第三章 CMake主要语法 cmake-language(7)

第三章 CMake主要语法 cmake-language(7) 1
1. 3.1. if 控制流程 7
1.1. 语法格式..7
1.1.1. if(<condition>) <commands> elseif(<condition>) # optional block, can be repeated <commands> else() # optional block <commands> endif() 8
1.2. 基本表达式..8
1.2.1. if(<constant>)..8
1.2.2. if(<variable>) 8
1.2.3. if(<string>)..9
1.3. 逻辑操作符..9
1.3.1. NOT AND OR 9
1.4. if 判断语句..10
1.4.1. 一元判断 10
1.4.2. 二元判断 10
1.4.3. 存在性检查 11
1.4.4. 文件操作 12
1.4.5. 比较 14
1.4.6. 版本比较 15
1.5. 遗留问题..16
1.5.1. 判断语句过长 17
1.5.2. 无法嵌入到其他功能函数中 17
2. 3.2. 变量和缓存 17
2.1. 202cmake_cache..17
2.1.1. 202cmake_cache/ ├── CMakeLists.txt ├── sub1 │ └── CMakeLists.txt └── sub2 └── CMakeLists.txt 17
2.2. 缓存变量的基础语法和使用..18
2.2.1. set(<variable> <value>... CACHE <type> <docstring> [FORCE]) 18
2.3. 缓存变量对应cmake-gui和ccmake..20
2.3.1. cmake-gui..20
2.3.2. ccmake 22
2.3.3. 分类型展示 23
2.3.4. option(<variable> "<help_text>" [value])..23
2.4. CMake CACHE覆盖策略设置 23
2.4.1. CMP0126 3.21 版中的新功能。
当此政策设置为NEW时，set(CACHE)命令不会从当前范围中删除任何同名的普通变量。在以下情况下，该OLD行为会从当前作用域中删除任何同名的普通变量： 24

2.4.2. cmake_policy(SET CMP0126 NEW) 24
2.4.3. $CACHE{NVAR1} 25
2.5. -D 传递缓存变量 25
2.5.1. cmake -S . -B build -D PARA1=para001 25
2.6. CMake内置缓存变量 25
2.6.1. BUILD_SHARED_LIBS 25
2.6.2. set(BUILD_SHARED_LIBS OFF CACHE BOOL "lib") 25
2.6.3. message("BUILD_SHARED_LIBS = ${BUILD_SHARED_LIBS}") 25
3. 3.3. 属性与变量 25
3.1. CMake 变量和属性有什么区别 26
3.1.1. 一种简短的说明是，属性是作用域为目标的变量。 26
3.1.2. global property can be a useful uncached global variable 26
3.2. 属性语法 26
3.2.1. set_property 26
3.2.2. get_property 28
3.2.3. define_property 30
3.3. 属性分类 30
3.3.1. 全局属性 30
3.3.2. 目录属性 32
3.3.3. 文件属性 33
3.3.4. 目标属性 33
3.4. 打印属性 34
3.4.1. include(CMakePrintHelpers) 34
3.4.2. cmake_print_properties 35
3.4.3. cmake_print_variables(var1 var2 .. varN) 35
3.5. CMake预置属性 35
3.5.1. 全局属性 35
3.5.2. 目录属性 39
3.5.3. 目标属性 43
3.5.4. 源码属性 56
4. 3.4. 环境变量 60
4.1. 环境变量语法 60
4.1.1. set(ENV{<variable>} [<value>]) 60
4.1.2. $ENV{<variable>} 60
4.2. 环境变量特性 60
4.2.1. 只影响当前的 CMake 进程，不影响调用 CMake 的进程，也不影响整个系统环境，也不影响后续构建或测试进程的环境。 ...60
4.2.2. 环境变量与全局属性 61

4.2.3. Environment Variables are like ordinary Variables, with the following differences: Scope Environment variables have global scope in a CMake process. They are never cached. 61
4.3. 环境变量类型 61
4.3.1. cmake预置 61
4.3.2. 自定义环境变量 64
4.3.3. 系统变量 64
5. 3.5 cmake math数学运算 64
5.1. math(EXPR <variable> "<expression>" [OUTPUT_FORMAT <format>]) 64
5.2. "5 * (10 + 13)". 支持 +, -, *, /, %, |, &, ^, ~, <<, >> 64
5.3. 结果必须是64位有符号整数 64
5.4. 输出格式 64
5.4.1. HEXADECIMAL 64
5.4.2. DECIMAL 65
6. 3.6 cmake string字符串处理 65
6.1. 语法 66
6.1.1. 搜索和替换 66
6.1.2. 操作 66
6.1.3. 比较 67
6.1.4. 哈希值 68
6.1.5. 生成 68
6.1.6. JSON 68
7. 3.7. list基础语法 69
7.1. set(srcs a.c b.c c.c) # sets "srcs" to "a.c;b.c;c.c" 69
7.2. CMake中存储所有值都是字符串，有”；"间隔符的字符串被拆分为列表 69
7.3. set(x a "b;c") # sets "x" to "a;b;c", not "a;b\;c" 69
7.4. 语法 69
7.4.1. Reading list(LENGTH <list> <out-var>) list(GET <list> <element index> [<index> ...] <out-var>) list(JOIN <list> <glue> <out-var>) list(SUBLIST <list> <begin> <length> <out-var>) Search list(FIND <list> <value> <out-var>) Modification list(APPEND <list> [<element>...]) list(FILTER <list> {INCLUDE | EXCLUDE} REGEX <regex>) list(INSERT <list> <index> [<element>...]) list(POP_BACK <list> [<out-var>...]) list(POP_FRONT <list> [<out-var>...]) list(PREPEND <list> [<element>...]) list(REMOVE_ITEM <list> <value>...) list(REMOVE_AT <list> <index>...) list(REMOVE_DUPLICATES <list>) list(TRANSFORM <list> <ACTION> [...]) Ordering list(REVERSE <list>) list(SORT <list> [...]) 70
7.5. code 71
7.5.1. set(src "a" "b" "c;d") list(APPEND src "e") list(APPEND src "f") list(APPEND src "ca1") list(APPEND src "ca2") list(APPEND src "test") message("src = ${src}") #list(APPEND ENV{PATH} "/code") #message($ENV{PATH}) list(LENGTH src length)

message("src length ${length}") # list(GET <list> <element index> [<element index> ...] <output variable>) list(GET src 1 var) message("src 1 = ${var}") list(GET src 12 var) message("src 12 = ${var}") list(GET src -1 var) message("src -1 = ${var}") list(GET src -2 var) message("src -2 = ${var}") #list(JOIN <list> <glue> <output variable>) #a|b|c|d|e|f list(JOIN src "|" var) message("JOIN = ${var}") list(JOIN src "" var) message("JOIN = ${var}") #list(SUBLIST <list> <begin> <length> <output variable>) list(SUBLIST src 0 3 var) message("SUBLIST = ${var}") #list(FIND <list> <value> <output variable>) #全字匹配 list(FIND src "ca1" var) message("FIND = ${var}") # list(INSERT <list> <element_index> <element> [<element> ...]) list(INSERT src 1 "ff") list(INSERT src 3 "ff") message("src = ${src}") list(POP_BACK src var) # list(POP_BACK <list> [<out-var>...]) message("POP_BACK = ${var}") # list(POP_FRONT <list> [<out-var>...]) list(POP_FRONT src var) message("POP_FRONT = ${var}") message("src = ${src}") # list(SORT <list> [COMPARE <compare>] [CASE <case>] [ORDER <order>]) #[[
使用COMPARE关键字选择排序的比较方法。该<compare>选项应该是以下之一：
STRING：按字母顺序对字符串列表进行排序。COMPARE如果未给出该选项，这是默认行为。
FILE_BASENAME：按文件的基本名称对文件的路径名列表进行排序。
NATURAL：使用自然顺序对字符串列表进行排序（参见strverscmp(3)手册），即将连续数字作为整数进行比较。例如：以下列表10.0 1.1 2.1 8.0 2.0 3.1如果选择了比较，则将 排序为1.1 2.0 2.1 3.1 8.0 10.0 ，与比较将排序为1.1 10.0 2.0 2.1 3.1 8.0。NATURALSTRING
CASE关键字选择区分大小写或不区分大小写的排序模式。该<case>选项应该是以下之一：
SENSITIVE：列表项以区分大小写的方式排序。CASE如果未给出该选项，这是默认行为。
INSENSITIVE：列表项不区分大小写。未指定仅大写/小写不同的项目的顺序。
要控制排序顺序，ORDER可以给出关键字。该<order>选项应该是以下之一：
ASCENDING：按升序对列表进行排序。ORDER这是未给出选项时的默认行为。
DESCENDING：按降序对列表进行排序]] list(SORT src) message("SORT src = ${src}") #[[list(REMOVE_ITEM <list> <value> [<value> ...])]] list(REMOVE_DUPLICATES src) message("REMOVE_DUPLICATES src = ${src}") list(REMOVE_ITEM src f) message("REMOVE_ITEM f src = ${src}") list(REMOVE_AT src 2) message("REMOVE_AT 2 src = ${src}") 71

8. 3.8. CMake foreach 循环语句 75

8.1. 语法 75

8.1.1. foreach(<loop_var> <items>) <commands> endforeach() 75

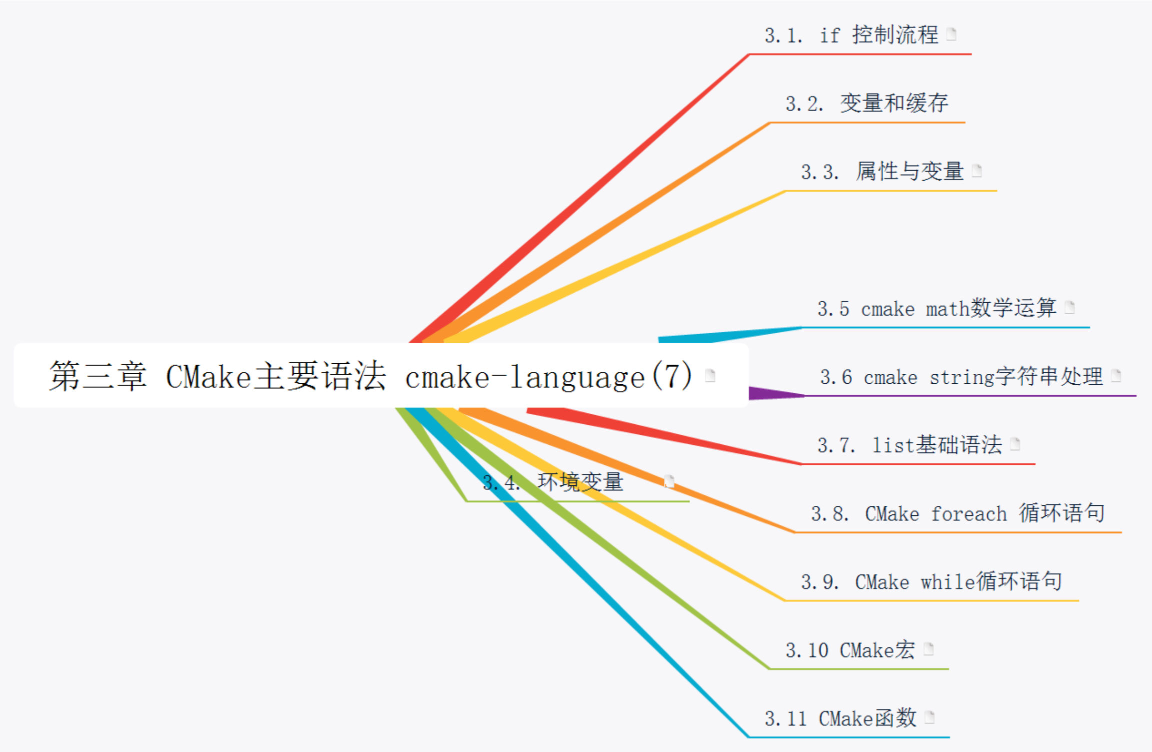
8.2. RANGE 75

8.2.1. foreach(<loop_var> RANGE <stop>) 75

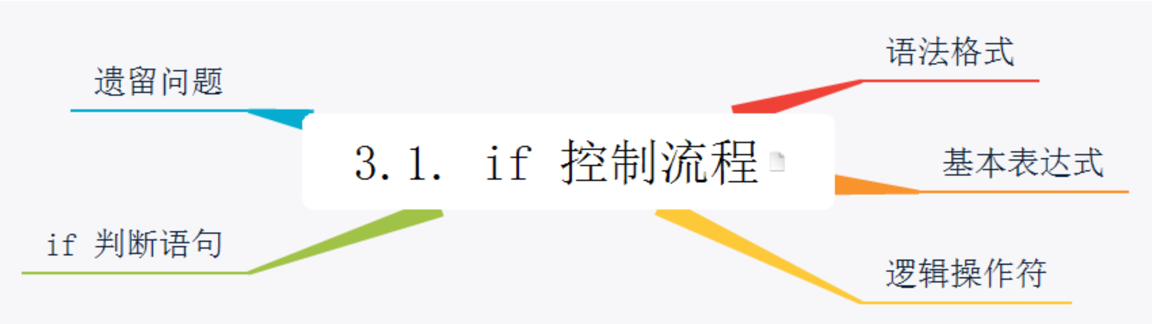
8.2.2. foreach(<loop_var> RANGE <start> <stop> [<step>]) 76

