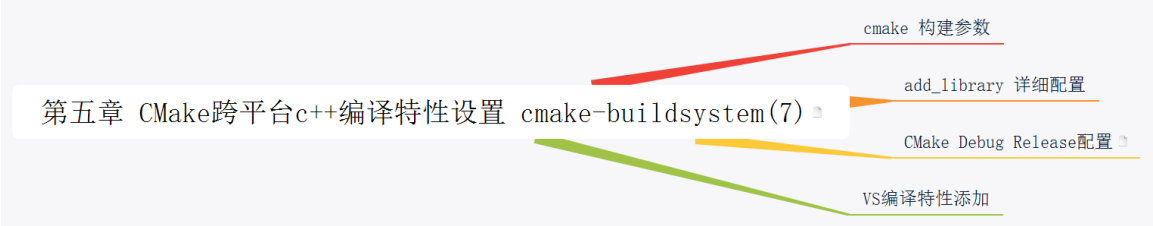
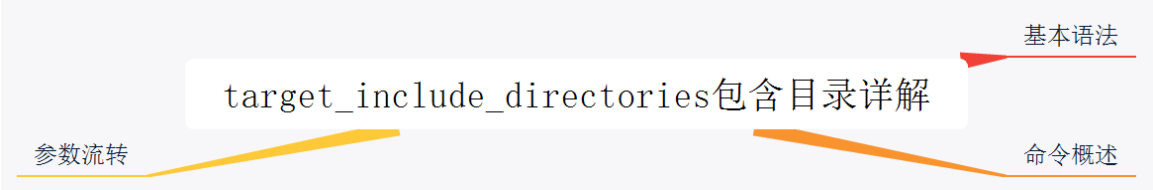
第五章 CMake跨平台c++编译特性设置 cmake-buildsystem(7)

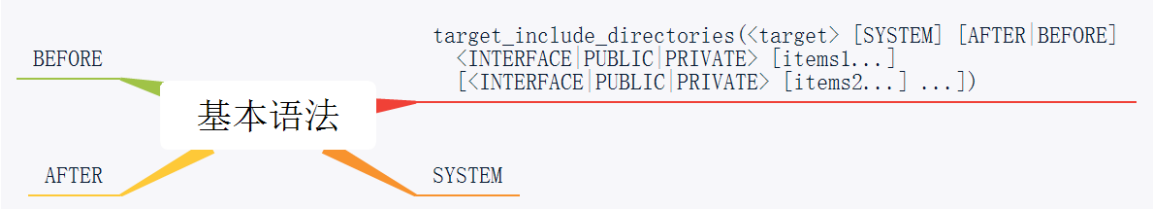


1. cmake 构建参数

1.1. target_include_directories包含目录详解



1.1.1. 基本语法

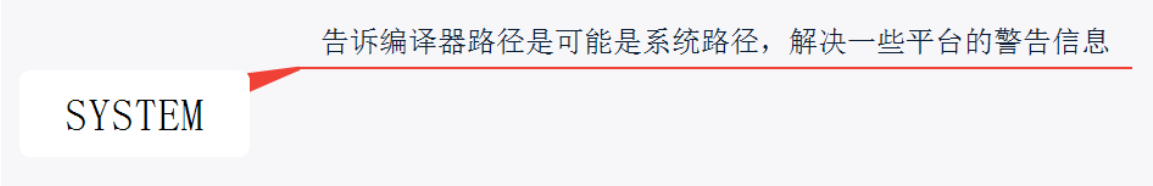


target_include_directories(<target> [SYSTEM] [AFTER|BEFORE]

<INTERFACE|PUBLIC|PRIVATE> [items1...]

[<INTERFACE|PUBLIC|PRIVATE> [items2...] ...])

SYSTEM

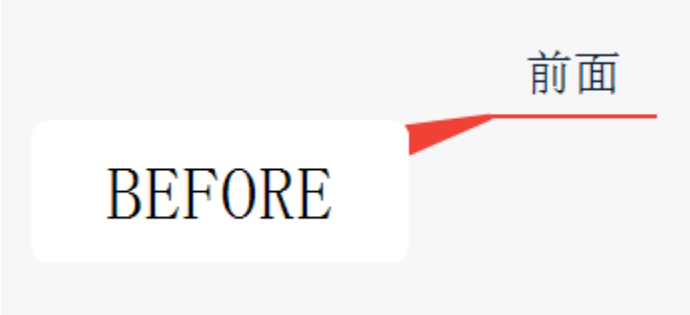


告诉编译器路径是可能是系统路径，解决一些平台的警告信息

AFTER

最后

BEFORE



前面

1.1.2. 命令概述

指定目标要使用的包含目录

名称<target>必须是由命令创建的，不能是Alias Targets别名目标

例如add_executable()或者add_library()

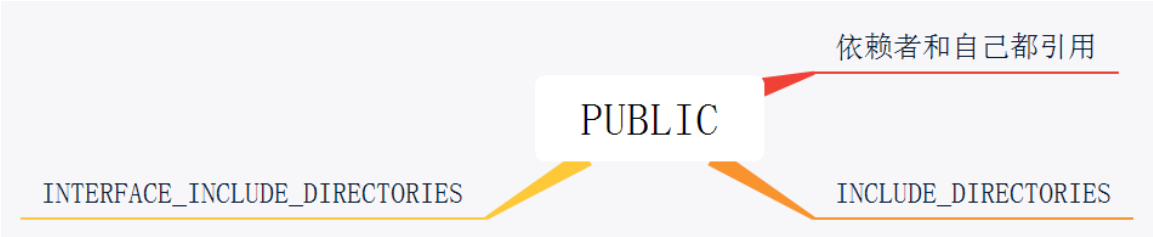
1.1.3. 参数流转

INTERFACE

只有依赖者引用

INTERFACE_INCLUDE_DIRECTORIES

PUBLIC



依赖者和自己都引用

INCLUDE_DIRECTORIES

INTERFACE_INCLUDE_DIRECTORIES

PRIVATE

只有自己用

INCLUDE_DIRECTORIES

1.2. target_link_libraries导入依赖库

1.2.1. target_link_libraries(<target>

<PRIVATE|PUBLIC|INTERFACE> <item>...

[<PRIVATE|PUBLIC|INTERFACE> <item>...]...)

