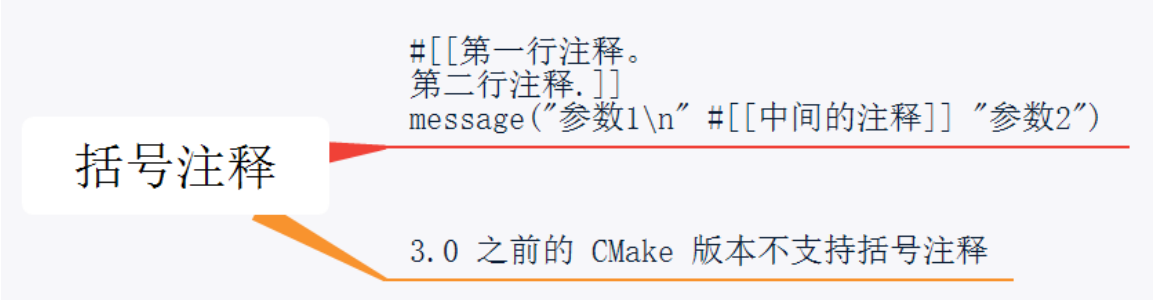
第二章 CMake常用功能

1. 2.1 cmake注释

1.1. 括号注释



1.1.1. #[[第一行注释。

第二行注释.]]

message("参数1\n" #[[中间的注释]] "参数2")

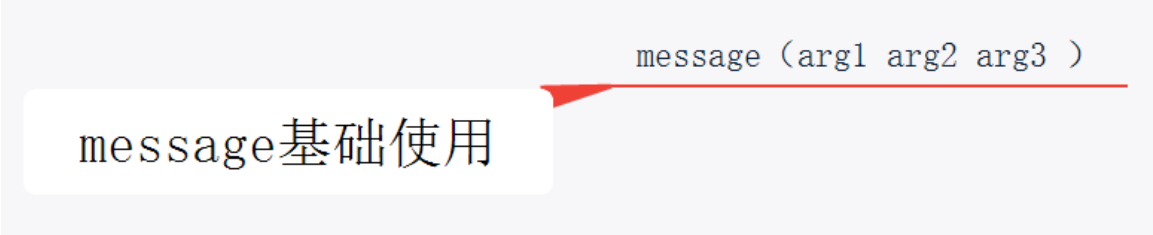
1.1.2. 3.0 之前的 CMake 版本不支持括号注释

1.2. 行注释

1.2.1. 行注释，一直运行到行尾

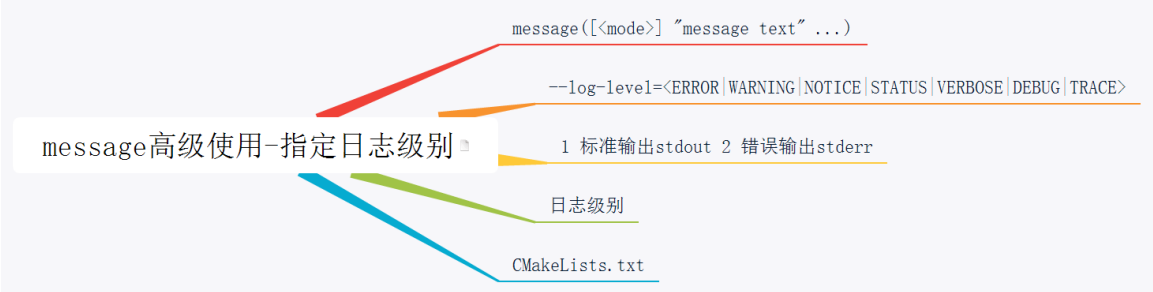
2. 2.2 cmake message详解

2.1. message基础使用



2.1.1. message（arg1 arg2 arg3 ）

2.2. message高级使用-指定日志级别

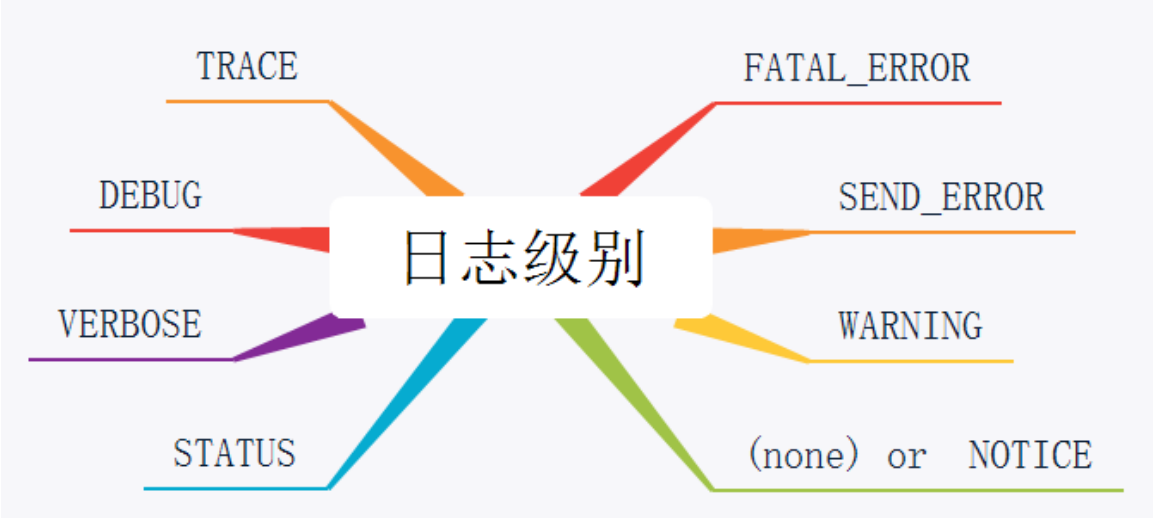


2.2.1. message([<mode>] "message text" ...)

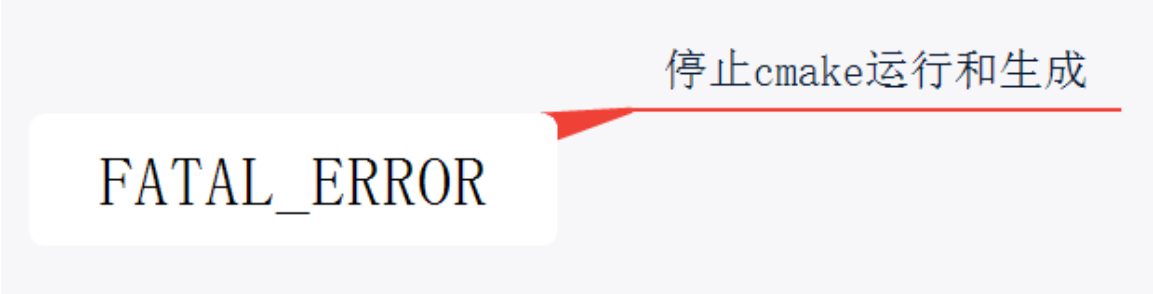
2.2.2. --log-level=<ERROR|WARNING|NOTICE|STATUS|VERBOSE|DEBUG|TRACE>