8.3. IN 76
8.3.1. LISTS 76
8.3.2. ITEMS 76
8.3.3. ZIP_LISTS 77
8.4. break() 77
8.4.1. if(var GREATER 50) break() endif() 78
8.5. continue() 78
8.5.1. if(NOT re) message(${var}) continue() endif() 78
8.6. code 78
8.6.1. foreach(var RANGE 100) #string(APPEND out ${var} " ") math(EXPR re "${var} % 3") if(NOT re) message(${var}) continue() endif() if(var GREATER 50) break() endif() message(".") endforeach() message("end for") 79
8.6.2. #[[foreach(<loop_var> <items>) <commands> endforeach()]] #foreach(<loop_var> RANGE <stop>) # var 0 , 1, 2 ,3 ,4 ..10 string(out "") foreach(var RANGE 10) string(APPEND out ${var} " ") message(${var}) endforeach() message("out = ${out}") # foreach(<loop_var> RANGE <start> <stop> [<step>]) foreach(var RANGE 0 10 2) #string(APPEND out ${var} " ") message(${var}) endforeach() # foreach(<loop_var> IN [LISTS [<lists>]] [ITEMS [<items>]]) set(args a b c d e) foreach(var IN LISTS args) message(${var}) endforeach() set(A 0;1) set(B 2 3) set(C "4 5") set(D 6;7 8) set(E "") foreach(X IN LISTS A B C D E) message(STATUS "X=${X}") endforeach() list(APPEND English one two three four) list(APPEND Bahasa satu dua tiga) # 同步遍历两组数组 foreach(num IN ZIP_LISTS English Bahasa) message(STATUS "num_0=${num_0}, num_1=${num_1}") endforeach() foreach(en ba IN ZIP_LISTS English Bahasa) message(STATUS "en=${en}, ba=${ba}") endforeach() 80
9. 3.9. CMake while循环语句 81
9.1. while(<condition>) <commands> endwhile() 82
9.2. code 82
9.2.1. while(var) message(${var}) math(EXPR var "${var}+1") if(var GREATER 100) set(var 0) endif() endwhile() 82
10. 3.10 CMake宏 82
10.1. 基本语法 82
10.1.1. macro(foo) <commands> endmacro() 83
10.1.2. 宏名称大小写不敏感 83
10.2. 普通参数 83
10.2.1. 必需的参数 83
10.2.2. ARGC 84
10.2.3. ARGN 84
10.2.4. ARGV0 ARGV1 ARGV2 84
10.2.5. 参数不是变量 84

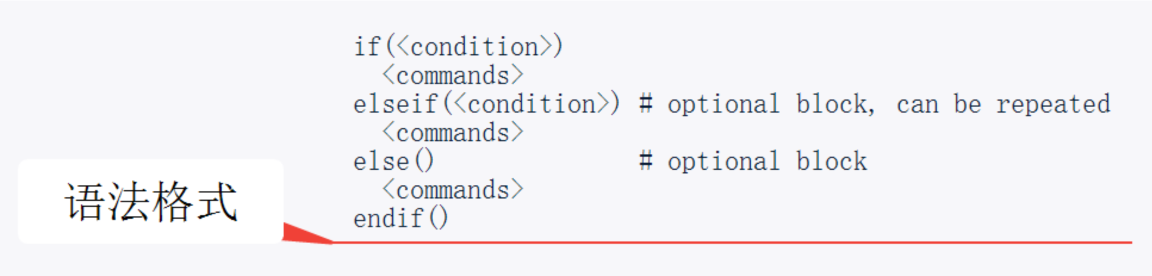
10.2.6. 如果在调用宏的范围内有一个同名的变量，则使用未引用的名称将使用现有变量而不是参数 85
10.3. 属性式参数 85
10.3.1. cmake_parse_arguments....85
10.3.2. my_macro(TARGETS foo bar DESTINATION bin) 89
10.4. code 89
10.4.1. macro(foo) set(foo_var "foovar") #ARGN, ARGC,ARGV等ARGV0不是变量 # 通常宏使用全小写的名称 message(" ${ARGC} ${ARGV} ") message("ARGV0 = ${ARGV0}") message("ARGV1 = ${ARGV1}") message("ARGV2 = ${ARGV2}") message("ARGV3 = ${ARGV3}") message("macro(foo)") message("para1 = ${para1}") foreach(arg IN LISTS ARGN) message("arg = ${arg}") endforeach() endmacro() foo(1) Foo(33) FOO(44 "tt" 111) 90
11. 3.11 CMake函数 91
11.1. 函数的参数是变量 91
11.2. 函数内部设置的普通变量作用域只在函数内 91
11.2.1. set(fun_var2 "fun2 var value" PARENT_SCOPE) 91
11.3. 函数可以用return返回 91
11.3.1. return() 91
11.4. code 92
11.4.1. function(fun arg1 arg2) # 通常函数使用全小写的名称 set(fun_var "fun var value") set(fun_var2 "fun2 var value" PARENT_SCOPE) message("call fun") message(" ${ARGC} ${ARGV} ") message("ARGV0 = ${ARGV0}") message("ARGV1 = ${ARGV1}") message("ARGV2 = ${ARGV2}") message("ARGV3 = ${ARGV3}") endif() 92
11.4.2. set(testm "001") macro(TestM) set(testm "002") endmacro() function(TestF) set(testm "003") endfunction() TestM() TestF() 92



1. 3.1. if 控制流程



1.1. 语法格式

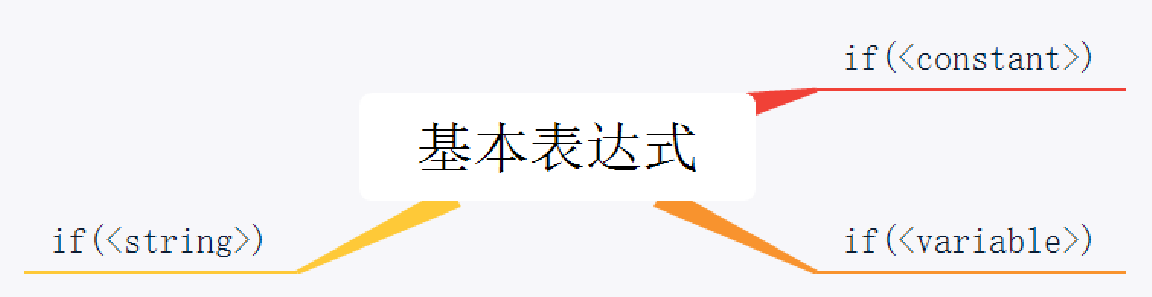


```
if(<condition>)
  <commands>
elseif(<condition>) # optional block, can be repeated
  <commands>
else()              # optional block
  <commands>
endif()
```

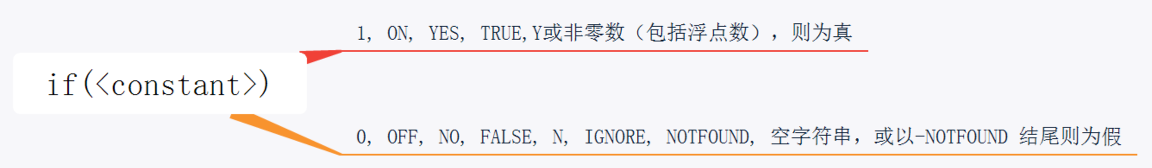
1.1.1. if(<condition>)

```
  <commands>
elseif(<condition>) # optional block, can be repeated
  <commands>
else()              # optional block
  <commands>
endif()
```

1.2. 基本表达式



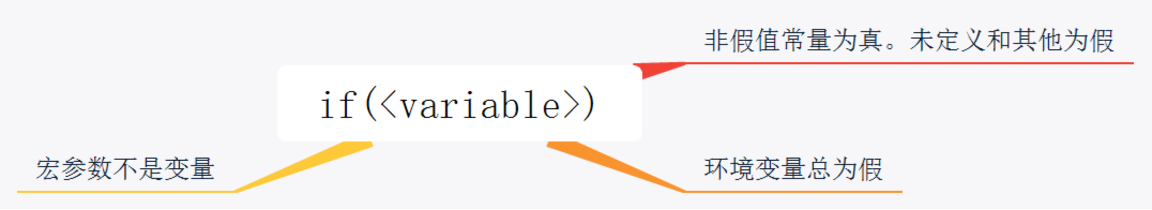
1.2.1. if(<constant>)


if(<constant>)
1, ON, YES, TRUE, Y或非零数（包括浮点数），则为真
0, OFF, NO, FALSE, N, IGNORE, NOTFOUND, 空字符串，或以-NOTFOUND 结尾则为假

1, ON, YES, TRUE,Y或非零数（包括浮点数），则为真

0, OFF, NO, FALSE, N, IGNORE, NOTFOUND, 空字符串，或以-NOTFOUND 结尾则为假

1.2.2. if(<variable>)



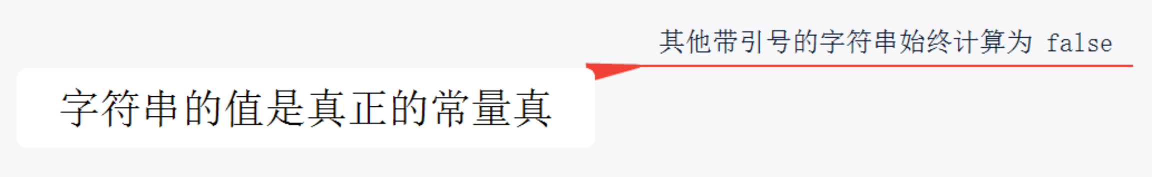
非假值常量为真。未定义和其他为假

环境变量总为假

宏参数不是变量

1.2.3. if(<string>)

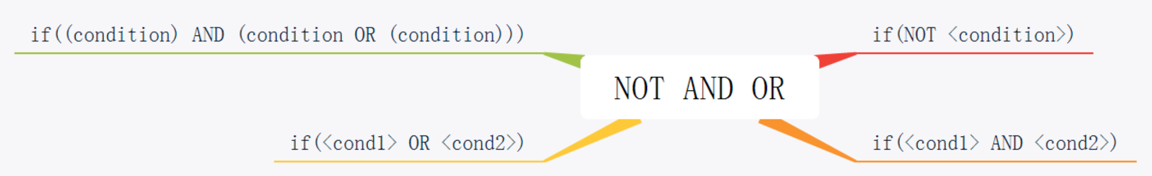
字符串的值是真正的常量真



其他带引号的字符串始终计算为 false

1.3. 逻辑操作符

1.3.1. NOT AND OR



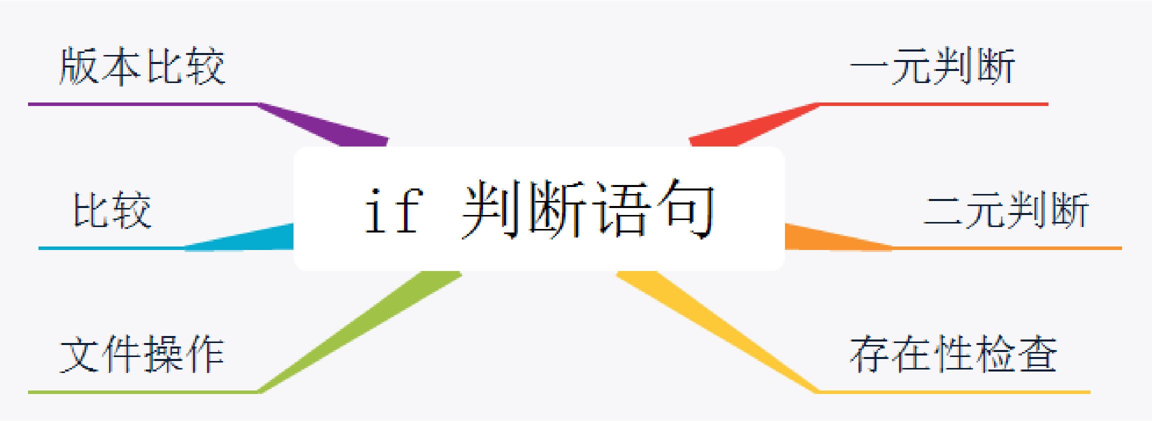
if(NOT <condition>)

if(<cond1> AND <cond2>)