1.2.2. $<TARGET_OBJECTS:xlog> 依赖的头文件路径、宏定义等没有

1.2.3. INTERFACE

只有依赖者引用

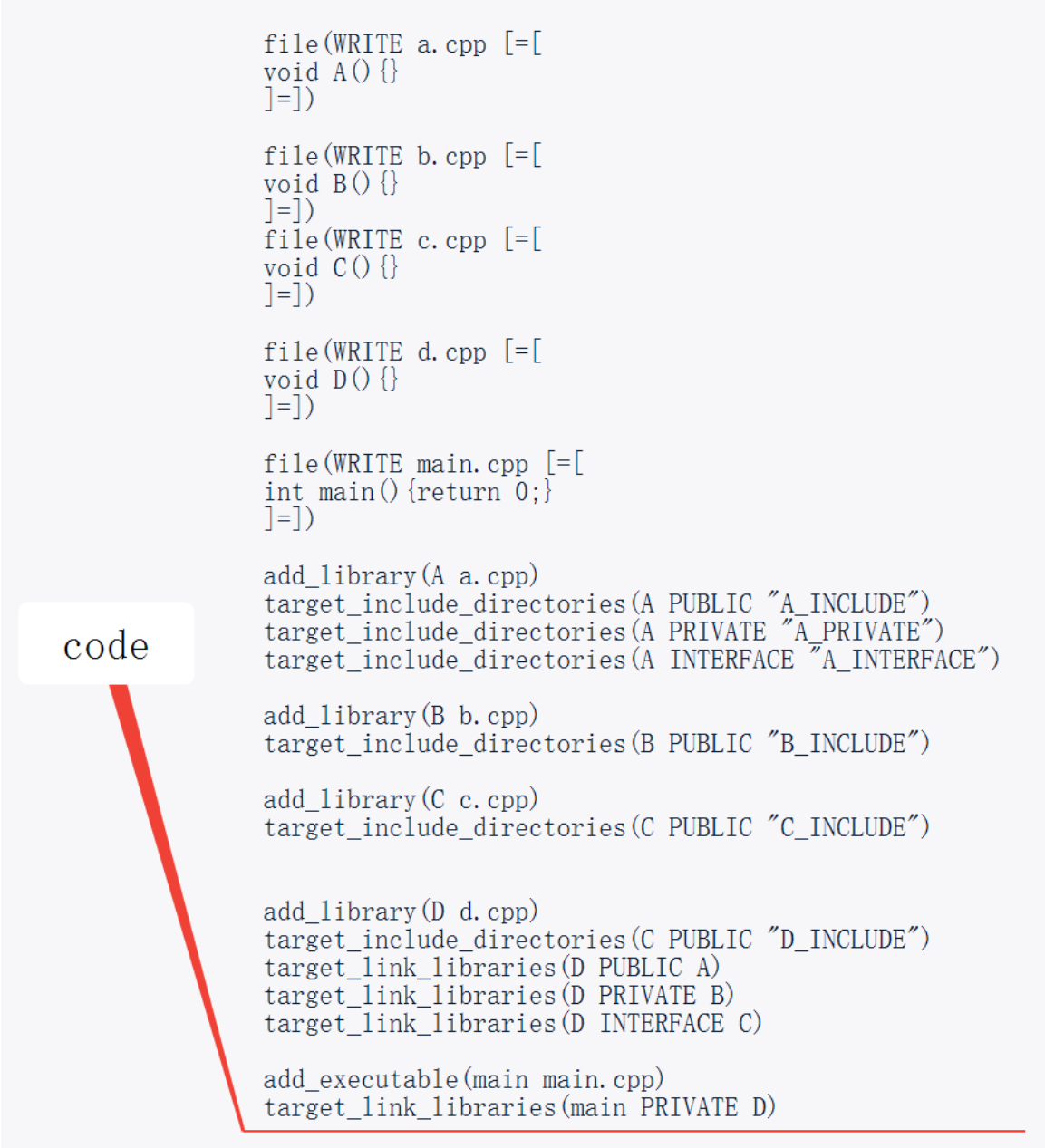
1.2.4. PUBLIC

依赖者和自己都引用

1.2.5. PRIVATE

只有自己用

1.2.6. code



file(WRITE a.cpp [=[

void A(){}

]=])

file(WRITE b.cpp [=[

void B(){}

]=])

```
file(WRITE c.cpp [=[
void C(){}
]=])

file(WRITE d.cpp [=[
void D(){}
]=])

file(WRITE main.cpp [=[
int main(){return 0;}
]=])

add_library(A a.cpp)
target_include_directories(A PUBLIC "A_INCLUDE")
target_include_directories(A PRIVATE "A_PRIVATE")
target_include_directories(A INTERFACE "A_INTERFACE")

add_library(B b.cpp)
target_include_directories(B PUBLIC "B_INCLUDE")

add_library(C c.cpp)
target_include_directories(C PUBLIC "C_INCLUDE")


add_library(D d.cpp)
target_include_directories(C PUBLIC "D_INCLUDE")
target_link_libraries(D PUBLIC A)
target_link_libraries(D PRIVATE B)
target_link_libraries(D INTERFACE C)
```

add_executable(main main.cpp)

target_link_libraries(main PRIVATE D)

1.3. target_compile_definitions()编译传递宏

1.3.1. COMPILE_DEFINITIONS

1.3.2. INTERFACE_COMPILE_DEFINITIONS

1.3.3. target_compile_definitions(foo PUBLIC FOO)

target_compile_definitions(foo PUBLIC -DFOO) # -D removed

target_compile_definitions(foo PUBLIC "" FOO) # "" ignored

target_compile_definitions(foo PUBLIC -D FOO) # -D becomes "", then ignored

1.3.4. target_compile_definitions(<target>

<INTERFACE|PUBLIC|PRIVATE> [items1...]

[<INTERFACE|PUBLIC|PRIVATE> [items2...] ...])

1.4. target_compile_features c++ 11 14 17 20 22

1.4.1. target_compile_features(<target> <PRIVATE|PUBLIC|INTERFACE> <feature> [...])

1.4.2. foreach(var IN LISTS CMAKE_CXX_COMPILE_FEATURES)

message(${var})

endforeach()

1.4.3. vs2022


vs2022
cxx_std_98
cxx_template_template_parameters
cxx_std_11
cxx_alias_templates
cxx_alignas
cxx_alignof
cxx_attributes
cxx_auto_type
cxx_constexpr
cxx_decltype
cxx_decltype_incomplete_return_types
cxx_default_function_template_args
cxx_defaulted_functions
cxx_defaulted_move_initializers
cxx_delegating_constructors
cxx_deleted_functions
cxx_enum_forward_declarations
cxx_explicit_conversions
cxx_extended_friend_declarations
cxx_extern_templates
cxx_final
cxx_func_identifier
cxx_generalized_initializers
cxx_inheriting_constructors
cxx_inline_namespaces
cxx_lambdas
cxx_local_type_template_args
cxx_long_long_type
cxx_noexcept
cxx_nonstatic_member_init
cxx_nullptr
cxx_override
cxx_range_for
cxx_raw_string_literals
cxx_reference_qualified_functions
cxx_right_angle_brackets
cxx_rvalue_references
cxx_sizeof_member
cxx_static_assert
cxx_strong_enums
cxx_thread_local
cxx_trailing_return_types
cxx_unicode_literals
cxx_uniform_initialization
cxx_unrestricted_unions
cxx_user_literals
cxx_variadic_macros
cxx_variadic_templates
cxx_std_14
cxx_aggregate_default_initializers
cxx_attribute_deprecated
cxx_binary_literals
cxx_contextual_conversions
cxx_decltype_auto
cxx_digit_separators
cxx_generic_lambdas
cxx_lambda_init_captures
cxx_relaxed_constexpr
cxx_return_type_deduction
cxx_variable_templates
cxx_std_17
cxx_std_20
cxx_std_23