2.2.3. 1 标准输出stdout 2 错误输出stderr

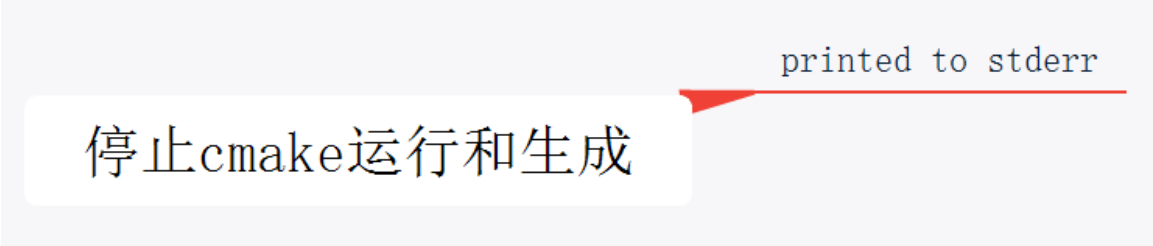
2.2.4. 日志级别



FATAL_ERROR

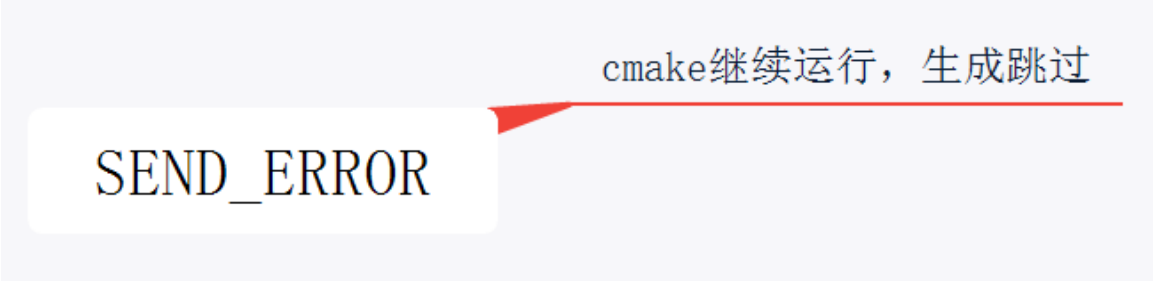


停止cmake运行和生成

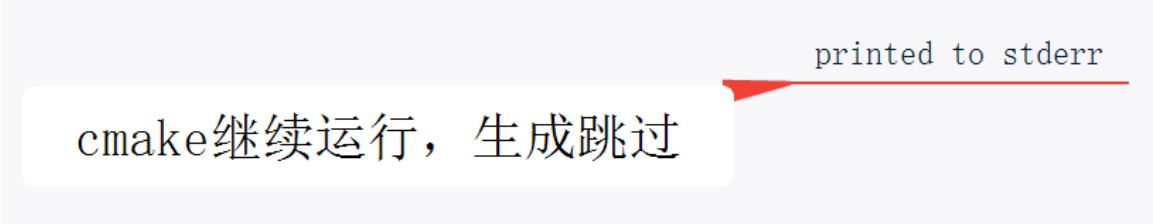


printed to stderr

SEND_ERROR



cmake继续运行，生成跳过

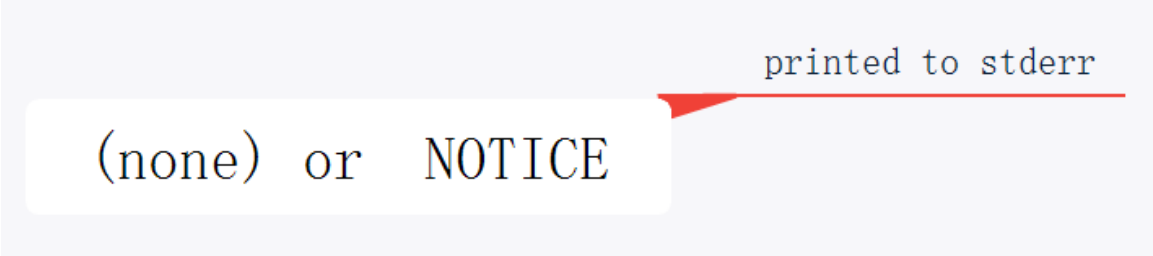


printed to stderr

WARNING

printed to stderr

(none) or NOTICE



printed to stderr

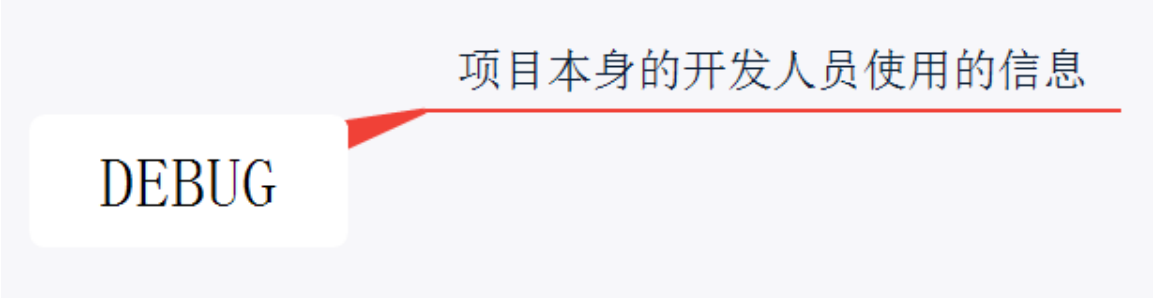
STATUS

项目用户可能感兴趣的信息

VERBOSE

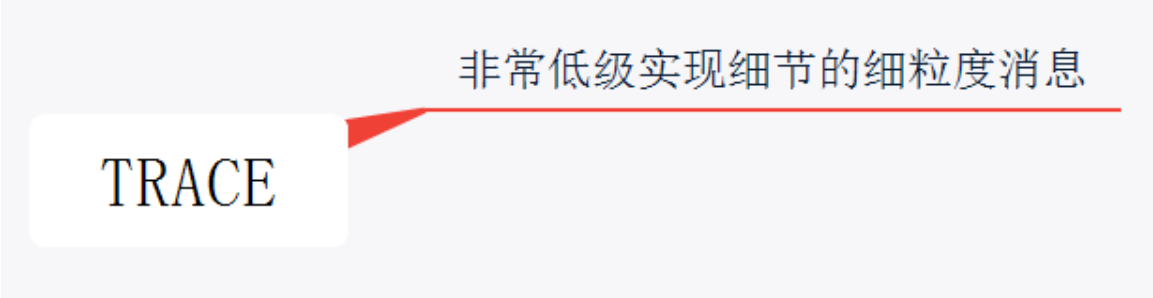
针对项目用户的详细信息

DEBUG



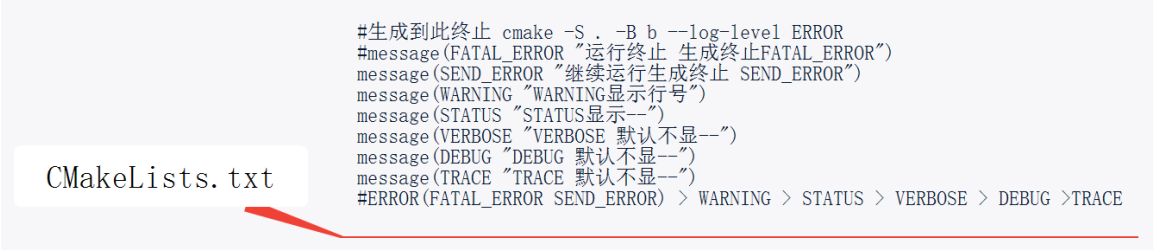
项目本身的开发人员使用的信息

TRACE



非常低级实现细节的细粒度消息

2.2.5. CMakeLists.txt



#生成到此终止 cmake -S . -B b --log-level ERROR

#message(FATAL_ERROR "运行终止 生成终止FATAL_ERROR")

message(SEND_ERROR "继续运行生成终止 SEND_ERROR")

message(WARNING "WARNING显示行号")

message(STATUS "STATUS显示--")

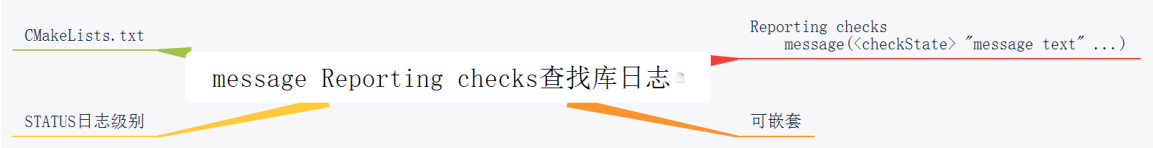
message(VERBOSE "VERBOSE 默认不显--")

message(DEBUG "DEBUG 默认不显--")

message(TRACE "TRACE 默认不显--")

#ERROR(FATAL_ERROR SEND_ERROR) > WARNING > STATUS > VERBOSE > DEBUG >TRACE

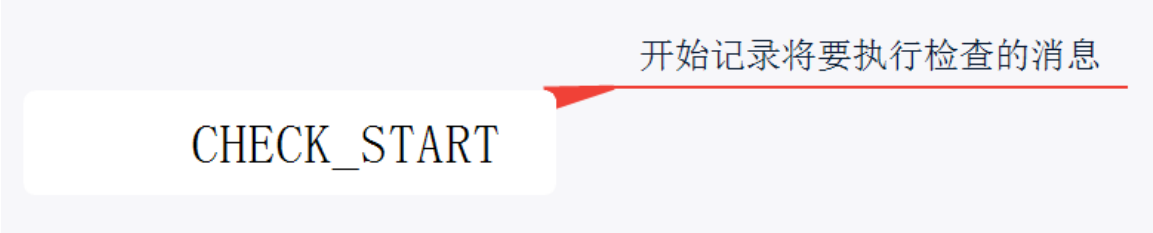
2.3. message Reporting checks查找库日志



2.3.1. Reporting checks

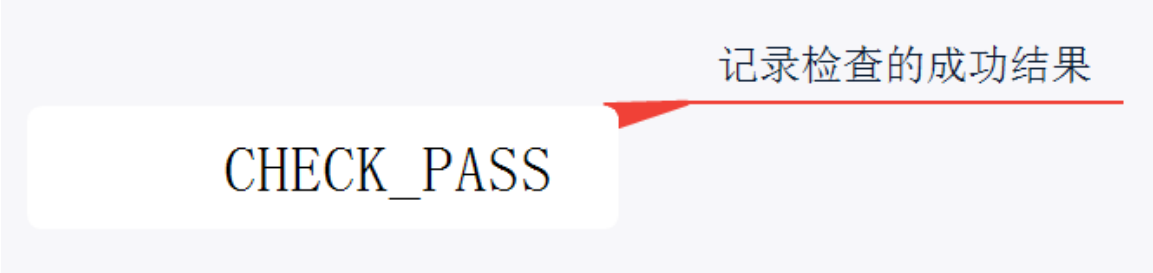
message(<checkState> "message text" ...)