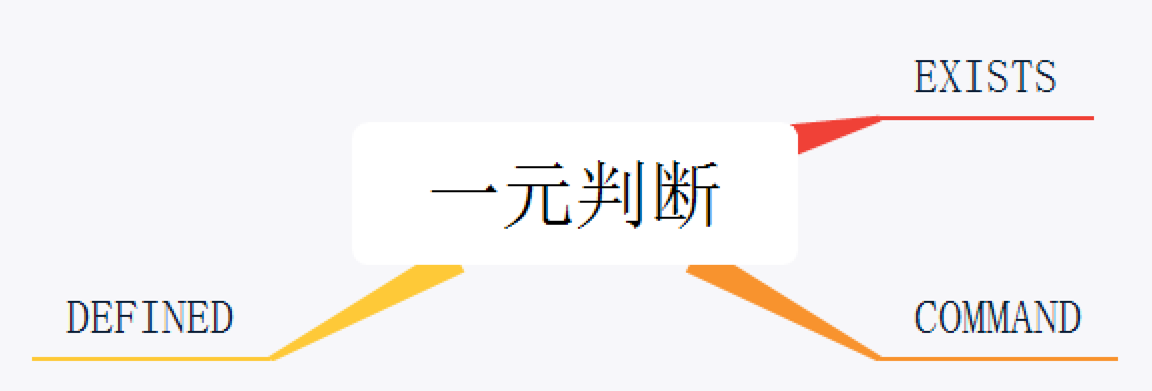
if(<cond1> OR <cond2>)

if((condition) AND (condition OR (condition)))

1.4. if 判断语句



1.4.1. 一元判断

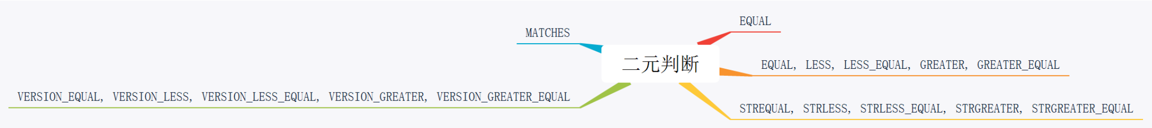


EXISTS

COMMAND

DEFINED

1.4.2. 二元判断


MATCHES
二元判断
EQUAL
EQUAL, LESS, LESS_EQUAL, GREATER, GREATER_EQUAL
VERSION_EQUAL, VERSION_LESS, VERSION_LESS_EQUAL, VERSION_GREATER, VERSION_GREATER_EQUAL
STREQUAL, STRLESS, STRLESS_EQUAL, STRGREATER, STRGREATER_EQUAL

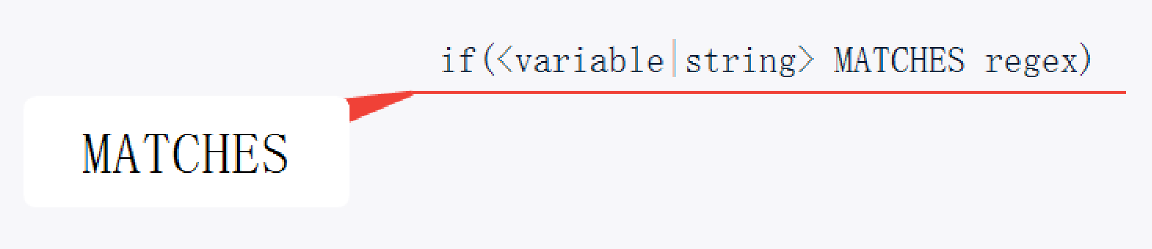
EQUAL

EQUAL, LESS, LESS_EQUAL, GREATER, GREATER_EQUAL

STREQUAL, STRLESS, STRLESS_EQUAL, STRGREATER, STRGREATER_EQUAL

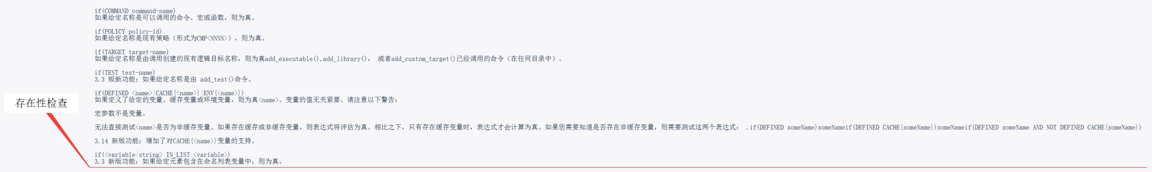
VERSION_EQUAL, VERSION_LESS, VERSION_LESS_EQUAL, VERSION_GREATER, VERSION_GREATER_EQUAL

MATCHES



if(<variable|string> MATCHES regex)

1.4.3. 存在性检查



if(COMMAND command-name)
如果给定名称是可以调用的命令、宏或函数，则为真。

if(POLICY policy-id)
如果给定名称是现有策略（形式为CMP<NNNN>），则为真。

if(TARGET target-name)
如果给定名称是由调用创建的现有逻辑目标名称，则为真add_executable(),add_library()， 或者add_custom_target()已经调用的命令（在任何目录中）。

if(TEST test-name)
3.3 版新功能：如果给定名称是由 add_test()命令。

if(DEFINED <name>|CACHE{<name>}|ENV{<name>})
如果定义了给定的变量、缓存变量或环境变量，则为真<name>。变量的值无关紧要。请注意以下警告：

宏参数不是变量。

无法直接测试<name>是否为非缓存变量。如果存在缓存或非缓存变量，则表达式将评估为真。相比之下，只有存在缓存变量时，表达式才会计算为真。如果您需要知道是否存在非缓存变量，则需要测试这两个表达式： .if(DEFINED someName)someNameif(DEFINED CACHE{someName})someNameif(DEFINED someName AND NOT DEFINED CACHE{someName})

3.14 新版功能：增加了对CACHE{<name>}变量的支持。

if(<variable|string> IN_LIST <variable>)
3.3 新版功能：如果给定元素包含在命名列表变量中，则为真。

if(COMMAND command-name)

如果给定名称是可以调用的命令、宏或函数，则为真。

if(POLICY policy-id)

如果给定名称是现有策略（形式为CMP<NNNN>），则为真。

if(TARGET target-name)

如果给定名称是由调用创建的现有逻辑目标名称，则为真**add_executable(),add_library()**， 或者**add_custom_target()**已经调用的命令（在任何目录中）。

if(TEST test-name)

3.3 版新功能：如果给定名称是由 **add_test()**命令。

if(DEFINED <name>|CACHE{<name>}|ENV{<name>})

如果定义了给定的变量、缓存变量或环境变量，则为真**<name>**。变量的值无关紧要。请注意以下警告：

宏参数不是变量。

无法直接测试**<name>**是否为非缓存变量。如果存在缓存或非缓存变量，则表达式将评估为真。相比之下，只有存在缓存变量时，表达式才会计算为真。如果您需要知道是否存在非缓存变量，则需要测试这两个表达式：

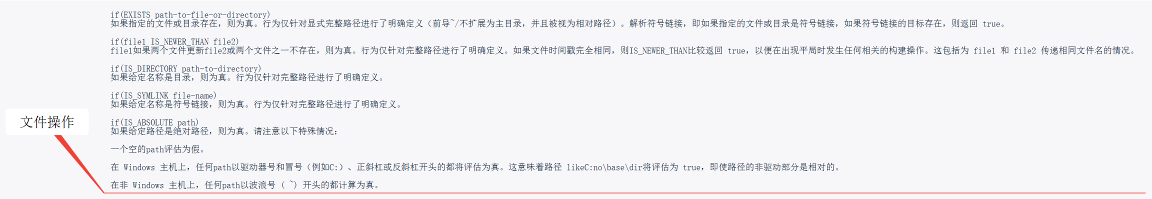
.if(DEFINED someName)someNameif(DEFINED CACHE{someName})someNameif(DEFINED someName AND NOT DEFINED CACHE{someName})

3.14 新版功能：增加了对**CACHE{<name>}**变量的支持。

if(<variable|string> IN_LIST <variable>)

3.3 新版功能：如果给定元素包含在命名列表变量中，则为真。

1.4.4. 文件操作



if(EXISTS path-to-file-or-directory)
如果指定的文件或目录存在，则为真。行为仅针对显式完整路径进行了明确定义（前导~/不扩展为主目录，并且被视为相对路径）。解析符号链接，即如果指定的文件或目录是符号链接，如果符号链接的目标存在，则返回 true。

if(file1 IS_NEWER_THAN file2)
file1如果两个文件更新file2或两个文件之一不存在，则为真。行为仅针对完整路径进行了明确定义。如果文件时间戳完全相同，则IS_NEWER_THAN比较返回 true，以便在出现平局时发生任何相关的构建操作。这包括为 file1 和 file2 传递相同文件名的情况。

if(IS_DIRECTORY path-to-directory)
如果给定名称是目录，则为真。行为仅针对完整路径进行了明确定义。

if(IS_SYMLINK file-name)
如果给定名称是符号链接，则为真。行为仅针对完整路径进行了明确定义。

if(IS_ABSOLUTE path)
如果给定路径是绝对路径，则为真。请注意以下特殊情况：

一个空的path评估为假。

在 Windows 主机上，任何path以驱动器号和冒号（例如C:）、正斜杠或反斜杠开头的都将评估为真。这意味着路径 likeC:no\base\dir将评估为 true，即使路径的非驱动部分是相对的。

在非 Windows 主机上，任何path以波浪号（ ~）开头的都计算为真。

if(EXISTS path-to-file-or-directory)

如果指定的文件或目录存在，则为真。行为仅针对显式完整路径进行了明确定义（前导~/不扩展为主目录，并且被视为相对路径）。解析符号链接，即如果指定的文件或目录是符号链接，如果符号链接的目标存在，则返回 **true**。

if(file1 IS_NEWER_THAN file2)

file1如果两个文件更新**file2**或两个文件之一不存在，则为真。行为仅针对完整路径进行了明确定义。如果文件时间戳完全相同，则**IS_NEWER_THAN**比较返回 **true**，以便在出现平局时发生任何相关的构建操作。这包括为 **file1** 和 **file2** 传递相同文件名的情况。

if(IS_DIRECTORY path-to-directory)

如果给定名称是目录，则为真。行为仅针对完整路径进行了明确定义。

if(IS_SYMLINK file-name)

如果给定名称是符号链接，则为真。行为仅针对完整路径进行了明确定义。

if(IS_ABSOLUTE path)

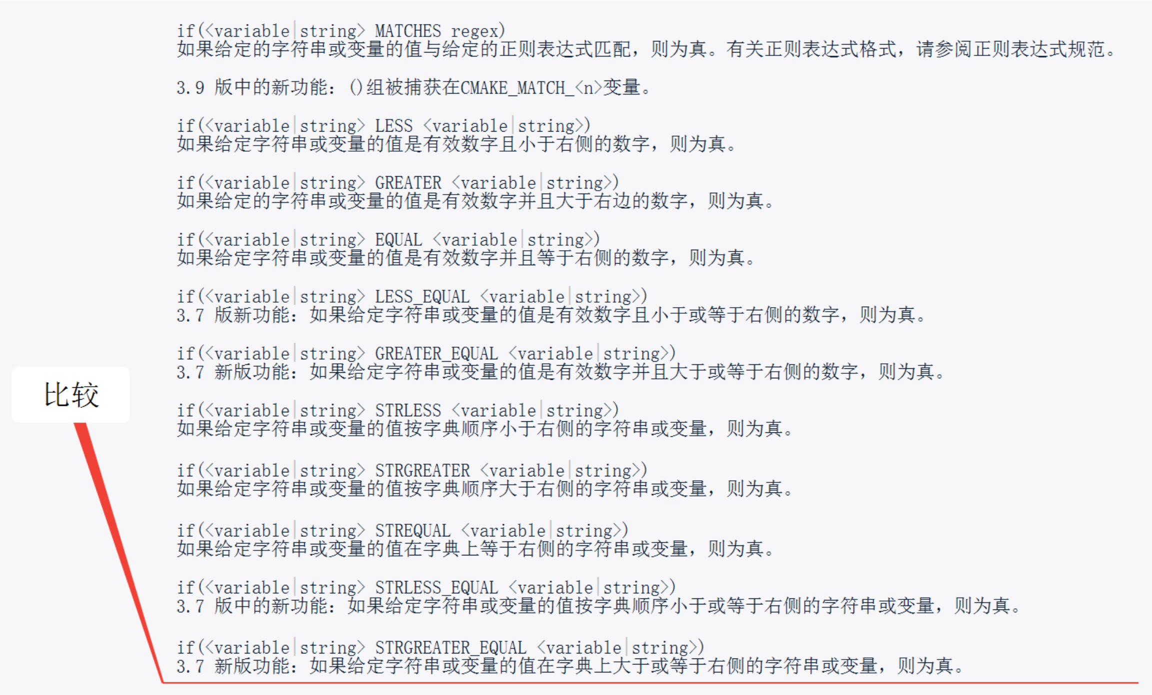
如果给定路径是绝对路径，则为真。请注意以下特殊情况：

一个空的**path**评估为假。

在 **Windows**
主机上，任何**path**以驱动器号和冒号（例如**C:**）、正斜杠或反斜杠开头的都将评估为真。这意味着路径 **likeC:no\base\dir**将评估为
true，即使路径的非驱动部分是相对的。

在非 **Windows** 主机上，任何**path**以波浪号 **(~)** 开头的都计算为真。

1.4.5. 比较



if(<variable|string> MATCHES regex)
如果给定的字符串或变量的值与给定的正则表达式匹配，则为真。有关正则表达式格式，请参阅正则表达式规范。