cxx_std_98

cxx_template_template_parameters

cxx_std_11

cxx_alias_templates

cxx_alignas

cxx_alignof

cxx_attributes

cxx_auto_type

cxx_constexpr

cxx_decltype

cxx_decltype_incomplete_return_types

cxx_default_function_template_args

cxx_defaulted_functions

cxx_defaulted_move_initializers

cxx_delegating_constructors

cxx_deleted_functions

cxx_enum_forward_declarations

cxx_explicit_conversions

cxx_extended_friend_declarations

cxx_extern_templates

cxx_final

cxx_func_identifier

cxx_generalized_initializers

cxx_inheriting_constructors

cxx_inline_namespaces

cxx_lambdas

cxx_local_type_template_args

cxx_long_long_type

cxx_noexcept

cxx_nonstatic_member_init

cxx_nullptr

cxx_override

cxx_range_for

cxx_raw_string_literals

cxx_reference_qualified_functions

cxx_right_angle_brackets

cxx_rvalue_references

cxx_sizeof_member

cxx_static_assert

cxx_strong_enums

cxx_thread_local

cxx_trailing_return_types

cxx_unicode_literals

cxx_uniform_initialization

cxx_unrestricted_unions

cxx_user_literals

cxx_variadic_macros

cxx_variadic_templates

cxx_std_14

cxx_aggregate_default_initializers

cxx_attribute_deprecated

cxx_binary_literals

cxx_contextual_conversions

cxx_decltype_auto

cxx_digit_separators

cxx_generic_lambdas

cxx_lambda_init_captures

cxx_relaxed_constexpr

cxx_return_type_deduction

cxx_variable_templates

cxx_std_17

cxx_std_20

cxx_std_23

1.4.4. gcc


gcc
cxx_std_98
cxx_template_template_parameters
cxx_std_11
cxx_alias_templates
cxx_alignas
cxx_alignof
cxx_attributes
cxx_auto_type
cxx_constexpr
cxx_decltype
cxx_decltype_incomplete_return_types
cxx_default_function_template_args
cxx_defaulted_functions
cxx_defaulted_move_initializers
cxx_delegating_constructors
cxx_deleted_functions
cxx_enum_forward_declarations
cxx_explicit_conversions
cxx_extended_friend_declarations
cxx_extern_templates
cxx_final
cxx_func_identifier
cxx_generalized_initializers
cxx_inheriting_constructors
cxx_inline_namespaces
cxx_lambdas
cxx_local_type_template_args
cxx_long_long_type
cxx_noexcept
cxx_nonstatic_member_init
cxx_nullptr
cxx_override
cxx_range_for
cxx_raw_string_literals
cxx_reference_qualified_functions
cxx_right_angle_brackets
cxx_rvalue_references
cxx_sizeof_member
cxx_static_assert
cxx_strong_enums
cxx_thread_local
cxx_trailing_return_types
cxx_unicode_literals
cxx_uniform_initialization
cxx_unrestricted_unions
cxx_user_literals
cxx_variadic_macros
cxx_variadic_templates
cxx_std_14
cxx_aggregate_default_initializers
cxx_attribute_deprecated
cxx_binary_literals
cxx_contextual_conversions
cxx_decltype_auto
cxx_digit_separators
cxx_generic_lambdas
cxx_lambda_init_captures
cxx_relaxed_constexpr
cxx_return_type_deduction
cxx_variable_templates
cxx_std_17
cxx_std_20

cxx_std_98

cxx_template_template_parameters

cxx_std_11

cxx_alias_templates

cxx_alignas

cxx_alignof

cxx_attributes

cxx_auto_type

cxx_constexpr

cxx_decltype

cxx_decltype_incomplete_return_types

cxx_default_function_template_args

cxx_defaulted_functions

cxx_defaulted_move_initializers

cxx_delegating_constructors

cxx_deleted_functions

cxx_enum_forward_declarations

cxx_explicit_conversions

cxx_extended_friend_declarations

cxx_extern_templates

cxx_final

cxx_func_identifier

cxx_generalized_initializers

cxx_inheriting_constructors

cxx_inline_namespaces

cxx_lambdas

cxx_local_type_template_args

cxx_long_long_type

cxx_noexcept

cxx_nonstatic_member_init

cxx_nullptr

cxx_override

cxx_range_for

cxx_raw_string_literals

cxx_reference_qualified_functions

cxx_right_angle_brackets

cxx_rvalue_references

cxx_sizeof_member

cxx_static_assert

cxx_strong_enums

cxx_thread_local

cxx_trailing_return_types

cxx_unicode_literals

cxx_uniform_initialization

cxx_unrestricted_unions

cxx_user_literals

cxx_variadic_macros

cxx_variadic_templates

cxx_std_14

cxx_aggregate_default_initializers

cxx_attribute_deprecated

cxx_binary_literals

cxx_contextual_conversions

cxx_decltype_auto

cxx_digit_separators

cxx_generic_lambdas

cxx_lambda_init_captures

cxx_relaxed_constexpr

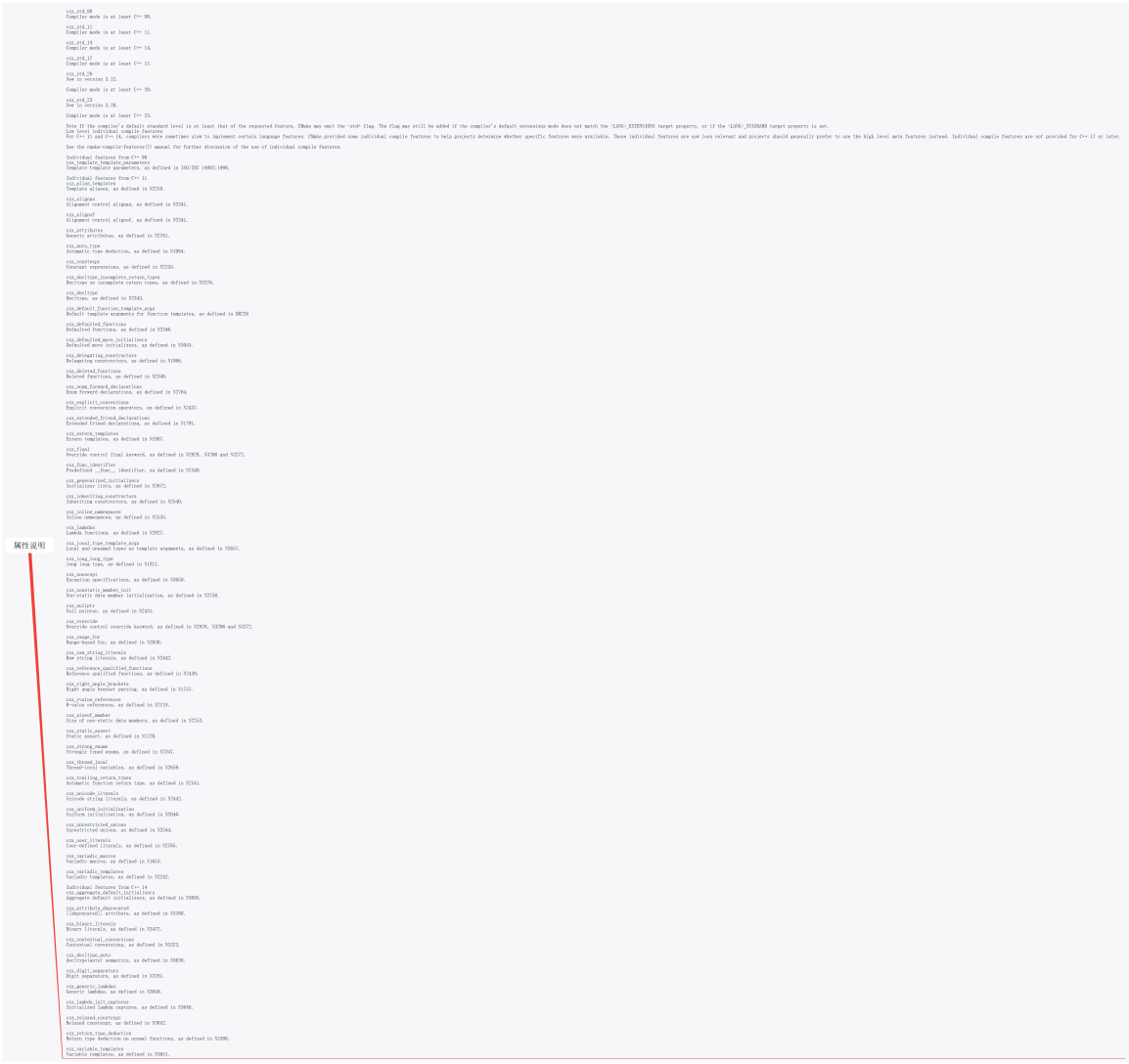
cxx_return_type_deduction

cxx_variable_templates

cxx_std_17

cxx_std_20

1.4.5. 属性说明



cxx_std_98

Compiler mode is at least C++ 98.

