc链接顺序规则，被链接的库放在依赖它的库的后面，即如果上面的命令中，lib1依赖于lib2, lib2又依赖于lib3，则在上面命令中必须严格按照lib1 lib2 lib3的顺序排列，否则会报错  
也可以自定义链接选项, 比如针对lib1使用-WL选项,target\_link\_libraries(<name> lib1 -WL, lib2 lib3)

**8. add\_definitions**

为当前路径以及子目录的源文件加入由-D引入得define flag

add\_definitions(-DFOO -DDEBUG ...)

**9. add\_subdirectory**

如果当前目录下还有子目录时可以使用add\_subdirectory，子目录中也需要包含有CMakeLists.txt

# sub\_dir指定包含CMakeLists.txt和源码文件的子目录位置

# binary\_dir是输出路径， 一般可以不指定

add\_subdirecroty(sub\_dir [binary\_dir])

**10. file**

文件操作命令

# 将message写入filename文件中,会覆盖文件原有内容

file(WRITE filename "message")

# 将message写入filename文件中，会追加在文件末尾

file(APPEND filename "message")

# 从filename文件中读取内容并存储到var变量中，如果指定了numBytes和offset，

# 则从offset处开始最多读numBytes个字节，另外如果指定了HEX参数，则内容会以十六进制形式存储在var变量中

file(READ filename var [LIMIT numBytes] [OFFSET offset] [HEX])

# 重命名文件

file(RENAME <oldname> <newname>)

# 删除文件， 等于rm命令

file(REMOVE [file1 ...])

# 递归的执行删除文件命令, 等于rm -r

file(REMOVE\_RECURSE [file1 ...])

# 根据指定的url下载文件

# timeout超时时间; 下载的状态会保存到status中; 下载日志会被保存到log; sum指定所下载文件预期的MD5值,如果指定会自动进行比对，如果不一致，则返回一个错误; SHOW\_PROGRESS，进度信息会以状态信息的形式被打印出来

file(DOWNLOAD url file [TIMEOUT timeout] [STATUS status] [LOG log] [EXPECTED\_MD5 sum] [SHOW\_PROGRESS])

# 创建目录

file(MAKE\_DIRECTORY [dir1 dir2 ...])

# 会把path转换为以unix的/开头的cmake风格路径,保存在result中

file(TO\_CMAKE\_PATH path result)

# 它会把cmake风格的路径转换为本地路径风格：windows下用"\"，而unix下用"/"

file(TO\_NATIVE\_PATH path result)

# 将会为所有匹配查询表达式的文件生成一个文件list，并将该list存储进变量variable里, 如果一个表达式指定了RELATIVE, 返回的结果将会是相对于给定路径的相对路径, 查询表达式例子: \*.cxx, \*.vt?

NOTE: 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

file(GLOB variable [RELATIVE path] [globbing expressions]...)

**11. set\_directory\_properties**

设置某个路径的一种属性

set\_directory\_properties(PROPERTIES prop1 value1 prop2 value2)

prop1 prop代表属性，取值为：

* INCLUDE\_DIRECTORIES
* LINK\_DIRECTORIES
* INCLUDE\_REGULAR\_EXPRESSION
* ADDITIONAL\_MAKE\_CLEAN\_FILES

**12. set\_property**

在给定的作用域内设置一个命名的属性

set\_property(<GLOBAL |

DIRECTORY [dir] |

TARGET [target ...] |

SOURCE [src1 ...] |

TEST [test1 ...] |

CACHE [entry1 ...]>

[APPEND]

PROPERTY <name> [value ...])

第一个参数决定了属性可以影响的作用域,必须为以下值：

* GLOBAL 全局作作用域,不接受名字
* DIRECTORY 默认为当前路径，但是同样也可以用[dir]指定路径
* TARGET 目标作用，可以是0个或多个已有的目标
* SOURCE 源作用域， 可以是0个过多个源文件
* TEST 测试作用域, 可以是0个或多个已有的测试
* CACHE 必须指定0个或多个cache中已有的条目

PROPERTY参数是必须的

**参考**

* [CMake使用总结](https://link.jianshu.com?t=https:/www.mawenbao.com/note/cmake.html)<https://www.mawenbao.com/note/cmake.html>
* [CMake语法](https://link.jianshu.com?t=https:/gearyyoung.gitbooks.io/cmake-api/content/cmake/CMake%E8%AF%AD%E6%B3%95.html) 链接见下需翻墙
* <https://link.jianshu.com/?t=https://gearyyoung.gitbooks.io/cmake-api/content/cmake/CMake%E8%AF%AD%E6%B3%95.html>