3.9 版中的新功能：()组被捕获在CMAKE_MATCH_<n>变量。

if(<variable|string> LESS <variable|string>)
如果给定字符串或变量的值是有效数字且小于右侧的数字，则为真。

if(<variable|string> GREATER <variable|string>)
如果给定的字符串或变量的值是有效数字并且大于右边的数字，则为真。

if(<variable|string> EQUAL <variable|string>)
如果给定字符串或变量的值是有效数字并且等于右侧的数字，则为真。

if(<variable|string> LESS_EQUAL <variable|string>)
3.7 版新功能：如果给定字符串或变量的值是有效数字且小于或等于右侧的数字，则为真。

if(<variable|string> GREATER_EQUAL <variable|string>)
3.7 新版功能：如果给定字符串或变量的值是有效数字并且大于或等于右侧的数字，则为真。

if(<variable|string> STRLESS <variable|string>)
如果给定字符串或变量的值按字典顺序小于右侧的字符串或变量，则为真。

if(<variable|string> STRGREATER <variable|string>)
如果给定字符串或变量的值按字典顺序大于右侧的字符串或变量，则为真。

if(<variable|string> STREQUAL <variable|string>)
如果给定字符串或变量的值在字典上等于右侧的字符串或变量，则为真。

if(<variable|string> STRLESS_EQUAL <variable|string>)
3.7 版中的新功能：如果给定字符串或变量的值按字典顺序小于或等于右侧的字符串或变量，则为真。

if(<variable|string> STRGREATER_EQUAL <variable|string>)
3.7 新版功能：如果给定字符串或变量的值在字典上大于或等于右侧的字符串或变量，则为真。

if(<variable|string> MATCHES regex)

如果给定的字符串或变量的值与给定的正则表达式匹配，则为真。有关正则表达式格式，请参阅正则表达式规范。

3.9 版中的新功能：()组被捕获在CMAKE_MATCH_<n>变量。

if(<variable|string> LESS <variable|string>)

如果给定字符串或变量的值是有效数字且小于右侧的数字，则为真。

if(<variable|string> GREATER <variable|string>)

如果给定的字符串或变量的值是有效数字并且大于右边的数字，则为真。

if(<variable|string> EQUAL <variable|string>)

如果给定字符串或变量的值是有效数字并且等于右侧的数字，则为真。

if(<variable|string> LESS_EQUAL <variable|string>)

3.7

版新功能：如果给定字符串或变量的值是有效数字且小于或等于右侧的数字，则为真。

if(<variable|string> GREATER_EQUAL <variable|string>)

3.7

新版功能：如果给定字符串或变量的值是有效数字并且大于或等于右侧的数字，则为真。

if(<variable|string> STRLESS <variable|string>)

如果给定字符串或变量的值按字典顺序小于右侧的字符串或变量，则为真。

if(<variable|string> STRGREATER <variable|string>)

如果给定字符串或变量的值按字典顺序大于右侧的字符串或变量，则为真。

if(<variable|string> STREQUAL <variable|string>)

如果给定字符串或变量的值在字典上等于右侧的字符串或变量，则为真。

if(<variable|string> STRLESS_EQUAL <variable|string>)

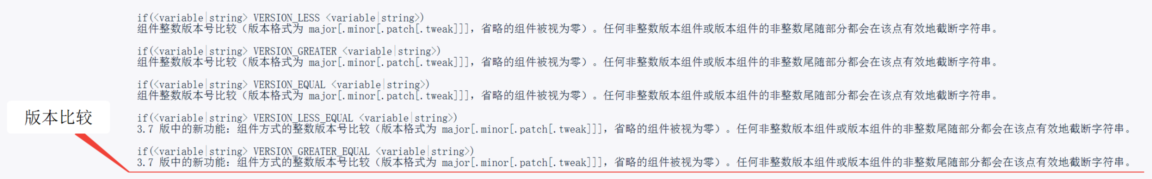
3.7 版中的新功能：如果给定字符串或变量的值按字典顺序小于或等于右侧的字符串或变量，则为真。

if(<variable|string> STRGREATER_EQUAL <variable|string>)

3.7

新版功能：如果给定字符串或变量的值在字典上大于或等于右侧的字符串或变量，则为真。

1.4.6. 版本比较



if(<variable|string> VERSION_LESS <variable|string>)
组件整数版本号比较（版本格式为 major[.minor[.patch[.tweak]]]，省略的组件被视为零）。任何非整数版本组件或版本组件的非整数尾随部分都会在该点有效地截断字符串。

if(<variable|string> VERSION_GREATER <variable|string>)
组件整数版本号比较（版本格式为 major[.minor[.patch[.tweak]]]，省略的组件被视为零）。任何非整数版本组件或版本组件的非整数尾随部分都会在该点有效地截断字符串。

if(<variable|string> VERSION_EQUAL <variable|string>)
组件整数版本号比较（版本格式为 major[.minor[.patch[.tweak]]]，省略的组件被视为零）。任何非整数版本组件或版本组件的非整数尾随部分都会在该点有效地截断字符串。

if(<variable|string> VERSION_LESS_EQUAL <variable|string>)
3.7 版中的新功能：组件方式的整数版本号比较（版本格式为 major[.minor[.patch[.tweak]]]，省略的组件被视为零）。任何非整数版本组件或版本组件的非整数尾随部分都会在该点有效地截断字符串。

if(<variable|string> VERSION_GREATER_EQUAL <variable|string>)
3.7 版中的新功能：组件方式的整数版本号比较（版本格式为 major[.minor[.patch[.tweak]]]，省略的组件被视为零）。任何非整数版本组件或版本组件的非整数尾随部分都会在该点有效地截断字符串。

if(<variable|string> VERSION_LESS <variable|string>)

组件整数版本号比较（版本格式为 **major[.minor[.patch[.tweak]]]**，省略的组件被视为零）。任何非整数版本组件或版本组件的非整数尾随部分都会在该点有效地截断字符串。

if(<variable|string> VERSION_GREATER <variable|string>)

组件整数版本号比较（版本格式为 **major[.minor[.patch[.tweak]]]**，省略的组件被视为零）。任何非整数版本组件或版本组件的非整数尾随部分都会在该点有效地截断字符串。

if(<variable|string> VERSION_EQUAL <variable|string>)

组件整数版本号比较（版本格式为 **major[.minor[.patch[.tweak]]]**，省略的组件被视为零）。任何非整数版本组件或版本组件的非整数尾随部分都会在该点有效地截断字符串。

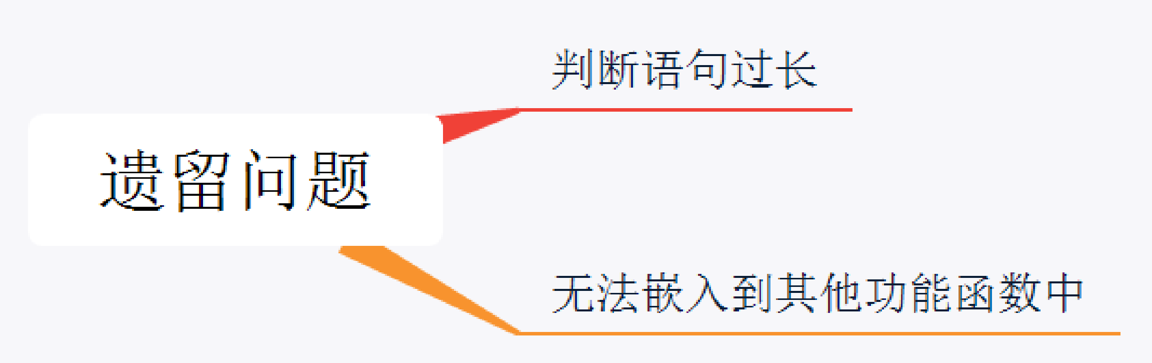
if(<variable|string> VERSION_LESS_EQUAL <variable|string>)

3.7 版中的新功能：组件方式的整数版本号比较（版本格式为 **major[.minor[.patch[.tweak]]]**，省略的组件被视为零）。任何非整数版本组件或版本组件的非整数尾随部分都会在该点有效地截断字符串。

if(<variable|string> VERSION_GREATER_EQUAL <variable|string>)

3.7 版中的新功能：组件方式的整数版本号比较（版本格式为 **major[.minor[.patch[.tweak]]]**，省略的组件被视为零）。任何非整数版本组件或版本组件的非整数尾随部分都会在该点有效地截断字符串。

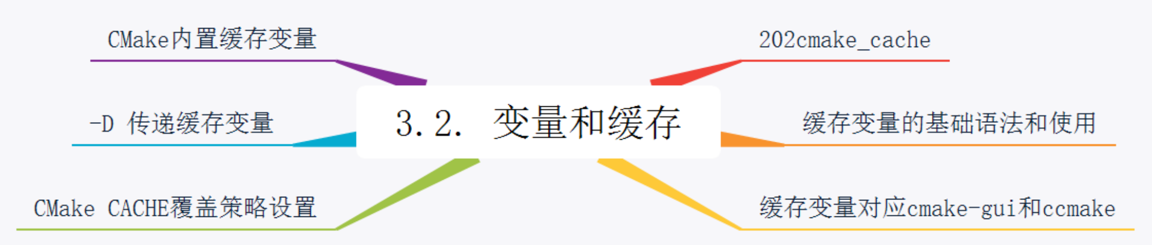
1.5.遗留问题



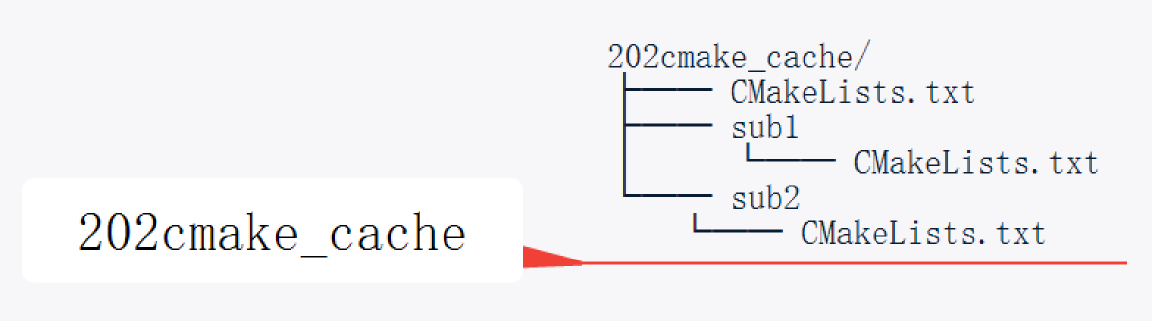
1.5.1. 判断语句过长

1.5.2. 无法嵌入到其他功能函数中

2. 3.2. 变量和缓存



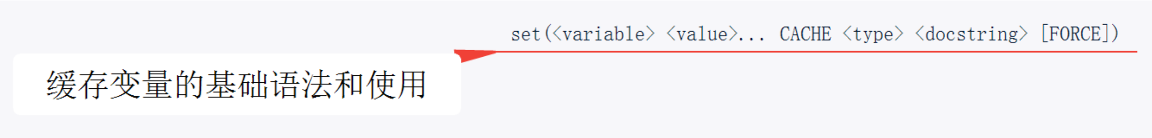
2.1. 202cmake_cache



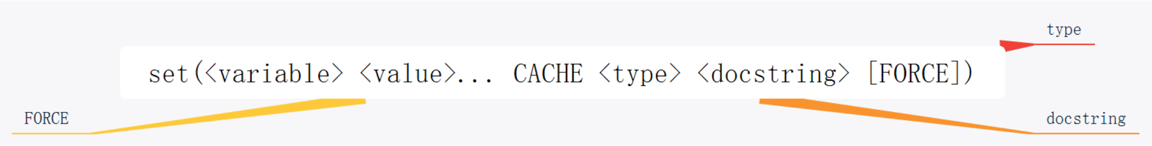
2.1.1. 202cmake_cache/

```
├── CMakeLists.txt
├── sub1
│   └── CMakeLists.txt
└── sub2
    └── CMakeLists.txt
```