cxx_std_11

Compiler mode is at least C++ 11.

cxx_std_14

Compiler mode is at least C++ 14.

cxx_std_17
Compiler mode is at least C++ 17.

cxx_std_20
New in version 3.12.

Compiler mode is at least C++ 20.

cxx_std_23
New in version 3.20.

Compiler mode is at least C++ 23.

Note If the compiler's default standard level is at least that of the requested feature, CMake may omit the -std= flag. The flag may still be added if the compiler's default extensions mode does not match the <LANG>_EXTENSIONS target property, or if the <LANG>_STANDARD target property is set.
Low level individual compile features
For C++ 11 and C++ 14, compilers were sometimes slow to implement certain language features. CMake provided some individual compile features to help projects determine whether specific features were available. These individual features are now less relevant and projects should generally prefer to use the high level meta features instead. Individual compile features are not provided for C++ 17 or later.

See the cmake-compile-features(7) manual for further discussion of the use of individual compile features.

Individual features from C++ 98
cxx_template_template_parameters
Template template parameters, as defined in ISO/IEC 14882:1998.

Individual features from C++ 11

cxx_alias_templates

Template aliases, as defined in N2258.

cxx_alignas

Alignment control alignas, as defined in N2341.

cxx_alignof

Alignment control alignof, as defined in N2341.

cxx_attributes

Generic attributes, as defined in N2761.

cxx_auto_type

Automatic type deduction, as defined in N1984.

cxx_constexpr

Constant expressions, as defined in N2235.

cxx_decltype_incomplete_return_types

Decltype on incomplete return types, as defined in N3276.

cxx_decltype

Decltype, as defined in N2343.

cxx_default_function_template_args

Default template arguments for function templates, as defined in DR226

cxx_defaulted_functions

Defaulted functions, as defined in N2346.

cxx_defaulted_move_initializers

Defaulted move initializers, as defined in N3053.

cxx_delegating_constructors

Delegating constructors, as defined in N1986.

cxx_deleted_functions

Deleted functions, as defined in N2346.

cxx_enum_forward_declarations

Enum forward declarations, as defined in N2764.

cxx_explicit_conversions

Explicit conversion operators, as defined in N2437.

cxx_extended_friend_declarations

Extended friend declarations, as defined in N1791.

cxx_extern_templates

Extern templates, as defined in N1987.

cxx_final

Override control final keyword, as defined in N2928, N3206 and N3272.

cxx_func_identifier

Predefined __func__ identifier, as defined in N2340.

cxx_generalized_initializers

Initializer lists, as defined in N2672.

cxx_inheriting_constructors

Inheriting constructors, as defined in N2540.

cxx_inline_namespaces

Inline namespaces, as defined in N2535.

cxx_lambdas

Lambda functions, as defined in N2927.

cxx_local_type_template_args

Local and unnamed types as template arguments, as defined in N2657.

cxx_long_long_type

long long type, as defined in N1811.

cxx_noexcept

Exception specifications, as defined in N3050.

cxx_nonstatic_member_init

Non-static data member initialization, as defined in N2756.

cxx_nullptr

Null pointer, as defined in N2431.

cxx_override

Override control override keyword, as defined in N2928, N3206 and N3272.

cxx_range_for

Range-based for, as defined in N2930.

cxx_raw_string_literals

Raw string literals, as defined in N2442.

cxx_reference_qualified_functions

Reference qualified functions, as defined in N2439.

cxx_right_angle_brackets

Right angle bracket parsing, as defined in N1757.

cxx_rvalue_references

R-value references, as defined in N2118.

cxx_sizeof_member

Size of non-static data members, as defined in N2253.

cxx_static_assert

Static assert, as defined in N1720.

cxx_strong_enums

Strongly typed enums, as defined in N2347.

cxx_thread_local

Thread-local variables, as defined in N2659.

cxx_trailing_return_types

Automatic function return type, as defined in N2541.

cxx_unicode_literals

Unicode string literals, as defined in N2442.

cxx_uniform_initialization

Uniform initialization, as defined in N2640.

cxx_unrestricted_unions

Unrestricted unions, as defined in N2544.

cxx_user_literals

User-defined literals, as defined in N2765.

cxx_variadic_macros

Variadic macros, as defined in N1653.

cxx_variadic_templates

Variadic templates, as defined in N2242.

Individual features from C++ 14

cxx_aggregate_default_initializers

Aggregate default initializers, as defined in N3605.

cxx_attribute_deprecated

[[deprecated]] attribute, as defined in N3760.

cxx_binary_literals

Binary literals, as defined in N3472.

cxx_contextual_conversions

Contextual conversions, as defined in N3323.

cxx_decltype_auto

decltype(auto) semantics, as defined in N3638.

cxx_digit_separators

Digit separators, as defined in N3781.

cxx_generic_lambdas

Generic lambdas, as defined in N3649.

cxx_lambda_init_captures

Initialized lambda captures, as defined in N3648.

cxx_relaxed_constexpr

Relaxed constexpr, as defined in N3652.

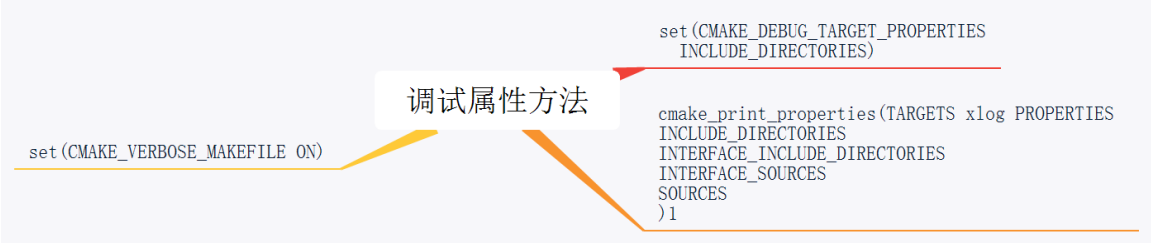
cxx_return_type_deduction

Return type deduction on normal functions, as defined in N3386.

cxx_variable_templates

Variable templates, as defined in N3651.