CHECK_START



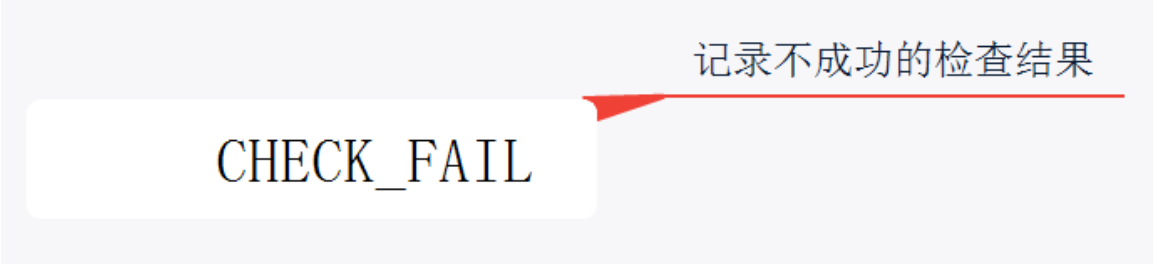
开始记录将要执行检查的消息

CHECK_PASS



记录检查的成功结果

CHECK_FAIL



记录不成功的检查结果

2.3.2. 可嵌套

2.3.3. STATUS日志级别

2.3.4. CMakeLists.txt

CMakeLists.txt

```
message("CMAKE_MESSAGE_INDENT = " ${CMAKE_MESSAGE_INDENT})
set(CMAKE_MESSAGE_INDENT " ## ") # 消息对齐
message(CHECK_START "Finding xcpp")
unset(miss)
message(CHECK_START "Finding xlog")
# ... do check, assume we find xlog
message(CHECK_PASS "found")

message(CHECK_START "Finding xthread")
# ... do check, assume we don't find xthread
set(miss ${miss}[xthread])
message(CHECK_FAIL "not found")

message(CHECK_START "Finding xsocket")
# ... do check, assume we don't find xsocket
set(miss ${miss}[xsocket])
message(CHECK_FAIL "not found")
set(CMAKE_MESSAGE_INDENT "")
if(miss)
  message(CHECK_FAIL "丢失组件: ${miss}")
else()
  message(CHECK_PASS "all components found")
endif()
```

message("CMAKE_MESSAGE_INDENT = " ${CMAKE_MESSAGE_INDENT})

set(CMAKE_MESSAGE_INDENT " ## ") # 消息对齐

message(CHECK_START "Finding xcpp")

unset(miss)

message(CHECK_START "Finding xlog")

 # ... do check, assume we find xlog

message(CHECK_PASS "found")

message(CHECK_START "Finding xthread")

 # ... do check, assume we don't find xthread

set(miss ${miss}[xthread])

message(CHECK_FAIL "not found")

message(CHECK_START "Finding xsocket")

 # ... do check, assume we don't find xsocket

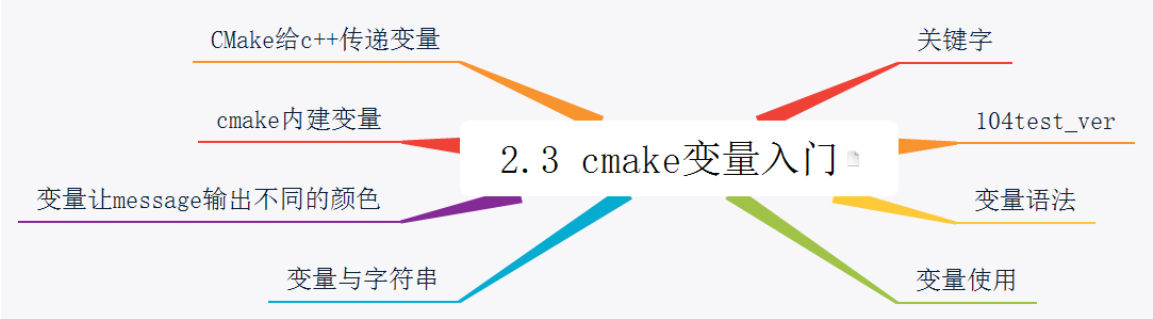
```
set(miss ${miss}[xsocket])
message(CHECK_FAIL "not found")
set(CMAKE_MESSAGE_INDENT "")
if(miss)
  message(CHECK_FAIL "丢失组件: ${miss}")
else()
  message(CHECK_PASS "all components found")
endif()
```

2.4. 关键词

2.4.1. cmake message

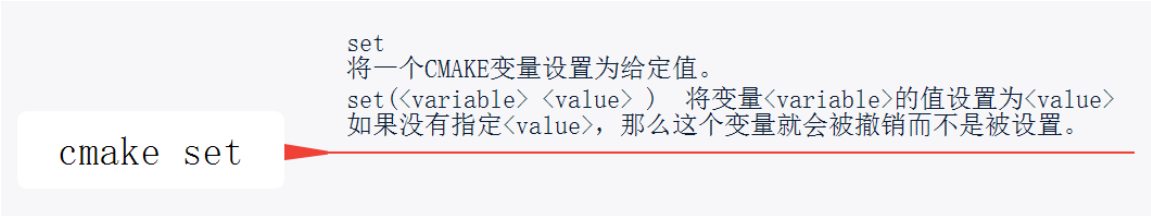
2.5. 103message

3. 2.3 cmake变量入门



3.1. 关键字

3.1.1. cmake set



set

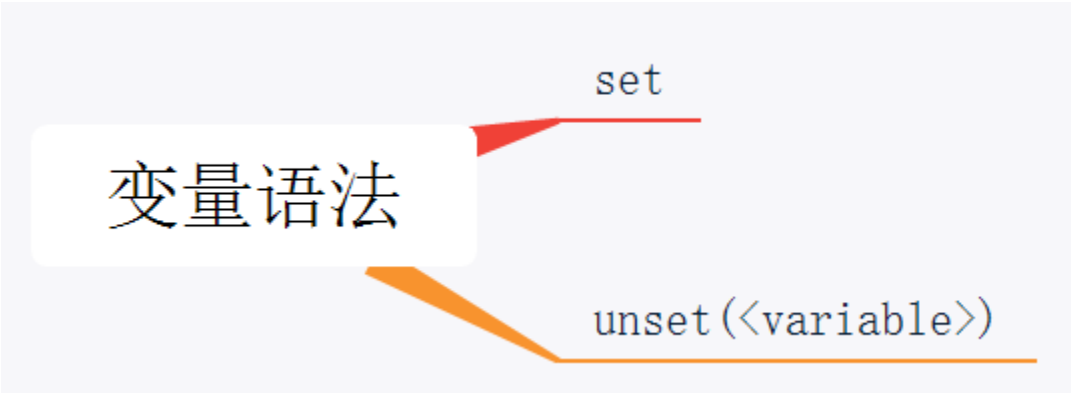
将□个CMAKE变量设置为给定值。

set(<variable> <value>) 将变量<variable>的值设置为<value>

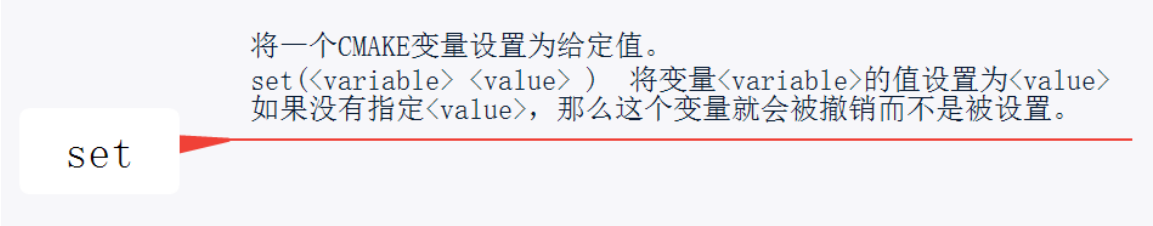
如果没有指定<value>，那么这个变量就会被撤销□不是被设置。

3.2. 104test_ver

3.3. 变量语法



3.3.1. set



将一个CMAKE变量设置为给定值。

set(<variable> <value>) 将变量<variable>的值设置为<value>

如果没有指定<value>，那么这个变量就会被撤销而不是被设置。

3.3.2. unset(<variable>)