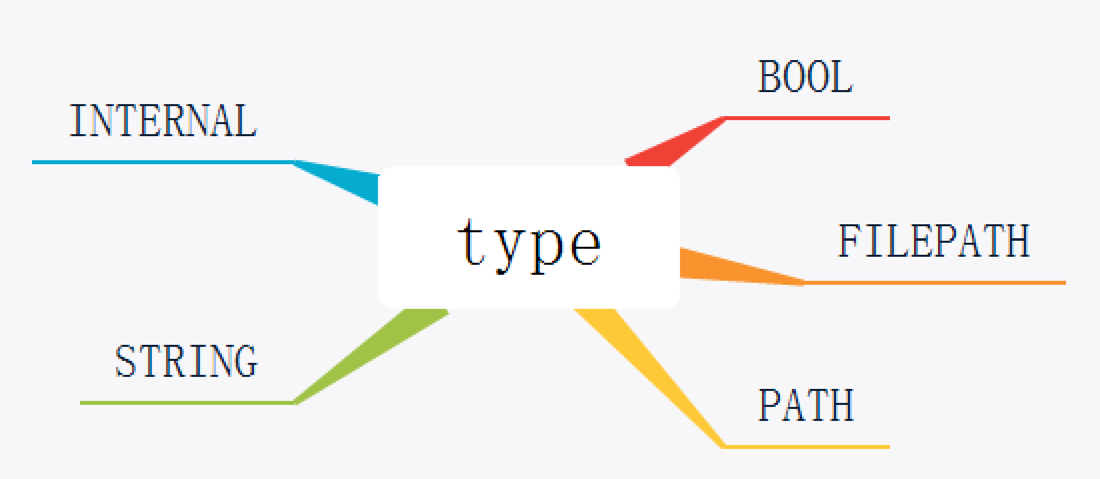
2.2. 缓存变量的基础语法和使用



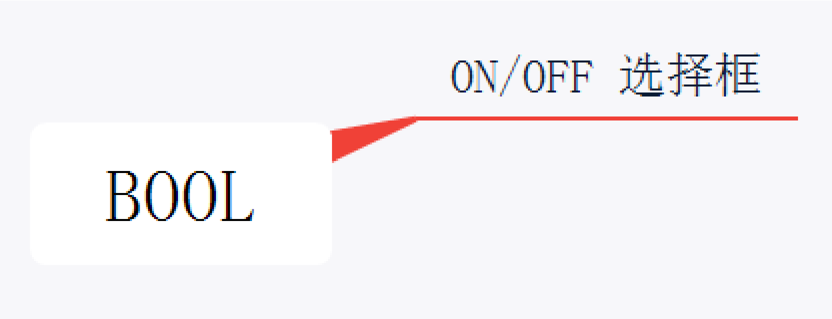
2.2.1. set(<variable> <value>... CACHE <type> <docstring> [FORCE])



type



BOOL



ON/OFF 选择框

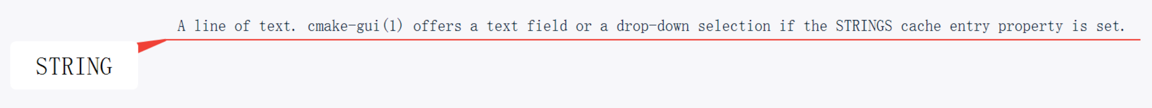
FILEPATH

文件选择

PATH

目录选择

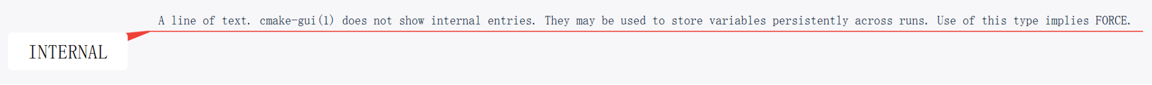
STRING



A line of text. cmake-gui(1) offers a text field or a drop-down selection if the STRINGS cache entry property is set.

A line of text. cmake-gui(1) offers a text field or a drop-down selection if the STRINGS cache entry property is set.

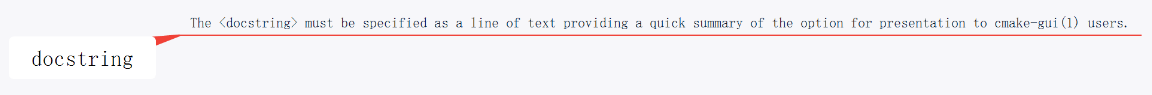
INTERNAL



A line of text. cmake-gui(1) does not show internal entries. They may be used to store variables persistently across runs. Use of this type implies FORCE.

A line of text. cmake-gui(1) does not show internal entries. They may be used to store variables persistently across runs. Use of this type implies FORCE.

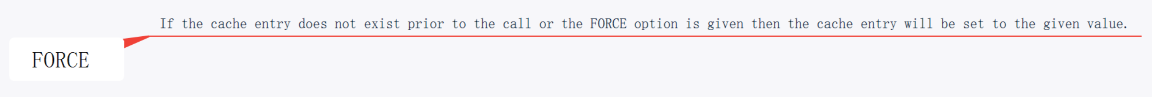
docstring



The <docstring> must be specified as a line of text providing a quick summary of the option for presentation to cmake-gui(1) users.

The <docstring> must be specified as a line of text providing a quick summary of the option for presentation to cmake-gui(1) users.

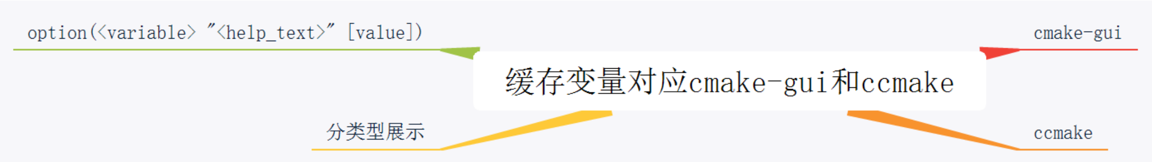
FORCE



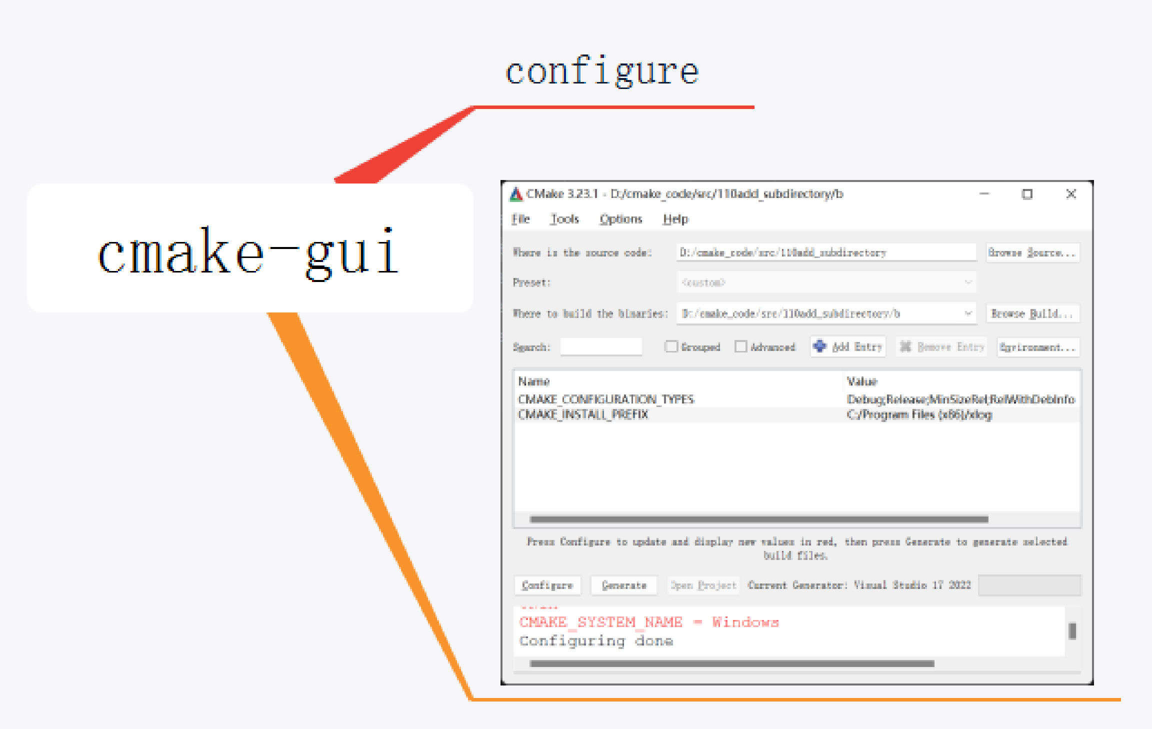
If the cache entry does not exist prior to the call or the FORCE option is given then the cache entry will be set to the given value.

If the cache entry does not exist prior to the call or the FORCE option is given then the cache entry will be set to the given value.

2.3. 缓存变量对应cmake-gui和ccmake


option(<variable> "<help_text>" [value])
缓存变量对应cmake-gui和ccmake
cmake-gui
分类型展示
ccmake

2.3.1. cmake-gui



Where is the source code: D:/cmake_code/src/110add_subdirectory

Preset: <custom>

Where to build the binaries: D:/cmake_code/src/110add_subdirectory/b

Search:

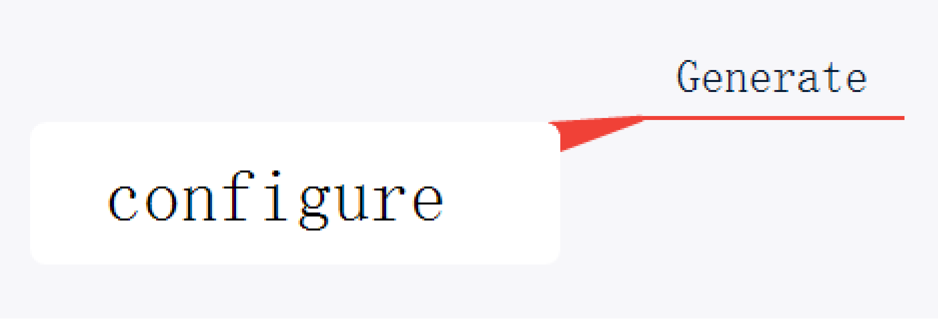
Name	Value
CMAKE_CONFIGURATION_TYPES	Debug;Release;MinSizeRel;RelWithDebInfo
CMAKE_INSTALL_PREFIX	C:/Program Files (x86)/xlog

Press Configure to update and display new values in red, then press Generate to generate selected build files.

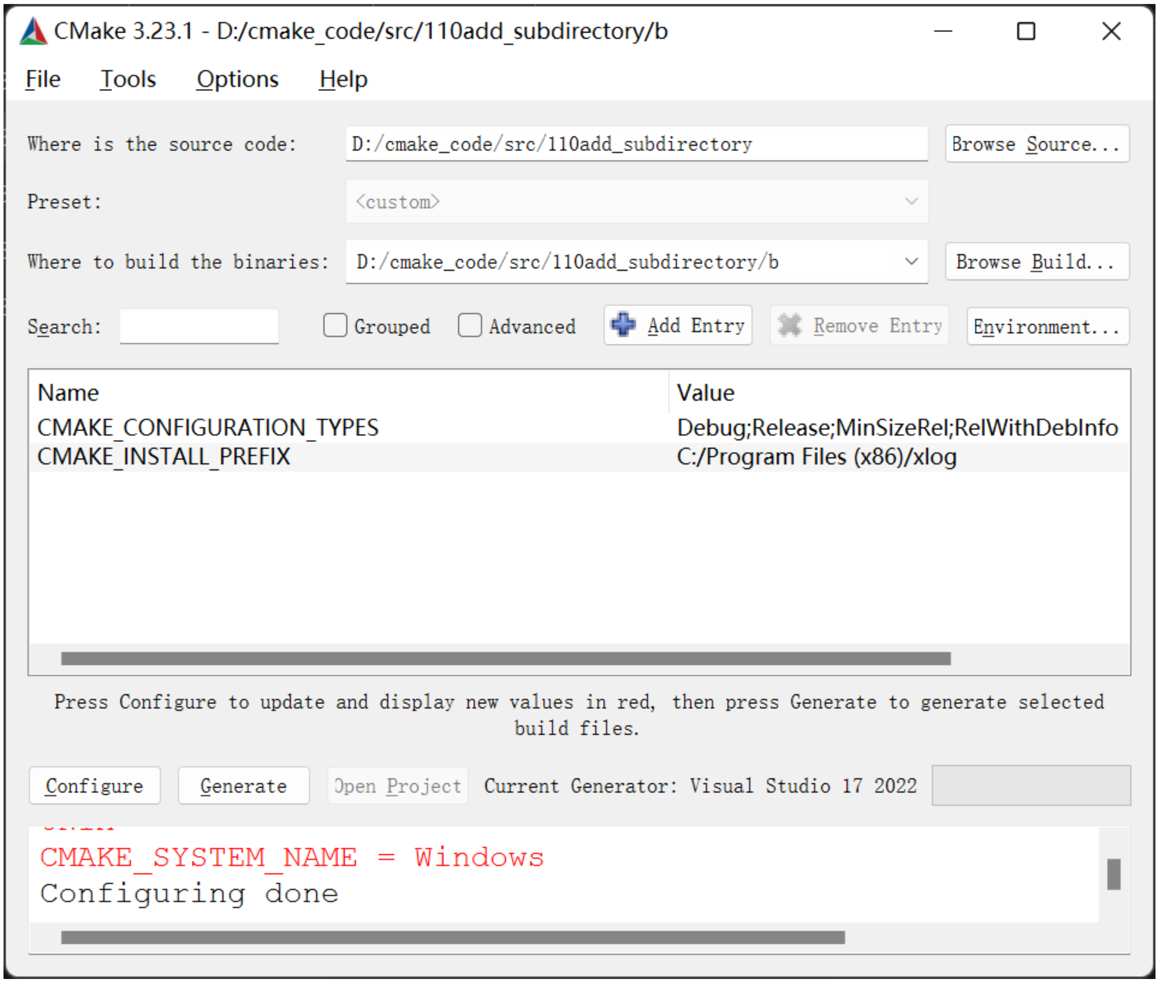
Current Generator: Visual Studio 17 2022

```
CMAKE_SYSTEM_NAME = Windows
Configuring done
```

configure



Generate



Where is the source code: D:/cmake_code/src/110add_subdirectory

Preset: <custom>

Where to build the binaries: D:/cmake_code/src/110add_subdirectory/b

Search:

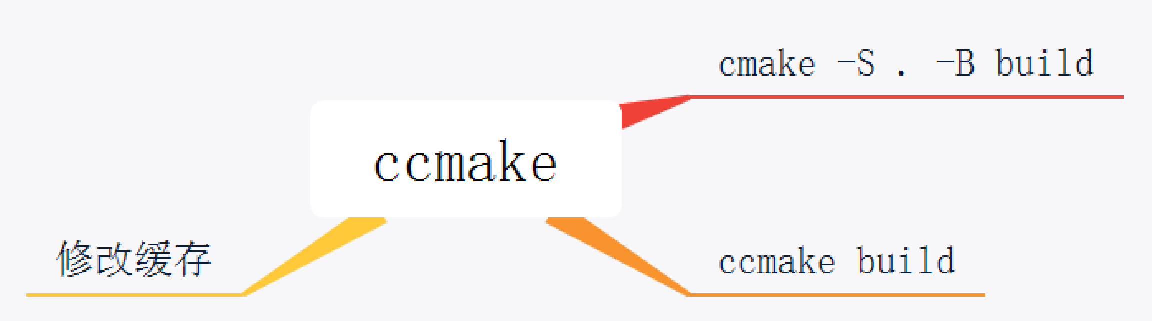
Name	Value
CMAKE_CONFIGURATION_TYPES	Debug;Release;MinSizeRel;RelWithDebInfo
CMAKE_INSTALL_PREFIX	C:/Program Files (x86)/xlog

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 17 2022

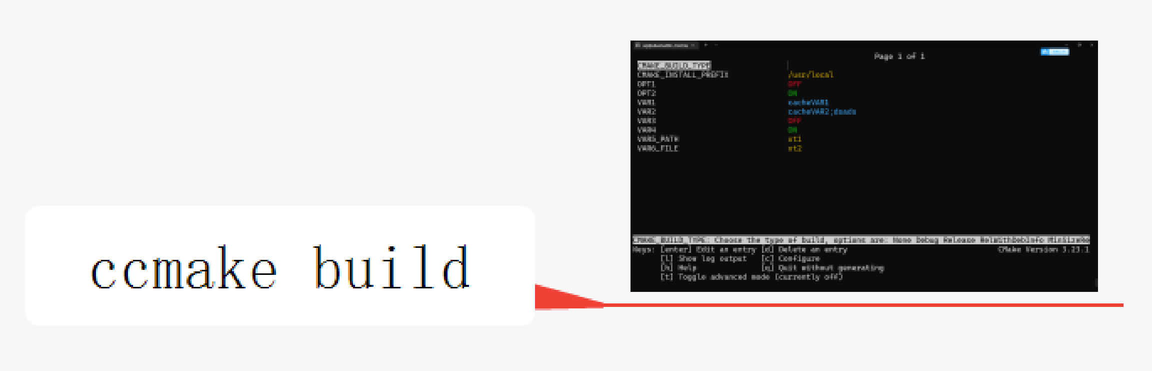
```
CMAKE_SYSTEM_NAME = Windows
Configuring done
```

2.3.2. ccmake

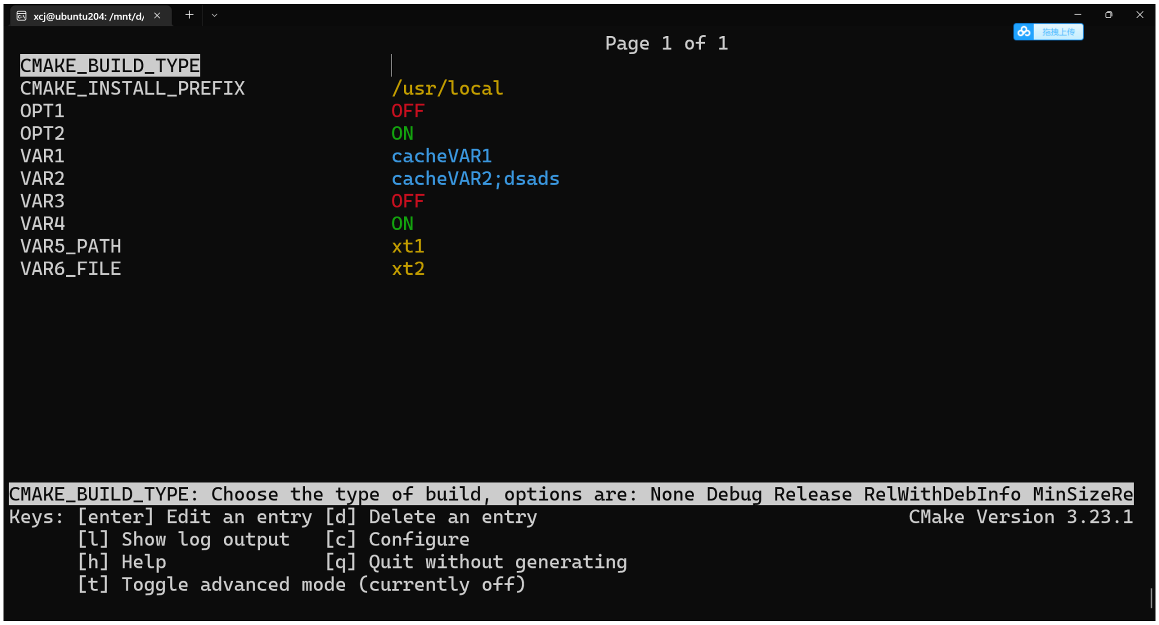


cmake -S . -B build

ccmake build



ccmake build



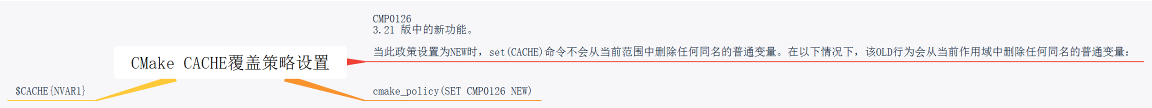
```
                                        Page 1 of 1
 CMAKE_BUILD_TYPE
 CMAKE_INSTALL_PREFIX             /usr/local
 OPT1                             OFF
 OPT2                             ON
 VAR1                             cacheVAR1
 VAR2                             cacheVAR2;dsads
 VAR3                             OFF
 VAR4                             ON
 VAR5_PATH                        xt1
 VAR6_FILE                        xt2







CMAKE_BUILD_TYPE: Choose the type of build, options are: None Debug Release RelWithDebInfo MinSizeRe
Keys: [enter] Edit an entry [d] Delete an entry                                  CMake Version 3.23.1
      [l] Show log output   [c] Configure
      [h] Help              [q] Quit without generating
      [t] Toggle advanced mode (currently off)
```

修改缓存

2.3.3. 分类型展示

2.3.4. option(<variable> "<help_text>" [value])

2.4. CMake CACHE覆盖策略设置



CMake CACHE覆盖策略设置

CMP0126
3.21 版中的新功能。

当此政策设置为NEW时，set(CACHE)命令不会从当前范围中删除任何同名的普通变量。在以下情况下，该OLD行为会从当前作用域中删除任何同名的普通变量：

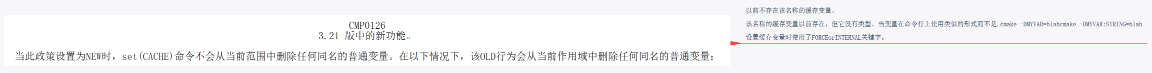
$CACHE{NVAR1}

cmake_policy(SET CMP0126 NEW)

2.4.1. CMP0126

3.21 版中的新功能。

当此政策设置为NEW时，set(CACHE)命令不会从当前范围中删除任何同名的普通变量。在以下情况下，该OLD行为会从当前作用域中删除任何同名的普通变量：



CMP0126
3.21 版中的新功能。

当此政策设置为NEW时，set(CACHE)命令不会从当前范围中删除任何同名的普通变量。在以下情况下，该OLD行为会从当前作用域中删除任何同名的普通变量：

以前不存在该名称的缓存变量。

该名称的缓存变量以前存在，但它没有类型。当变量在命令行上使用类似的形式而不是.cmake -DMYVAR=blahcmake -DMYVAR:STRING=blah

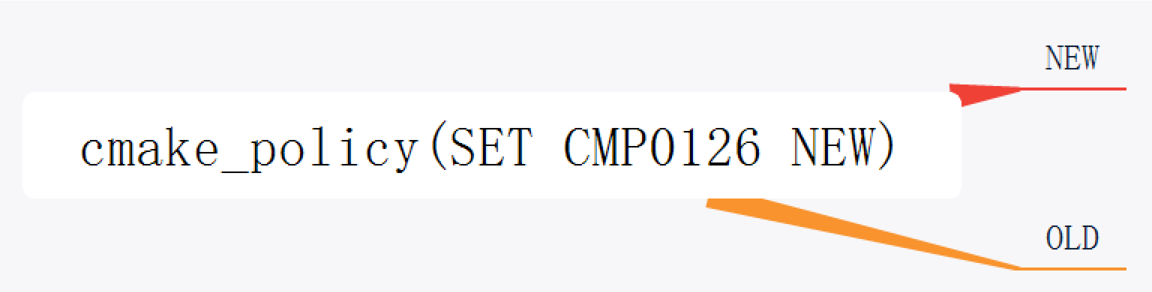
设置缓存变量时使用了FORCEorINTERNAL关键字。

以前不存在该名称的缓存变量。

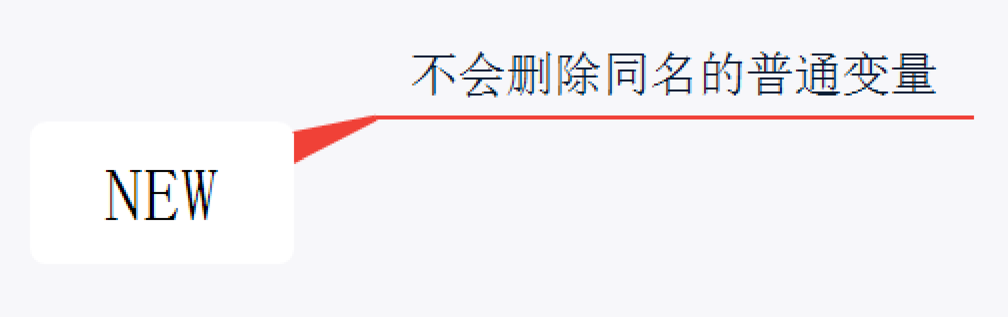
该名称的缓存变量以前存在，但它没有类型。当变量在命令行上使用类似的形式而不是.cmake -DMYVAR=blahcmake -DMYVAR:STRING=blah

设置缓存变量时使用了FORCEorINTERNAL关键字。

2.4.2. cmake_policy(SET CMP0126 NEW)



NEW



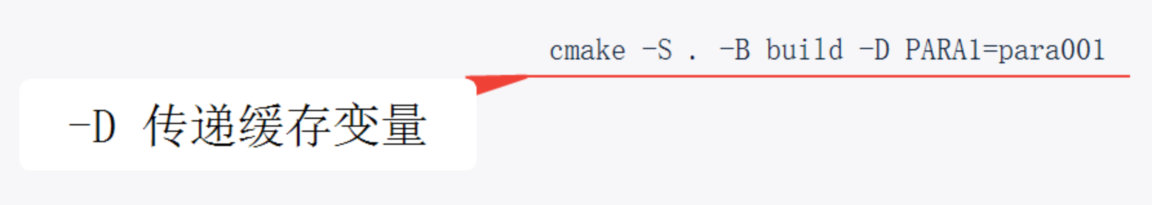
不会删除同名的普通变量

OLD

删除同名的普通变量

2.4.3. $CACHE{NVAR1}

2.5. -D 传递缓存变量



2.5.1. cmake -S . -B build -D PARA1=para001

2.6. CMake内置缓存变量

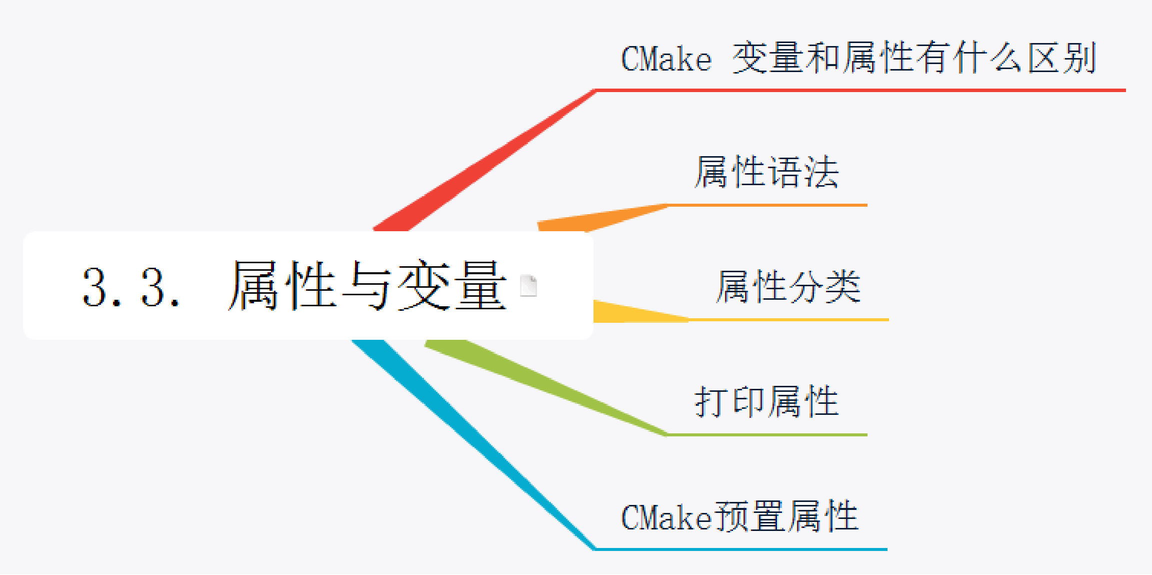
CMake内置缓存变量
BUILD_SHARED_LIBS
message("BUILD_SHARED_LIBS = ${BUILD_SHARED_LIBS}")
set(BUILD_SHARED_LIBS OFF CACHE BOOL "lib")