1.5. 调试属性方法



1.5.1. set(CMAKE_DEBUG_TARGET_PROPERTIES

INCLUDE_DIRECTORIES)

1.5.2. cmake_print_properties(TARGETS xlog PROPERTIES

INCLUDE_DIRECTORIES

INTERFACE_INCLUDE_DIRECTORIES

INTERFACE_SOURCES

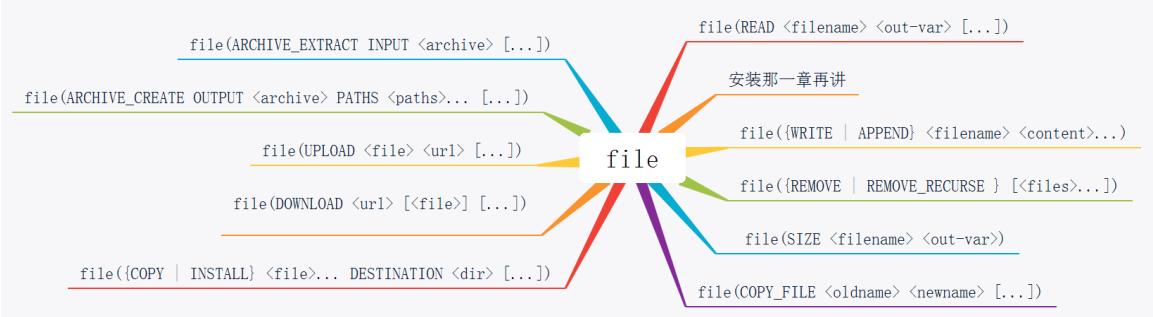
SOURCES

)1

1.5.3. set(CMAKE_VERBOSE_MAKEFILE ON)

看生成的g++、cl语句

1.6. file



1.6.1. file(READ <filename> <out-var> [...])

1.6.2. 安装那一章再讲

1.6.3. file({WRITE | APPEND} <filename> <content>...)

1.6.4. file({REMOVE | REMOVE_RECURSE } [<files>...])

1.6.5. file(SIZE <filename> <out-var>)

1.6.6. file(COPY_FILE <oldname> <newname> [...])

1.6.7. file({COPY | INSTALL} <file>... DESTINATION <dir> [...])

1.6.8. file(DOWNLOAD <url> [<file>] [...])

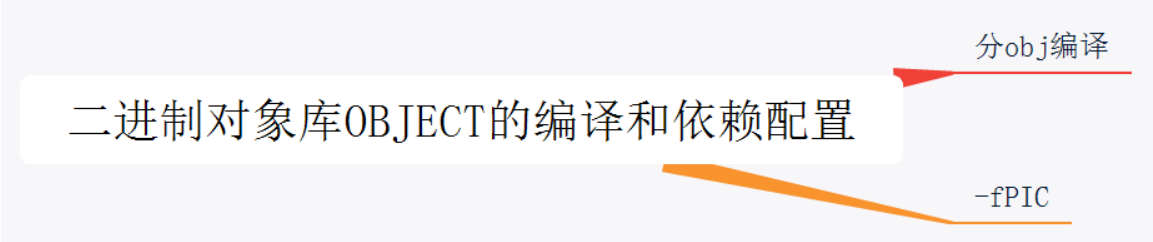
1.6.9. file(UPLOAD <file> <url> [...])

1.6.10. file(ARCHIVE_CREATE OUTPUT <archive> PATHS <paths>... [...])

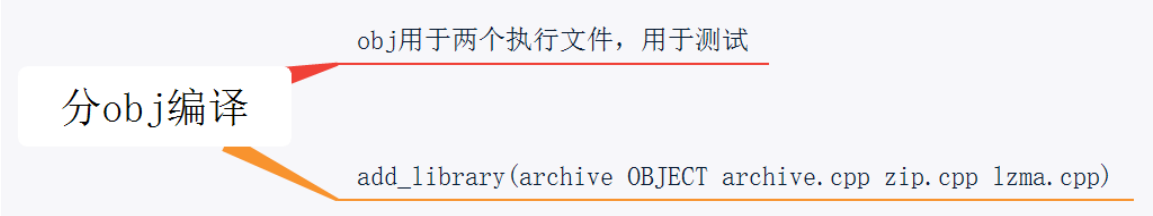
1.6.11. file(ARCHIVE_EXTRACT INPUT <archive> [...])

2. add_library 详细配置

2.1. 二进制对象库OBJECT的编译和依赖配置



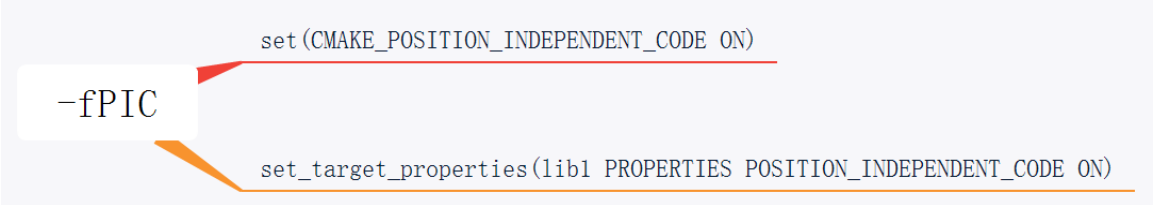
2.1.1. 分obj编译



obj用于两个执行文件，用于测试

add_library(archive OBJECT archive.cpp zip.cpp lzma.cpp)

2.1.2. -fPIC

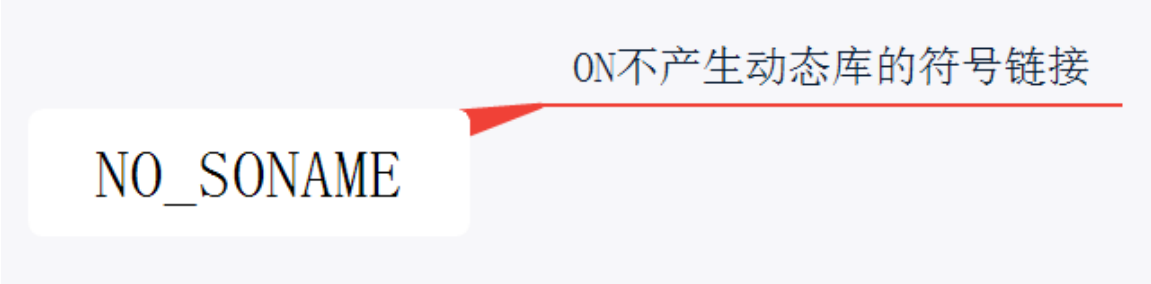


set(CMAKE_POSITION_INDEPENDENT_CODE ON)

set_target_properties(lib1 PROPERTIES POSITION_INDEPENDENT_CODE ON)