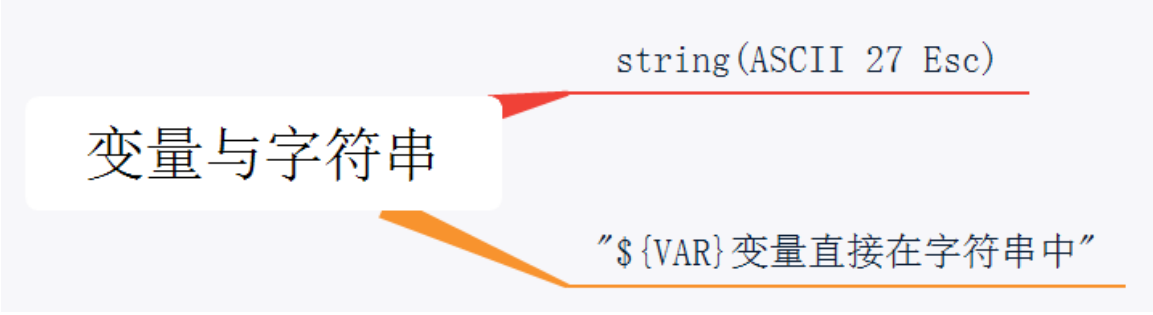
3.4. 变量使用

3.4.1. 变量引用是值替换，如果未设置变量，返回空字符串

3.4.2. 变量引用可以嵌套并从内向外求值

3.4.3. 变量名大小写敏感

3.5. 变量与字符串



3.5.1. string(ASCII 27 Esc)

3.5.2. "${VAR}变量直接在字符串中"

3.6. 变量让message输出不同的颜色

3.6.1. string(ASCII 27 Esc)

3.6.2. \033[1;31;40m <!--1-高亮显示 31-前景色红色 40-背景色黑色-->

\033[0m <!--采用终端默认设置，即取消颜色设置-->

3.6.3. Windows PowerShell

3.6.4. code

code

```
cmake_minimum_required(VERSION 3.20)
project("test_v1" )
string(ASCII 27 Esc)
string(ASCII 70 A)
message(${A})
#[[
格式：\33[显示方式;前景色;背景色m
显示方式              意义
-------------------------
0                   终端默认设置
1                   高亮显示
4                   使用下划线
5                   闪烁
7                   反白显示
8                   不可见
\033[1;31;40m     <!--1-高亮显示 31-前景色红色  40-背景色黑色-->
前景色              背景色              颜色
---------------------------------------
30                  40                  黑色
31                  41                  红色
32                  42                  绿色
33                  43                  黄色
34                  44                  蓝色
35                  45                  紫红色
36                  46                  青蓝色
37                  47                  白色
]]
message("33")
set(E "${Esc}[m")
set(R  "${Esc}[31m")
set(G  "${Esc}[32m")
set(Y  "${Esc}[33m")
set(B  "${Esc}[34m")
set(RB  "${Esc}[4;31;40m") #红黑
message("${R}这是红色${E}")
message("${G}这是绿色${E}")
message("${Y}这是黄色${E}")
message("${B}这是蓝色${E}")
message("${RB}这是红黑${E}")
```

cmake_minimum_required(VERSION 3.20)

project("test_v1")

string(ASCII 27 Esc)

string(ASCII 70 A)

message(${A})

#[[

格式：\33[显示方式;前景色;背景色m

显示方式　　意义

0　　终端默认设置

1 高亮显示

4 使用下划线

5 闪烁

7 反白显示

8 不可见

\033[1;31;40m <!--1-高亮显示 31-前景色红色 40-背景色黑色-->

前景色 背景色 颜色

---

30 40 黑色

31 41 红色

32 42 绿色

33 43 黄色

34 44 蓝色

35 45 紫红色

36 46 青蓝色

37 47 白色

]]

```
message("33")
set(E "${Esc}[m")
set(R  "${Esc}[31m")
set(G  "${Esc}[32m")
set(Y  "${Esc}[33m")
set(B  "${Esc}[34m")
set(RB  "${Esc}[4;31;40m") #红黑
message("${R}这是红色${E}")
message("${G}这是绿色${E}")
message("${Y}这是黄色${E}")
```

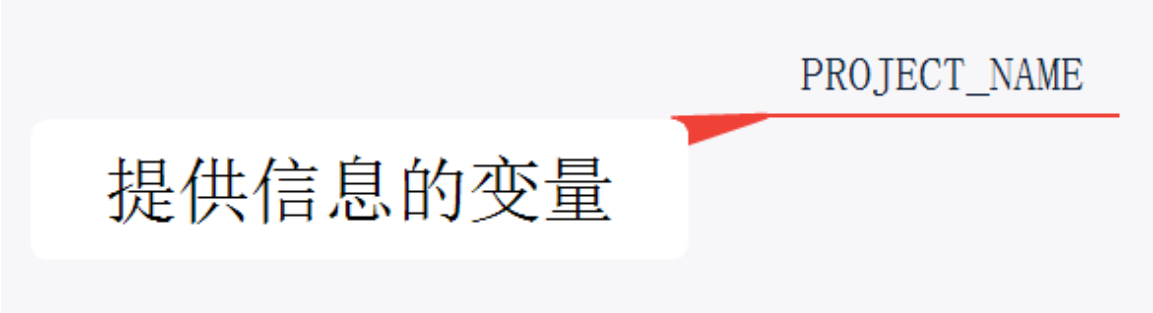
message("${B}这是蓝色${E}")

message("${RB}这是红黑${E}")

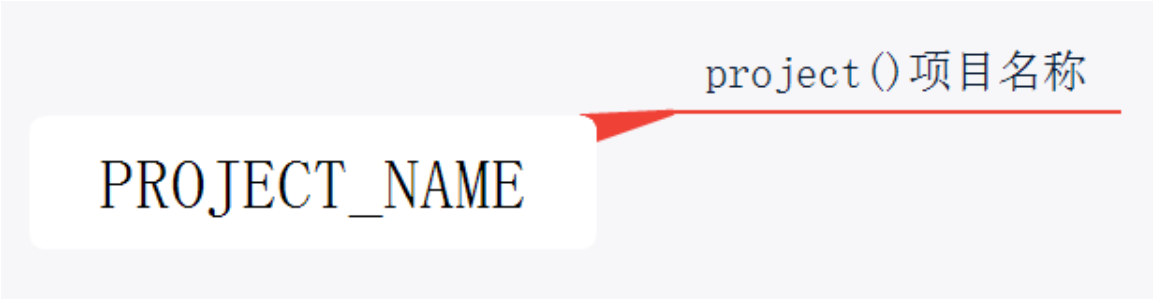
3.6.5. 105message_color

3.7. cmake内建变量

3.7.1. 提供信息的变量

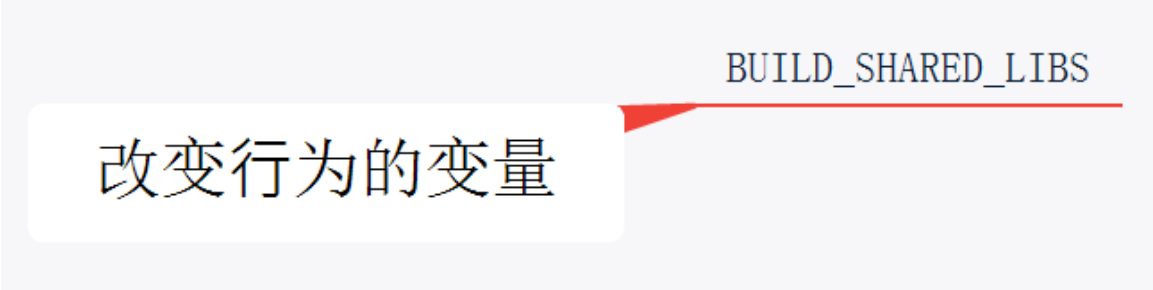


PROJECT_NAME



project()项目名称

3.7.2. 改变为的变量



BUILD_SHARED_LIBS

缓存变量

add_library()