2.6.1. BUILD_SHARED_LIBS

2.6.2. set(BUILD_SHARED_LIBS OFF CACHE BOOL "lib")

2.6.3. message("BUILD_SHARED_LIBS = ${BUILD_SHARED_LIBS}")

3. 3.3. 属性与变量



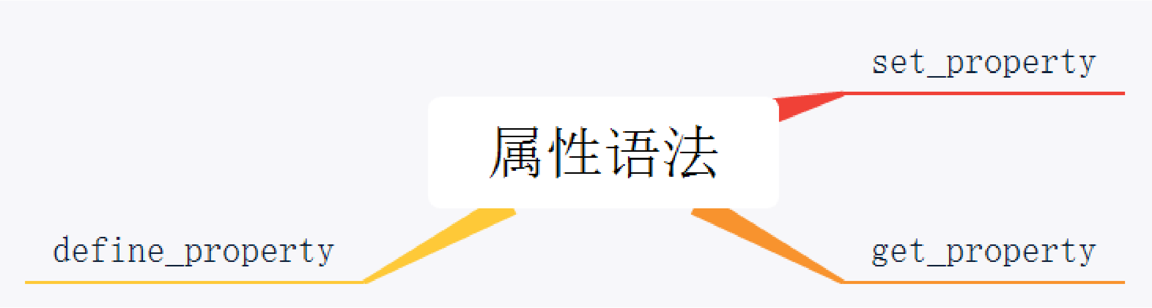
3.1. CMake 变量和属性有什么区别

CMake 变量和属性有什么区别
一种简短的说明是，属性是作用域为目标的变量。
global property can be a useful uncached global variable

3.1.1. 一种简短的说明是，属性是作用域为目标的变量。

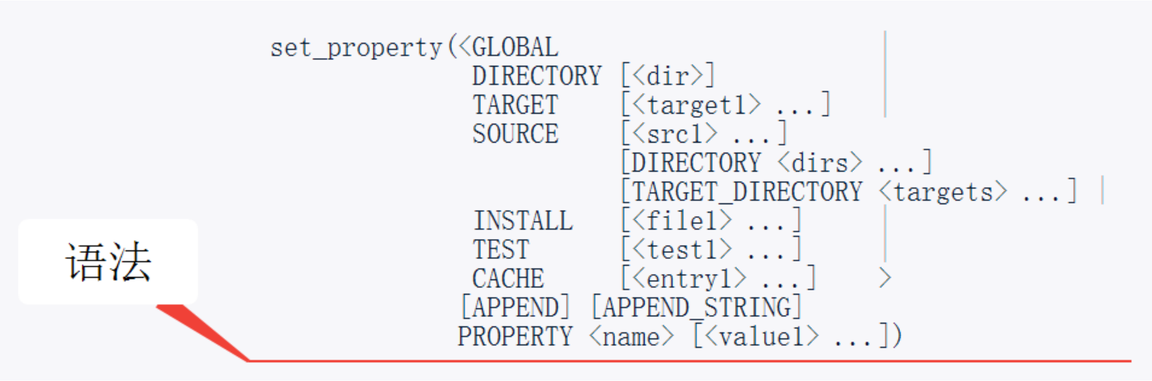
3.1.2. global property can be a useful uncached global variable

3.2. 属性语法



3.2.1. set_property

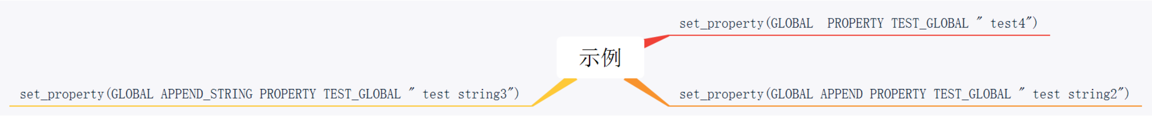
语法



```
set_property(<GLOBAL                      |
              DIRECTORY [<dir>]           |
              TARGET    [<target1> ...]   |
              SOURCE    [<src1> ...]
                        [DIRECTORY <dirs> ...]
                        [TARGET_DIRECTORY <targets> ...] |
              INSTALL   [<file1> ...]     |
              TEST      [<test1> ...]     |
              CACHE     [<entry1> ...]    >
             [APPEND] [APPEND_STRING]
             PROPERTY <name> [<value1> ...])
```

```
set_property(<GLOBAL                   |
         DIRECTORY [<dir>]         |
         TARGET    [<target1> ...]  |
         SOURCE    [<src1> ...]
               [DIRECTORY <dirs> ...]
               [TARGET_DIRECTORY <targets> ...] |
         INSTALL   [<file1> ...]    |
         TEST      [<test1> ...]    |
         CACHE     [<entry1> ...]   >
        [APPEND] [APPEND_STRING]
        PROPERTY <name> [<value1> ...])
```

示例



set_property(GLOBAL PROPERTY TEST_GLOBAL ″ test4″)

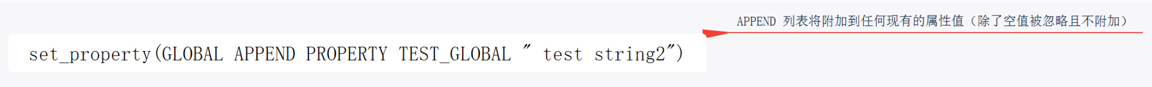
示例

set_property(GLOBAL APPEND_STRING PROPERTY TEST_GLOBAL ″ test string3″)

set_property(GLOBAL APPEND PROPERTY TEST_GLOBAL ″ test string2″)

set_property(GLOBAL PROPERTY TEST_GLOBAL " test4")

set_property(GLOBAL APPEND PROPERTY TEST_GLOBAL " test string2")

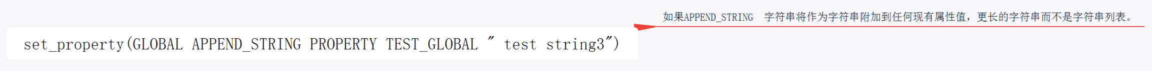


APPEND 列表将附加到任何现有的属性值（除了空值被忽略且不附加）

set_property(GLOBAL APPEND PROPERTY TEST_GLOBAL ″ test string2″)

APPEND 列表将附加到任何现有的属性值（除了空值被忽略且不附加）

set_property(GLOBAL APPEND_STRING PROPERTY TEST_GLOBAL " test string3")



如果APPEND_STRING 字符串将作为字符串附加到任何现有属性值，更长的字符串而不是字符串列表。

set_property(GLOBAL APPEND_STRING PROPERTY TEST_GLOBAL ″ test string3″)

如果APPEND_STRING

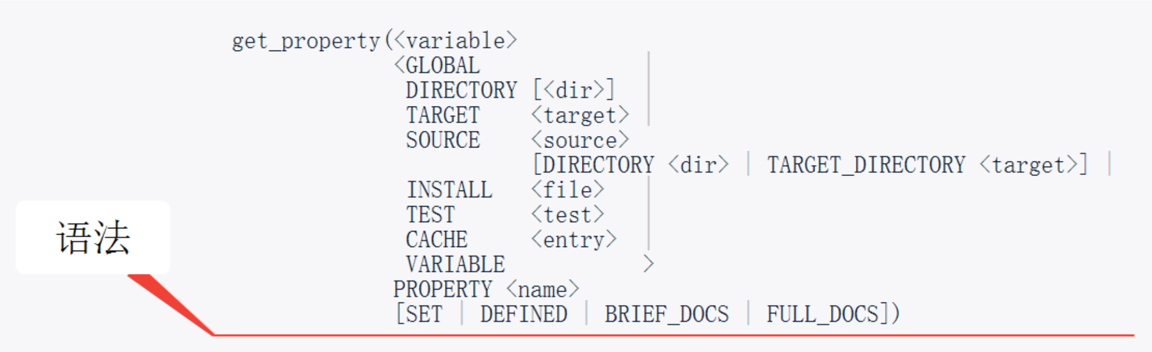
字符串将作为字符串附加到任何现有属性值，更长的字符串而不是字符串列表。

3.2.2. get_property

语法

get_property

语法



语法

```
get_property(<variable>
             <GLOBAL             |
              DIRECTORY [<dir>]  |
              TARGET    <target> |
              SOURCE    <source>
                        [DIRECTORY <dir> | TARGET_DIRECTORY <target>] |
              INSTALL   <file>   |
              TEST      <test>   |
              CACHE     <entry>  |
              VARIABLE           >
             PROPERTY <name>
             [SET | DEFINED | BRIEF_DOCS | FULL_DOCS])
```

get_property(<variable>

<GLOBAL |

DIRECTORY [<dir>] |

TARGET <target> |

SOURCE <source>

[DIRECTORY <dir> | TARGET_DIRECTORY <target>] |

INSTALL <file> |

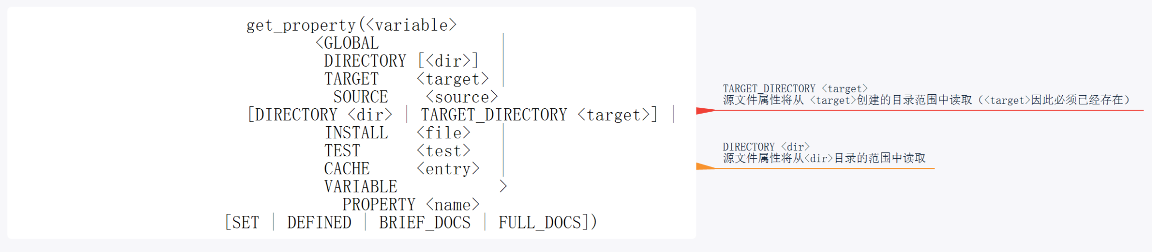
TEST <test> |

CACHE <entry> |

VARIABLE >

PROPERTY <name>

[SET | DEFINED | BRIEF_DOCS | FULL_DOCS])



```
get_property(<variable>
         <GLOBAL             |
          DIRECTORY [<dir>]  |
          TARGET    <target> |
           SOURCE    <source>
[DIRECTORY <dir> | TARGET_DIRECTORY <target>] |
          INSTALL   <file>   |
          TEST      <test>   |
          CACHE     <entry>  |
          VARIABLE           >
            PROPERTY <name>
[SET | DEFINED | BRIEF_DOCS | FULL_DOCS])
```

TARGET_DIRECTORY <target>
源文件属性将从 <target>创建的目录范围中读取（<target>因此必须已经存在）

DIRECTORY <dir>
源文件属性将从<dir>目录的范围中读取

TARGET_DIRECTORY <target>

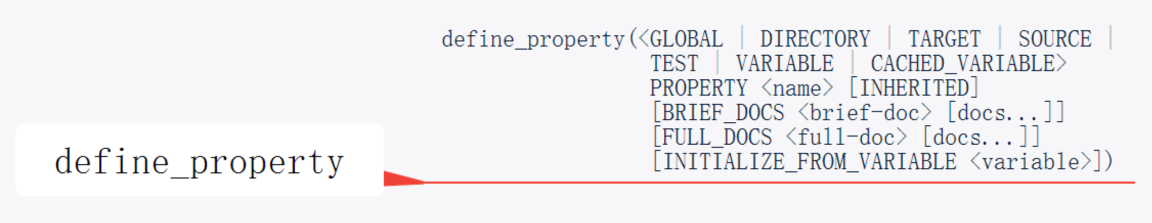
源文件属性将从

<target>创建的目录范围中读取（<target>因此必须已经存在）

DIRECTORY <dir>

源文件属性将从<dir>目录的范围中读取

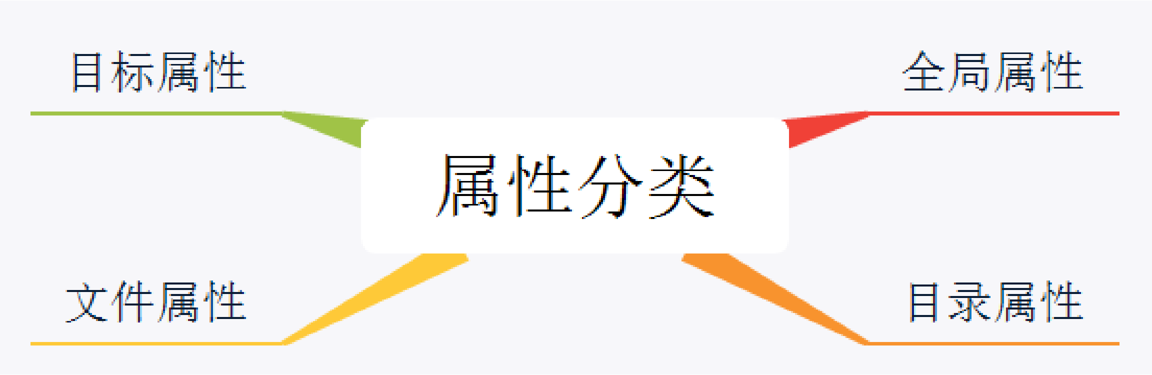
3.2.3. define_property



```
define_property(<GLOBAL | DIRECTORY | TARGET | SOURCE |
                TEST | VARIABLE | CACHED_VARIABLE>
                PROPERTY <name> [INHERITED]
                [BRIEF_DOCS <brief-doc> [docs...]]
                [FULL_DOCS <full-doc> [docs...]]
                [INITIALIZE_FROM_VARIABLE <variable>])
```