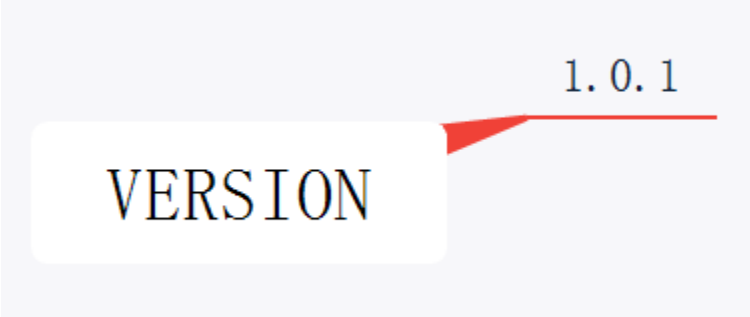
2.2. 带版本号的库符号链接

2.2.1. NO_SONAME



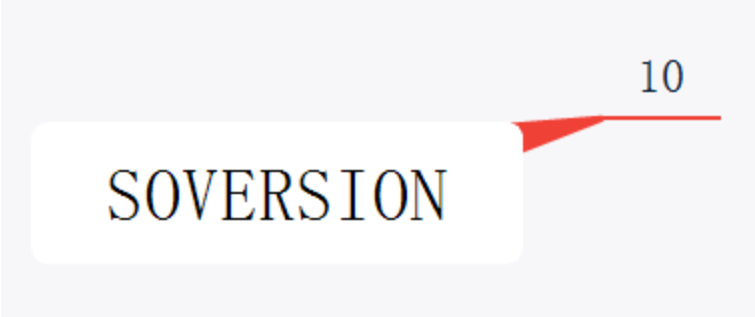
ON不产生动态库的符号链接

2.2.2. VERSION



1.0.1

2.2.3. SOVERSION



10

2.2.4. set_target_properties(A PROPERTIES

VERSION "1.0.1"

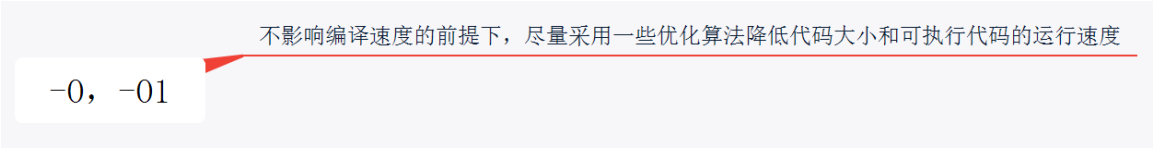
SOVERSION "10"

)