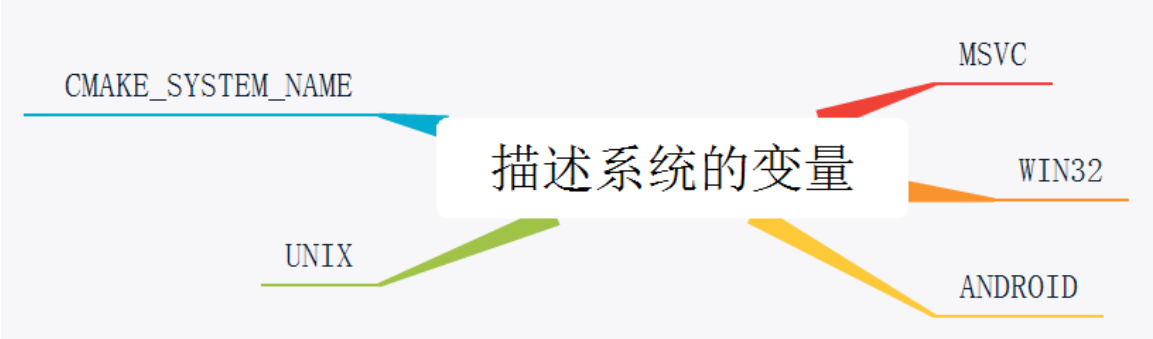
ON

创建共享库

OFF

创建静态库

3.7.3. 描述系统的变量

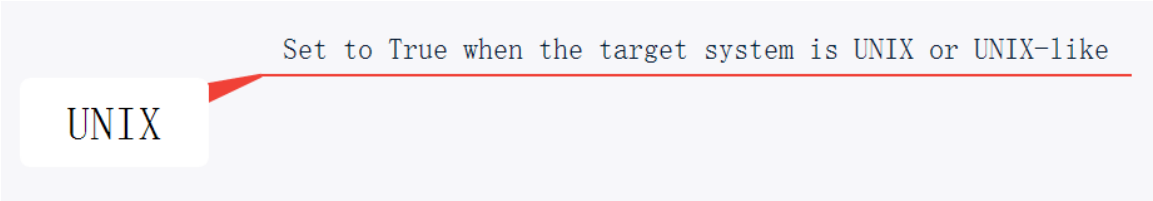


MSVC

WIN32

ANDROID

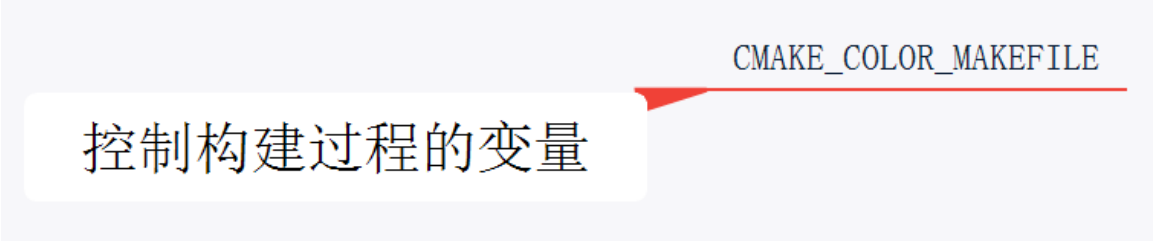
UNIX



Set to True when the target system is UNIX or UNIX-like

CMAKE_SYSTEM_NAME

3.7.4. 控制构建过程的变量



CMAKE_COLOR_MAKEFILE

生成的makefile是否有颜色，默认是ON

3.7.5. 项目代码

CMakeLists.h

xlog.h

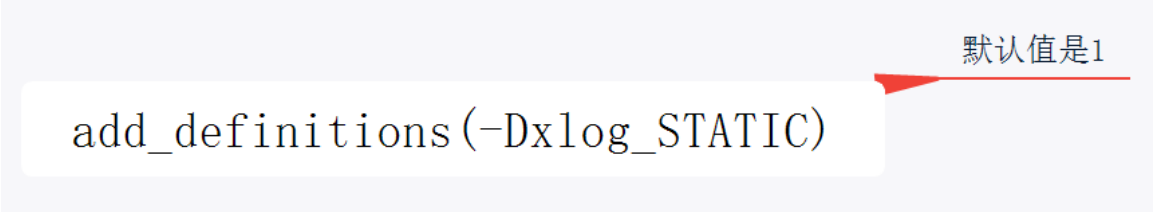
xlog.cpp

使用102cmake_lib的代码

3.7.6. http://cmake.org.cn/cmake_html/manual/cmake-variables.7.html

3.8. CMake给c++传递变量

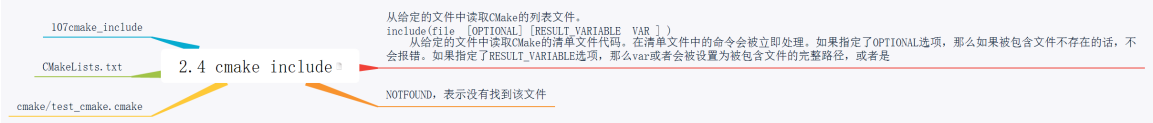
3.8.1. add_definitions(-Dxlog_STATIC)



默认值是1

3.8.2. add_definitions(-DSTATIC=1)

4. 2.4 cmake include



4.1. 从给定的□件中读取CMake的列表□件。

include(file [OPTIONAL] [RESULT_VARIABLE VAR])

□□

从给定的□件中读取CMake的清单□件代码。在清单□件中的命令会被□即处理。如果指定了OPTIONAL选项，那么如果被包含□件不存在的话，不会报错。如果指定了RESULT_VARIABLE选项，那么var或者会被设置为被包含□件的完整路径，或者是

4.2. NOTFOUND，表示没有找到该□件

4.3. cmake/test_cmake.cmake

cmake/test_cmake.cmake

```
message("in test cmake ")
```

4.3.1. message("in test cmake ")

4.4. CMakeLists.txt

CMakeLists.txt

```
cmake_minimum_required (VERSION 3.0)
project("test_include")
message("begin include")
include(cmake/test_cmake.cmake )
include(cmake/test_cmake.cmake )
include(cmake/test_cmake1.cmake OPTIONAL) # OPTIONAL 文件不存在，不报错
include(cmake/test_cmake1.cmake OPTIONAL RESULT_VARIABLE ret_val ) # NOTFOUND
MESSAGE("install return value is ${ret_val}")
include(cmake/test_cmake.cmake OPTIONAL RESULT_VARIABLE ret_val ) # NOTFOUND
MESSAGE("install return value is ${ret_val}") # 返回文件全路径
message("after include")
```

4.4.1. cmake_minimum_required (VERSION 3.0)

project("test_include")

message("begin include")

include(cmake/test_cmake.cmake)

include(cmake/test_cmake.cmake)

include(cmake/test_cmake1.cmake OPTIONAL) # OPTIONAL 件不存在，不报错

include(cmake/test_cmake1.cmake OPTIONAL RESULT_VARIABLE ret_val) # NOTFOUND

MESSAGE("install return value is ${ret_val}")

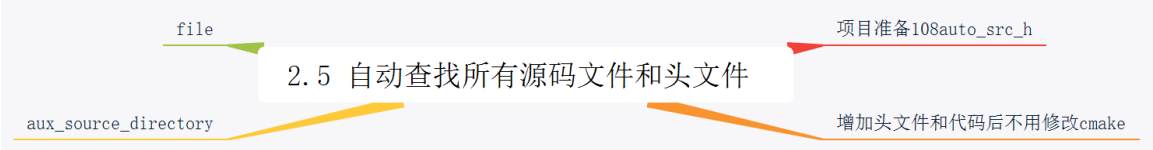
include(cmake/test_cmake.cmake OPTIONAL RESULT_VARIABLE ret_val) # NOTFOUND