3. CMake Debug Release配置

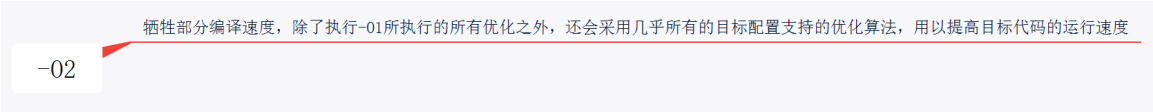
3.1. Debug/Release Mode

3.1.1. -O，-O1



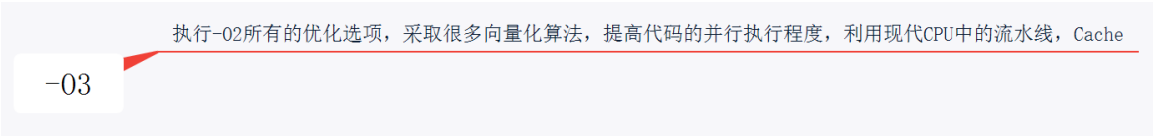
不影响编译速度的前提下，尽量采用一些优化算法降低代码大小和可执行代码的运行速度

3.1.2. -O2



牺牲部分编译速度，除了执行-O1所执行的所有优化之外，还会采用几乎所有的目标配置支持的优化算法，用以提高目标代码的运行速度

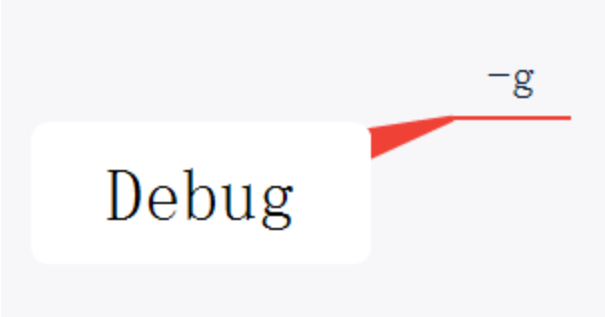
3.1.3. -O3



执行-O2所有的优化选项，采取很多向量化算法，提高代码的并行执行程度，利用现代CPU中的流水线，Cache

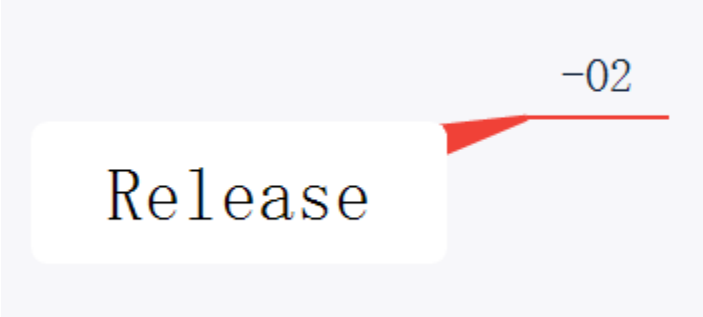
3.2. config对应的优化

3.2.1. Debug



-g

3.2.2. Release

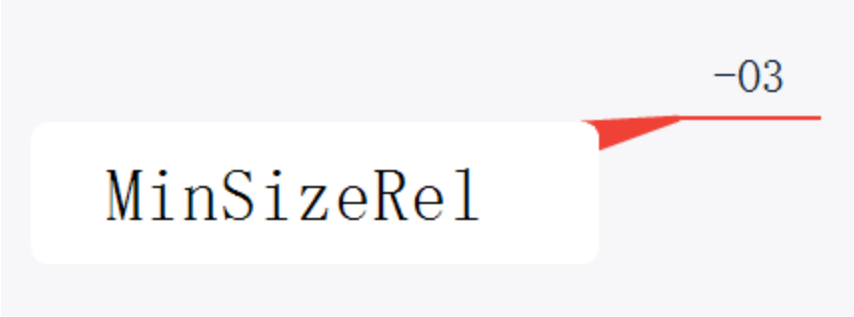


-O2

3.2.3. RelWithDebInfo

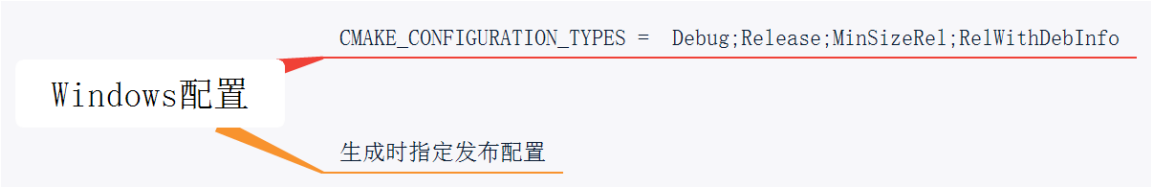
-O2 -g

3.2.4. MinSizeRel



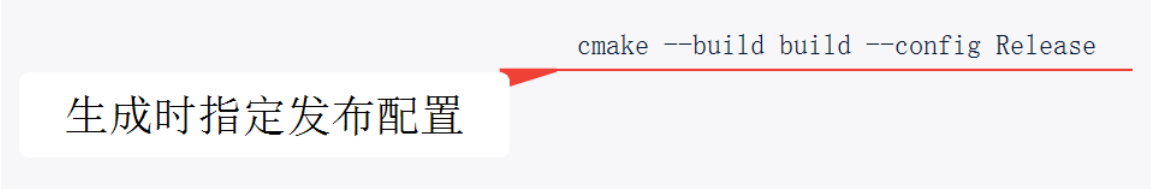
-O3

3.3. Windows配置



3.3.1. CMAKE_CONFIGURATION_TYPES = Debug;Release;MinSizeRel;RelWithDebInfo

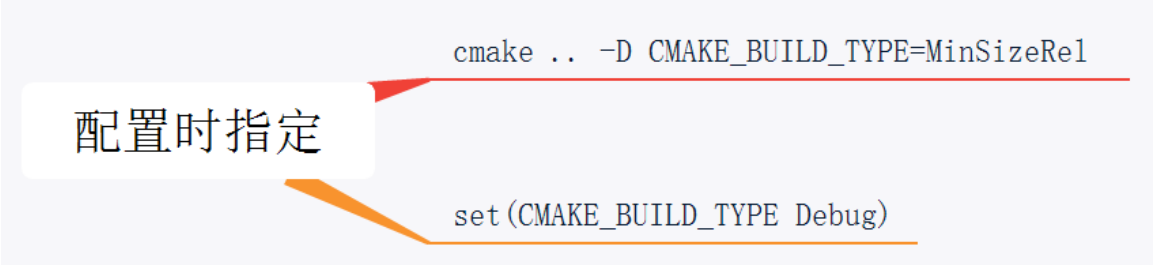
3.3.2. 生成时指定发布配置



cmake --build build --config Release

3.4. Linux配置

3.4.1. 配置时指定



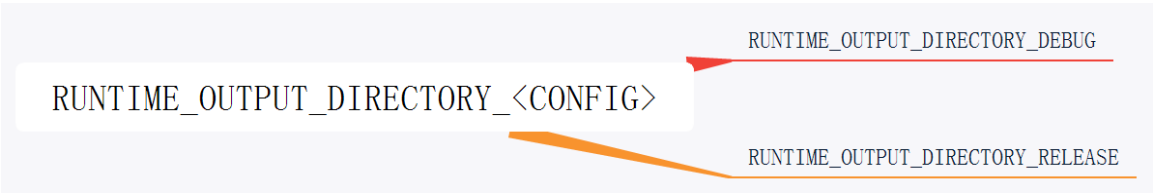
cmake .. -D CMAKE_BUILD_TYPE=MinSizeRel

set(CMAKE_BUILD_TYPE Debug)

3.5. 配置Debug Release不同输出路径

3.5.1. 执行程序和dll输出

RUNTIME_OUTPUT_DIRECTORY_<CONFIG>



RUNTIME_OUTPUT_DIRECTORY_DEBUG

RUNTIME_OUTPUT_DIRECTORY_RELEASE

3.5.2. lib和.a库输出

ARCHIVE_OUTPUT_DIRECTORY_<CONFIG>

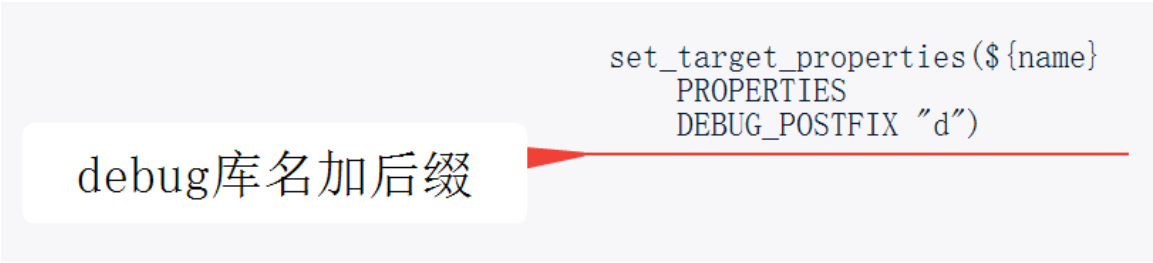
3.5.3. .so动态库输出

LIBRARY_OUTPUT_DIRECTORY_<CONFIG>

3.5.4. pdb文件输出

PDB_OUTPUT_DIRECTORY_<CONFIG>

3.6. debug库名加后缀

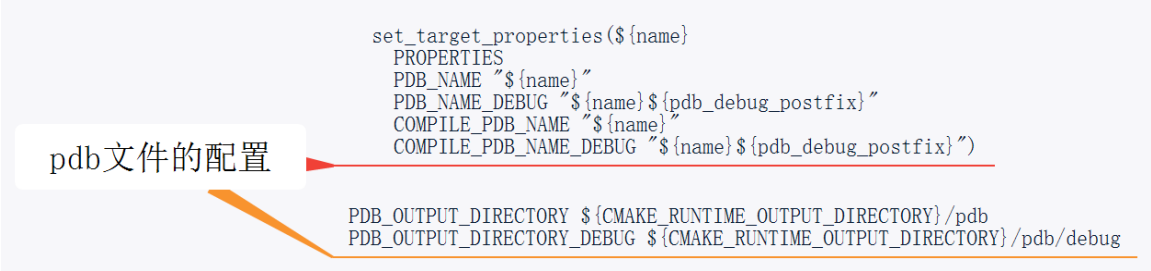


3.6.1. set_target_properties(${name}

PROPERTIES

DEBUG_POSTFIX "d")

3.7. pdb文件的配置



3.7.1. set_target_properties(${name}

PROPERTIES

PDB_NAME "${name}"

PDB_NAME_DEBUG "${name}${pdb_debug_postfix}"

COMPILE_PDB_NAME "${name}"

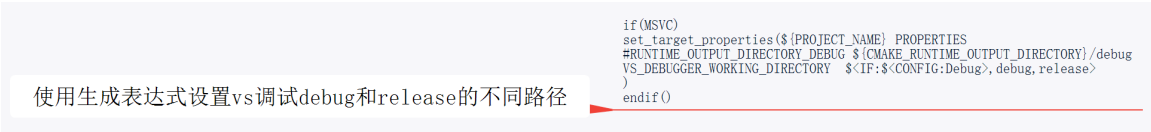
COMPILE_PDB_NAME_DEBUG "${name}${pdb_debug_postfix}")

3.7.2. PDB_OUTPUT_DIRECTORY ${CMAKE_RUNTIME_OUTPUT_DIRECTORY}/pdb

PDB_OUTPUT_DIRECTORY_DEBUG

${CMAKE_RUNTIME_OUTPUT_DIRECTORY}/pdb/debug

3.8. 使用生成表达式设置vs调试debug和release的不同路径



3.8.1. if(MSVC)

set_target_properties(${PROJECT_NAME} PROPERTIES

#RUNTIME_OUTPUT_DIRECTORY_DEBUG

${CMAKE_RUNTIME_OUTPUT_DIRECTORY}/debug

VS_DEBUGGER_WORKING_DIRECTORY $<IF:$<CONFIG:Debug>,debug,release>

)

endif()

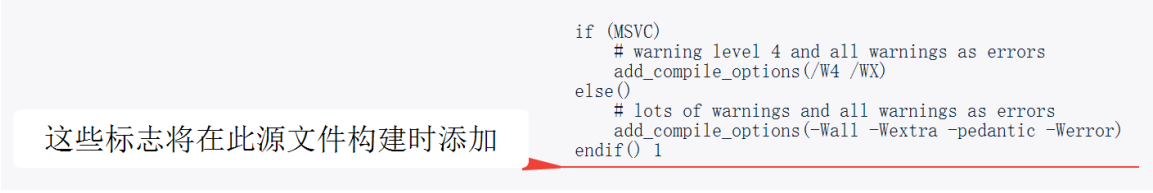
4. VS编译特性添加

4.1. target_compile_options编译参数

4.1.1. target_compile_options(myexe PRIVATE /bigobj)