MESSAGE("install return value is ${ret_val}") # 返回文件全路径

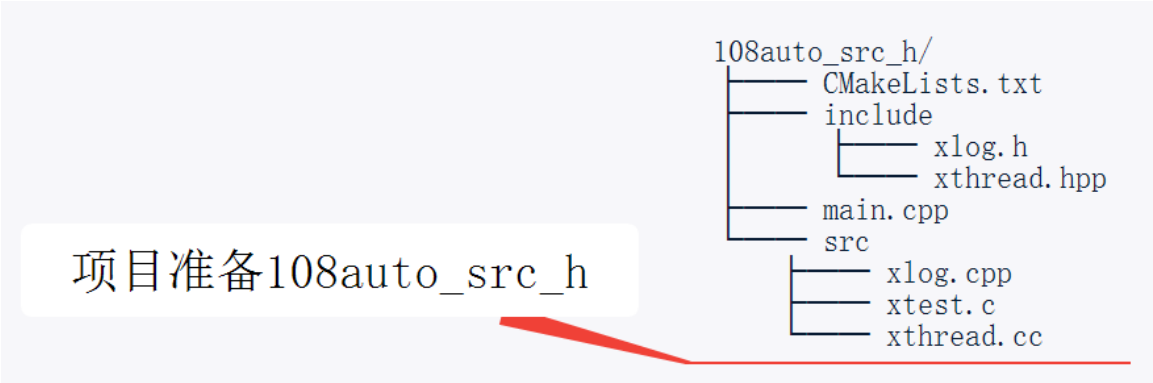
message("after include")

4.5. 107cmake_include

5. 2.5 自动查找所有源码文件和头文件



5.1. 项目准备108auto_src_h

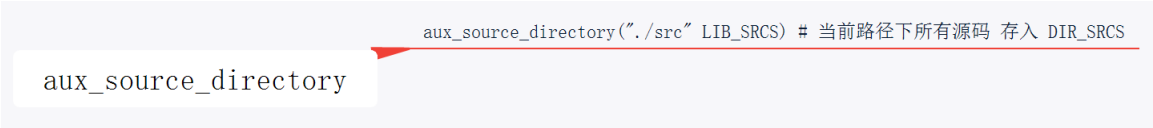


5.1.1. 108auto_src_h/

```
├── CMakeLists.txt
├── include
│   ├── xlog.h
│   └── xthread.hpp
├── main.cpp
└── src
    ├── xlog.cpp
    ├── xtest.c
    └── xthread.cc
```

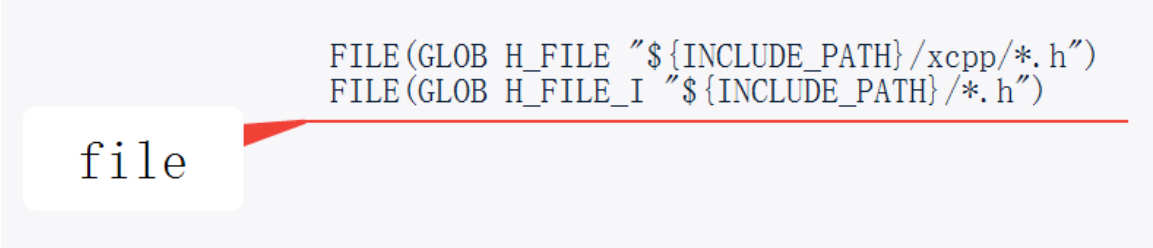
5.2. 增加头文件和代码后不用修改cmake

5.3. aux_source_directory



5.3.1. aux_source_directory("./src" LIB_SRCS) # 当前路径下所有源码 存入 DIR_SRCS

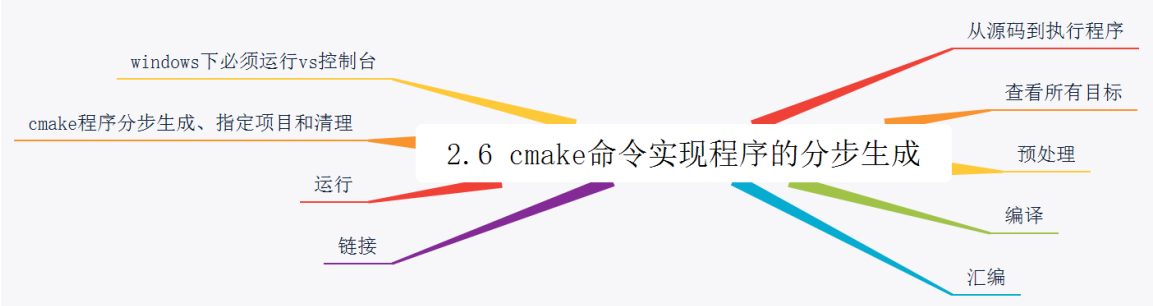
5.4. file



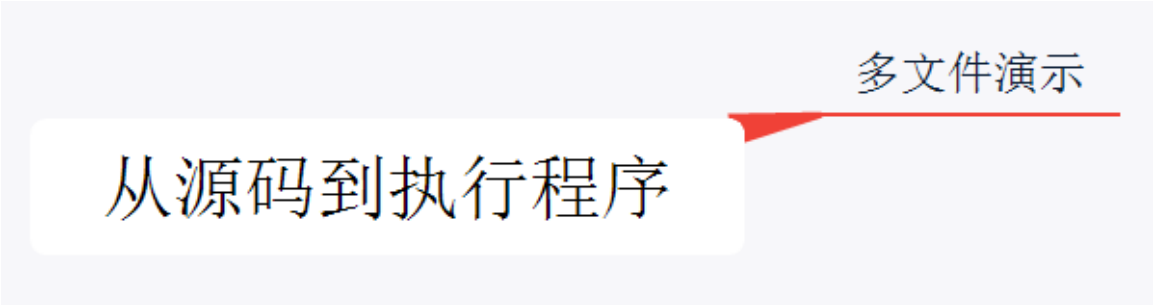
5.4.1. FILE(GLOB H_FILE "${INCLUDE_PATH}/xcpp/*.h")

FILE(GLOB H_FILE_I "${INCLUDE_PATH}/*.h")

6. 2.6 cmake命令实现程序的分步生成

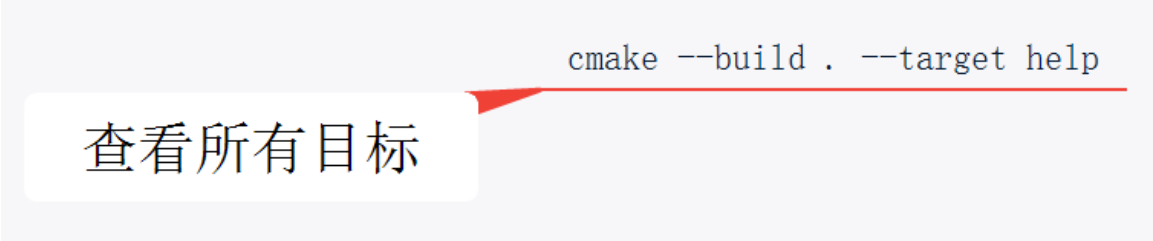


6.1. 从源码到执行程序



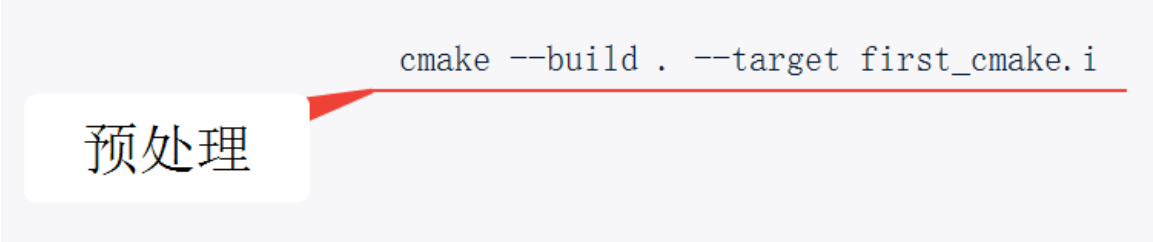
6.1.1. 多文件演示

6.2. 查看所有目标



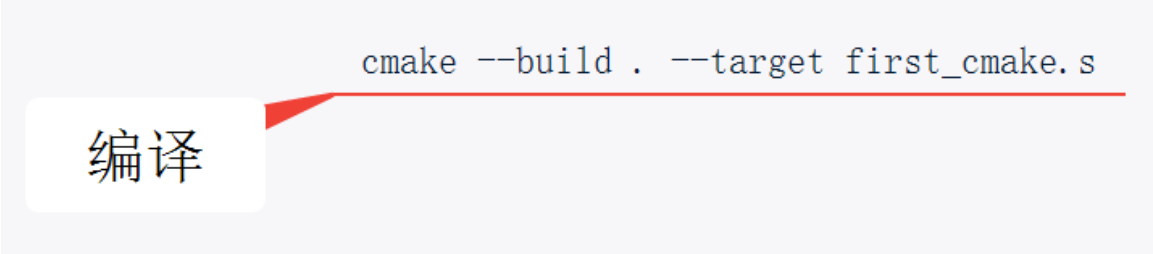
6.2.1. cmake --build . --target help

6.3. 预处理



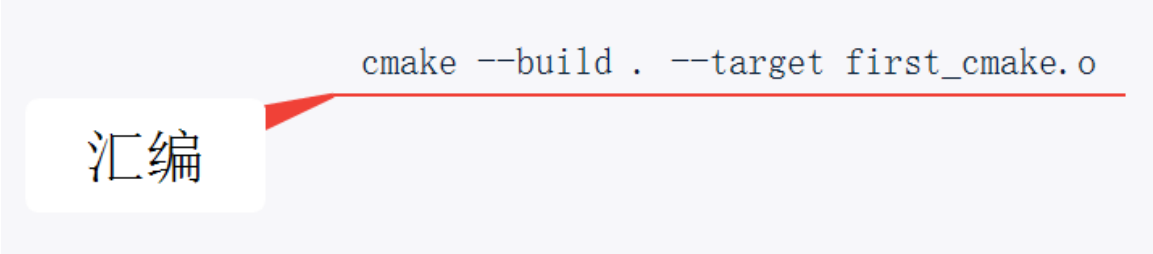
6.3.1. cmake --build . --target first_cmake.i

6.4. 编译



6.4.1. cmake --build . --target first_cmake.s

6.5. 汇编



6.5.1. cmake --build . --target first_cmake.o

6.6. 链接

6.7. 运行

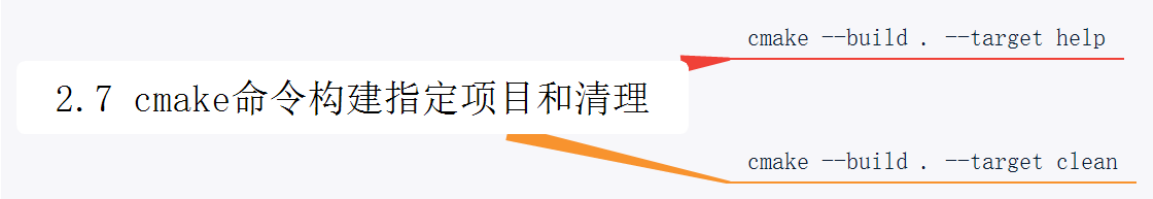
6.7.1. 动态库加载路径

6.8. cmake程序分步生成、指定项目和清理

6.9. windows下必须运行vs控制台

6.9.1. cmake -S . -B nmake -G "NMake Makefiles"

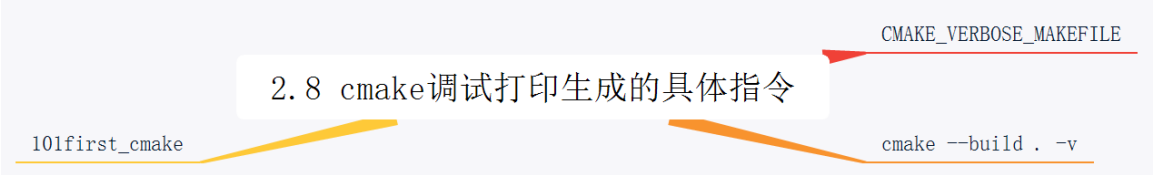
7. 2.7 cmake命令构建指定项目和清理



7.1. cmake --build . --target help

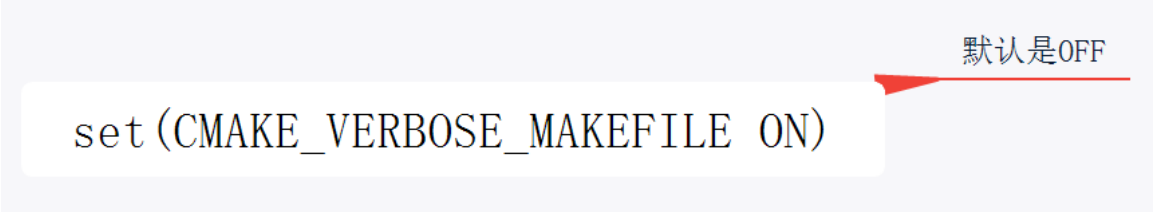
7.2. cmake --build . --target clean

8. 2.8 cmake调试打印生成的具体指令



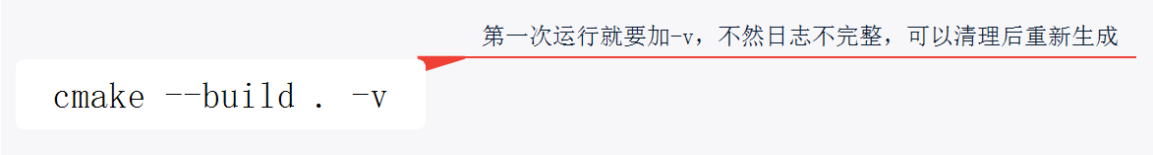
8.1. CMAKE_VERBOSE_MAKEFILE

8.1.1. set(CMAKE_VERBOSE_MAKEFILE ON)