这些标志将在此源文件构建时添加

4.1.2. 这些标志将在此源文件构建时添加



if (MSVC)

warning level 4 and all warnings as errors

add_compile_options(/W4 /WX)

else()

lots of warnings and all warnings as errors

add_compile_options(-Wall -Wextra -pedantic -Werror)

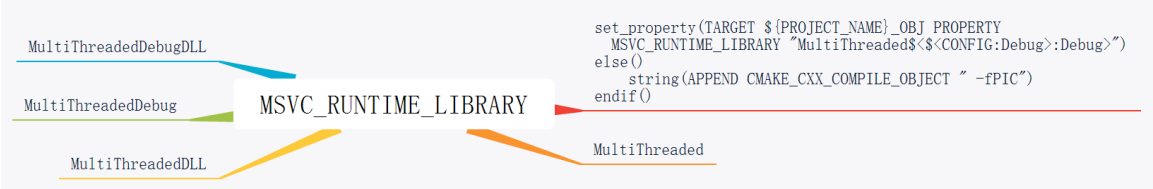
endif() 1

4.1.3. COMPILE_OPTIONS

4.1.4. INTERFACE_COMPILE_OPTIONS

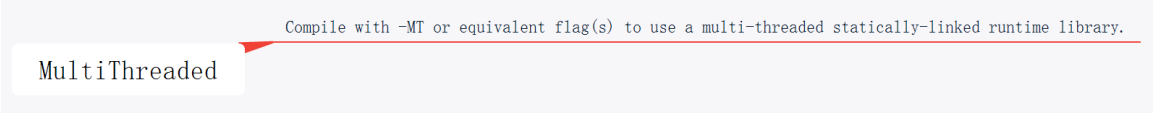
4.2. 调试、MD

4.2.1. MSVC_RUNTIME_LIBRARY



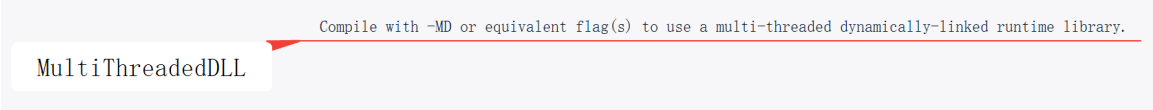
set_property(TARGET ${PROJECT_NAME}_OBJ PROPERTY
MSVC_RUNTIME_LIBRARY "MultiThreaded$<$<CONFIG:Debug>:Debug>")
else()
string(APPEND CMAKE_CXX_COMPILE_OBJECT " -fPIC")
endif()

MultiThreaded



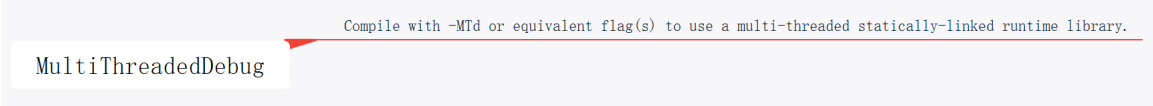
Compile with -MT or equivalent flag(s) to use a multi-threaded statically-linked runtime library.

MultiThreadedDLL



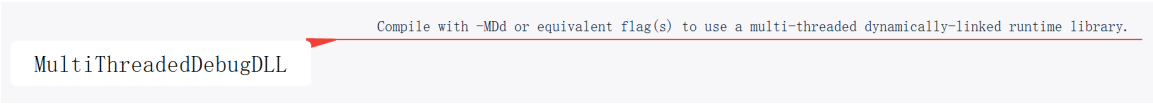
Compile with -MD or equivalent flag(s) to use a multi-threaded dynamically-linked runtime library.

MultiThreadedDebug



Compile with -MTd or equivalent flag(s) to use a multi-threaded statically-linked runtime library.

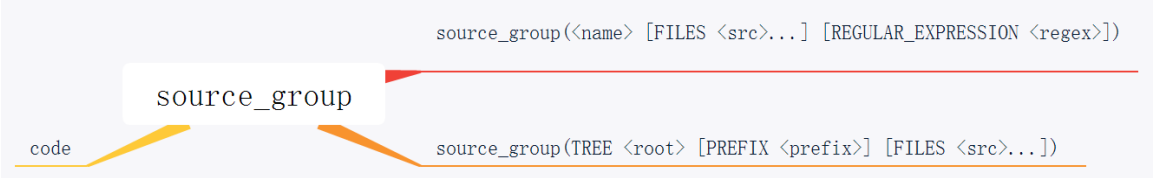
MultiThreadedDebugDLL



Compile with -MDd or equivalent flag(s) to use a multi-threaded dynamically-linked runtime library.

4.3. vs分组

4.3.1. source_group

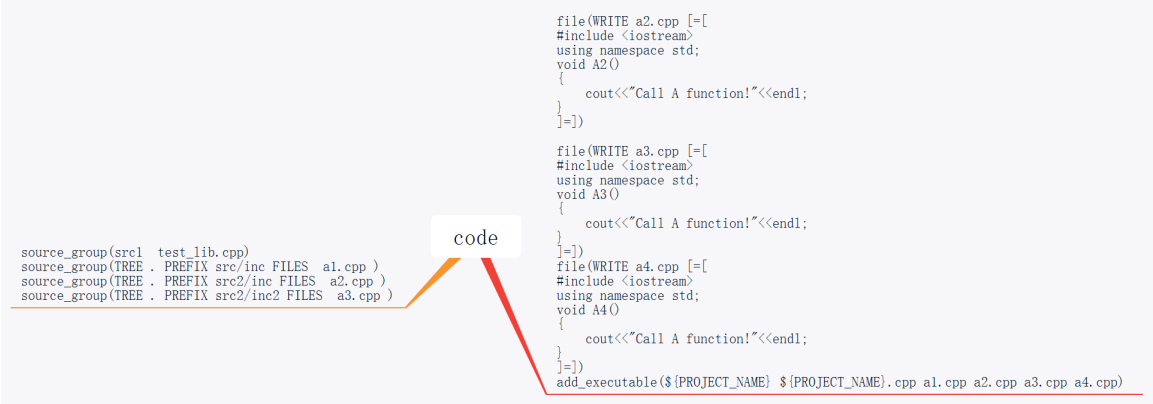


source_group(<name> [FILES <src>...] [REGULAR_EXPRESSION <regex>])

source_group(TREE <root> [PREFIX <prefix>] [FILES <src>...])

root 后面的src路径会去掉root的内容，显示剩下的路径

code



file(WRITE a2.cpp [=[

#include <iostream>

using namespace std;

void A2()

{

cout<<"Call A function!"<<endl;

}

]=])

file(WRITE a3.cpp [=[

#include <iostream>

```
using namespace std;
void A3()
{
   cout<<"Call A function!"<<endl;
}
]=])
file(WRITE a4.cpp [=[
#include <iostream>
using namespace std;
void A4()
{
   cout<<"Call A function!"<<endl;
}
]=])
add_executable(${PROJECT_NAME} ${PROJECT_NAME}.cpp a1.cpp a2.cpp
a3.cpp a4.cpp)

source_group(src1  test_lib.cpp)
source_group(TREE . PREFIX src/inc FILES  a1.cpp )
source_group(TREE . PREFIX src2/inc FILES  a2.cpp )
source_group(TREE . PREFIX src2/inc2 FILES  a3.cpp )
```