默认是OFF

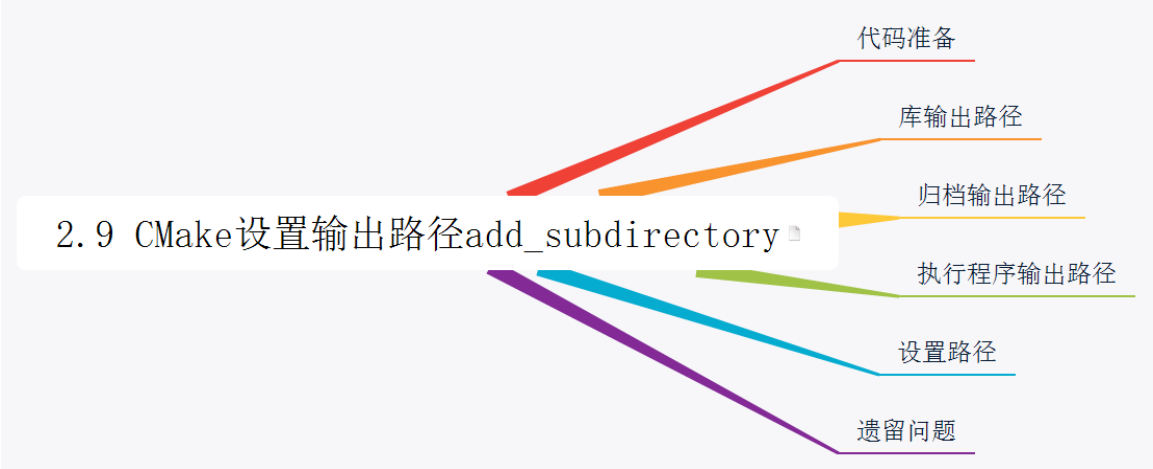
8.2. cmake --build . -v



8.2.1. 第一次运行就要加-v，不然日志不完整，可以清理后重新生成

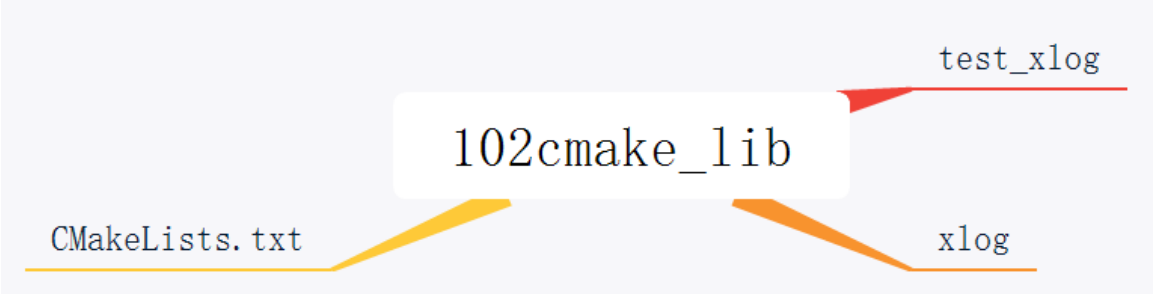
8.3. 101first_cmake

9. 2.9 CMake设置输出路径add_subdirectory



9.1. 代码准备

9.1.1. 102cmake_lib

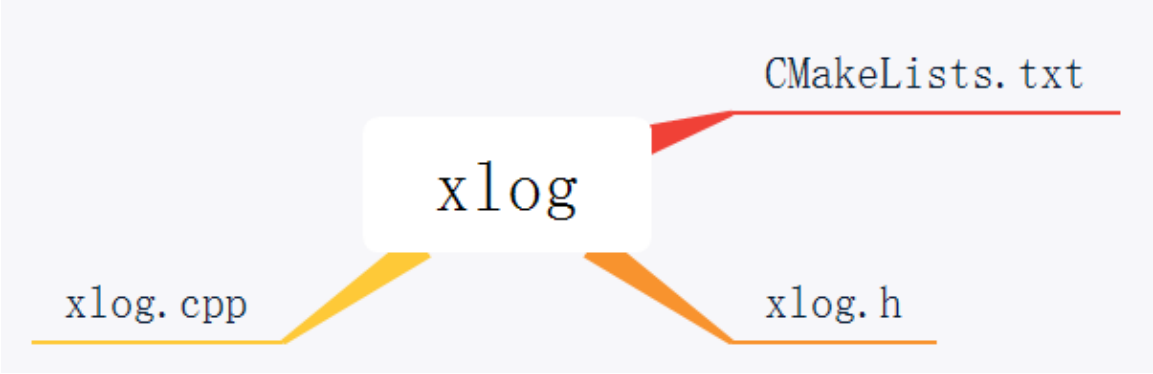


test_xlog

CMakeLists.txt

test_xlog.cpp

xlog



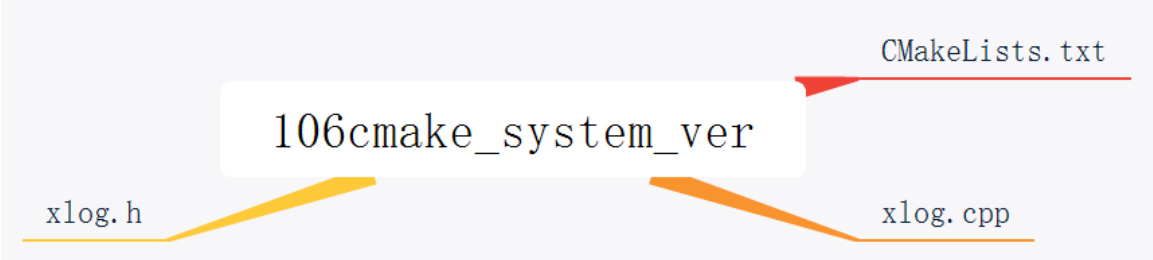
CMakeLists.txt

xlog.h

xlog.cpp

CMakeLists.txt

9.1.2. 106cmake_system_ver

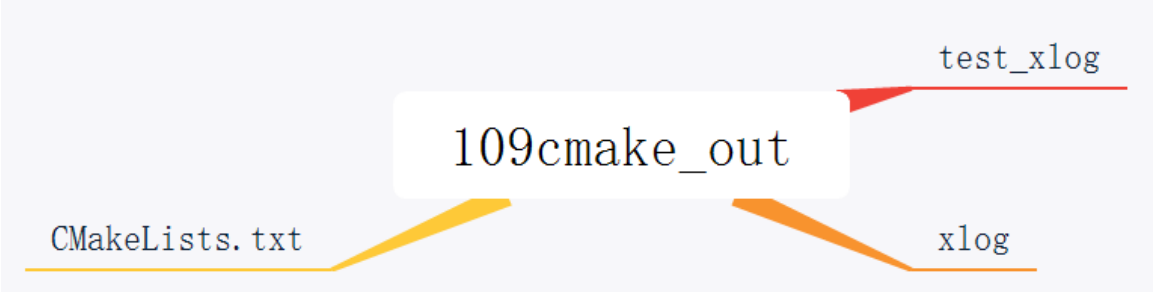


CMakeLists.txt

xlog.cpp

xlog.h

9.1.3. 109cmake_out



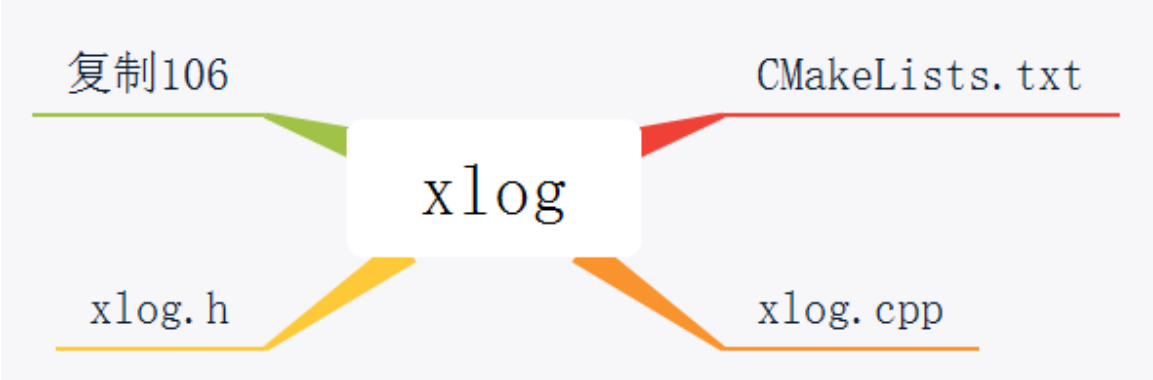
test_xlog

CMakeLists.txt

test_xlog.cpp

复制102

xlog



CMakeLists.txt

xlog.cpp

xlog.h

复制106

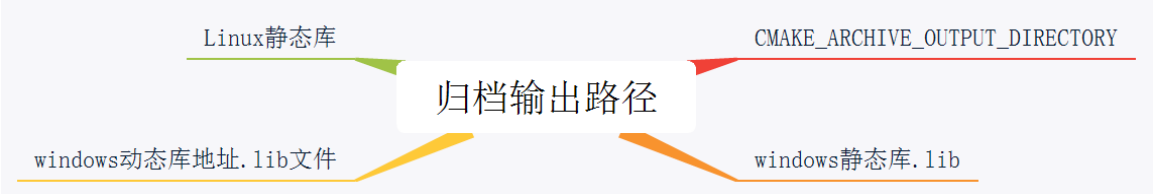
CMakeLists.txt

9.2. 库输出路径

9.2.1. CMAKE_LIBRARY_OUTPUT_DIRECTORY

9.2.2. linux动态库 .so

9.3. 归档输出路径

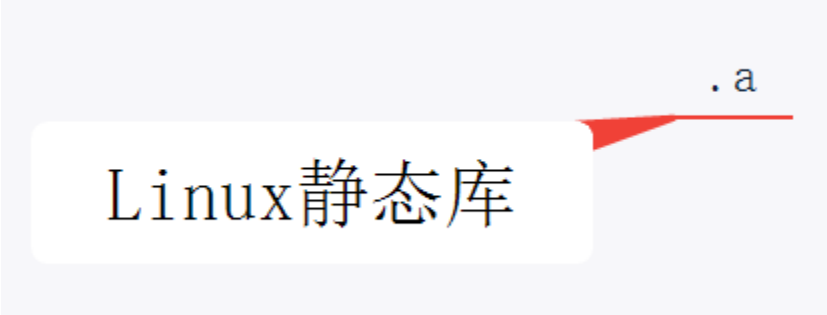


9.3.1. CMAKE_ARCHIVE_OUTPUT_DIRECTORY

9.3.2. windows静态库.lib

9.3.3. windows动态库地址.lib文件

9.3.4. Linux静态库



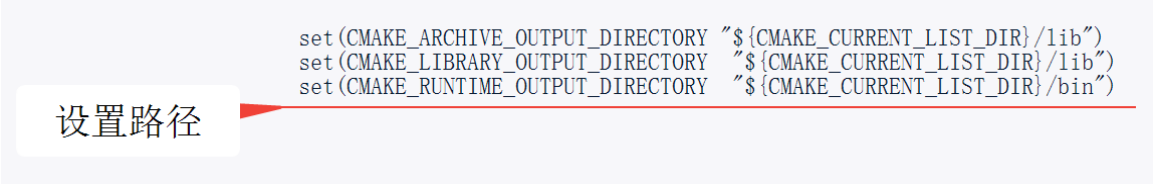
.a

9.4. 执行程序输出路径

9.4.1. CMAKE_RUNTIME_OUTPUT_DIRECTORY

9.4.2. 执行程序和dll动态库

9.5. 设置路径



9.5.1. set(CMAKE_ARCHIVE_OUTPUT_DIRECTORY "${CMAKE_CURRENT_LIST_DIR}/lib")

set(CMAKE_LIBRARY_OUTPUT_DIRECTORY "${CMAKE_CURRENT_LIST_DIR}/lib")

set(CMAKE_RUNTIME_OUTPUT_DIRECTORY "${CMAKE_CURRENT_LIST_DIR}/bin")

9.6. 遗留问题

9.6.1. 多个项目不同输出路径

9.6.2. Debug和Release不同输出

9.6.3. 一个项目同时要设置静态库和动态库