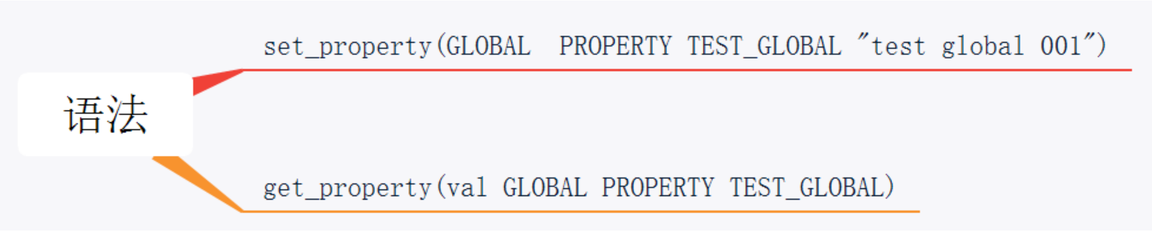
define_property(<GLOBAL | DIRECTORY | TARGET | SOURCE |
TEST | VARIABLE | CACHED_VARIABLE>
PROPERTY <name> [INHERITED]
[BRIEF_DOCS <brief-doc> [docs...]]
[FULL_DOCS <full-doc> [docs...]]
[INITIALIZE_FROM_VARIABLE <variable>])

3.3. 属性分类



3.3.1. 全局属性

语法



语法

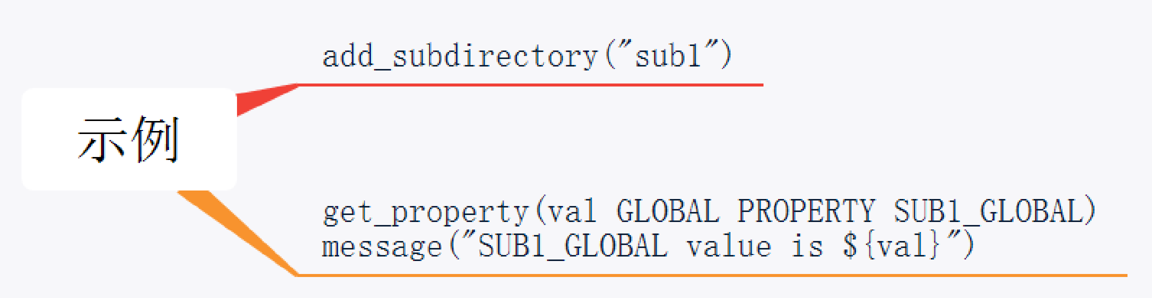
set_property(GLOBAL PROPERTY TEST_GLOBAL "test global 001")

get_property(val GLOBAL PROPERTY TEST_GLOBAL)

set_property(GLOBAL PROPERTY TEST_GLOBAL "test global 001")

get_property(val GLOBAL PROPERTY TEST_GLOBAL)

示例

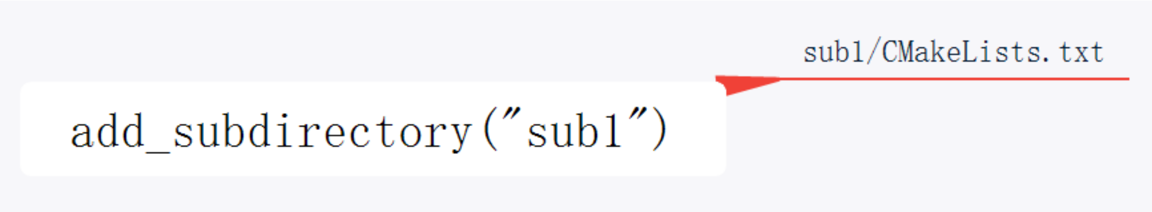


示例

add_subdirectory("sub1")

get_property(val GLOBAL PROPERTY SUB1_GLOBAL)
message("SUB1_GLOBAL value is ${val}")

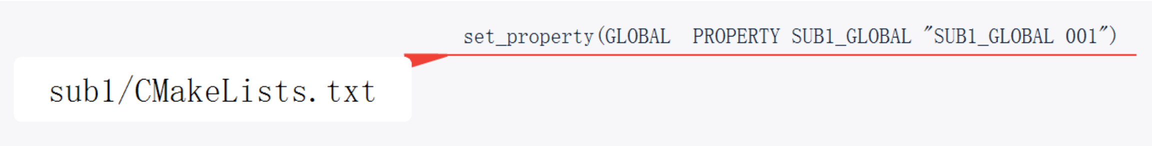
add_subdirectory("sub1")



add_subdirectory("sub1")

sub1/CMakeLists.txt

sub1/CMakeLists.txt



sub1/CMakeLists.txt

set_property(GLOBAL PROPERTY SUB1_GLOBAL "SUB1_GLOBAL 001")

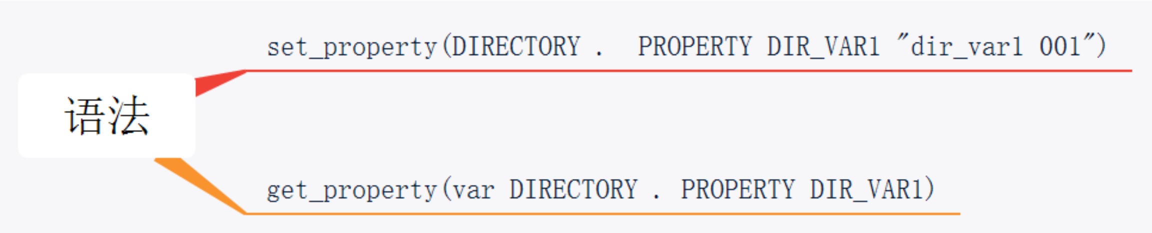
set_property(GLOBAL PROPERTY SUB1_GLOBAL "SUB1_GLOBAL 001")

get_property(val GLOBAL PROPERTY SUB1_GLOBAL)

message("SUB1_GLOBAL value is ${val}")

3.3.2. 目录属性

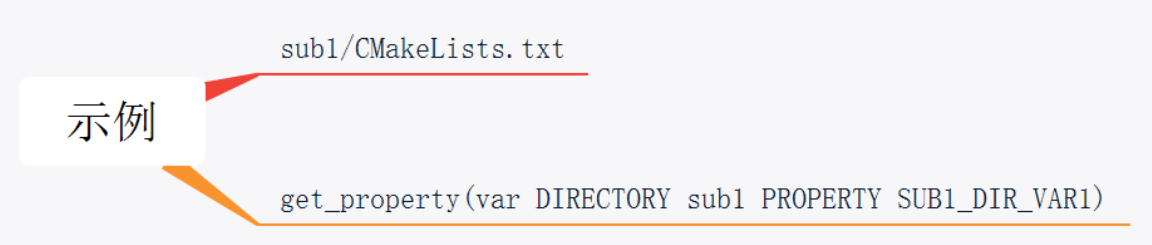
语法


语法
set_property(DIRECTORY . PROPERTY DIR_VAR1 "dir_var1 001")
get_property(var DIRECTORY . PROPERTY DIR_VAR1)

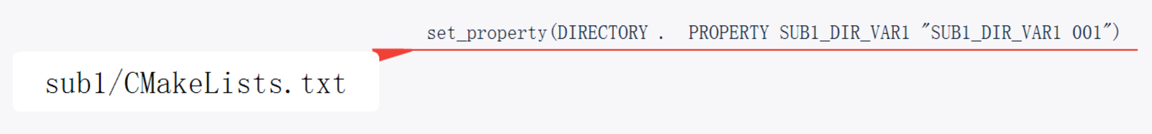
set_property(DIRECTORY . PROPERTY DIR_VAR1 "dir_var1 001")

get_property(var DIRECTORY . PROPERTY DIR_VAR1)

示例



sub1/CMakeLists.txt

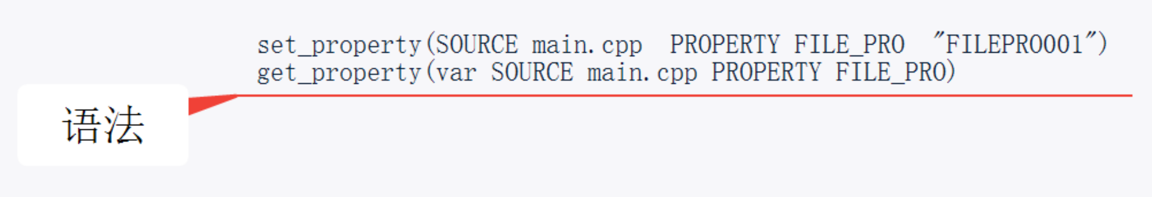

sub1/CMakeLists.txt
set_property(DIRECTORY . PROPERTY SUB1_DIR_VAR1 "SUB1_DIR_VAR1 001")

set_property(DIRECTORY . PROPERTY SUB1_DIR_VAR1 "SUB1_DIR_VAR1 001")

get_property(var DIRECTORY sub1 PROPERTY SUB1_DIR_VAR1)

3.3.3. 文件属性

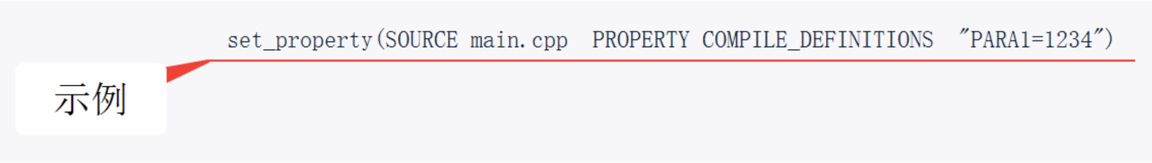
语法



```
set_property(SOURCE main.cpp  PROPERTY FILE_PRO  ″FILEPRO001″)
get_property(var SOURCE main.cpp PROPERTY FILE_PRO)
```

```
set_property(SOURCE main.cpp  PROPERTY FILE_PRO  "FILEPRO001")
get_property(var SOURCE main.cpp PROPERTY FILE_PRO)
```

示例

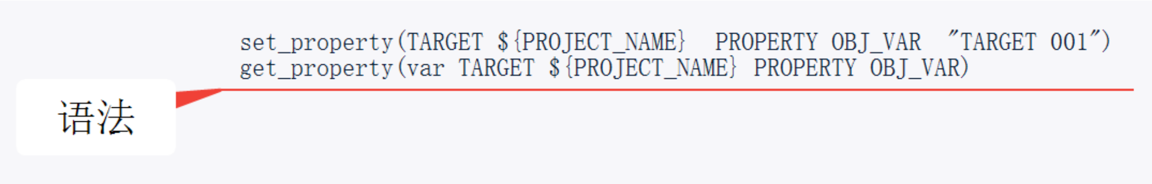


```
set_property(SOURCE main.cpp  PROPERTY COMPILE_DEFINITIONS  ″PARA1=1234″)
```

```
set_property(SOURCE main.cpp  PROPERTY COMPILE_DEFINITIONS
"PARA1=1234")
```

3.3.4. 目标属性

语法

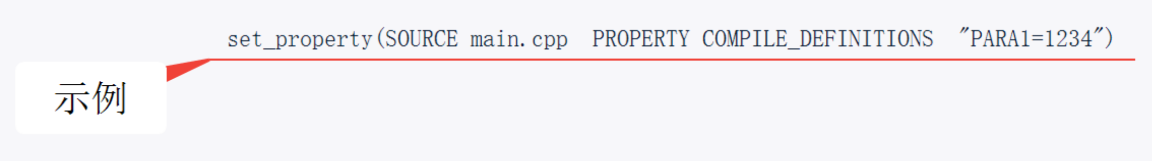


```
set_property(TARGET ${PROJECT_NAME}  PROPERTY OBJ_VAR  ”TARGET 001”)
get_property(var TARGET ${PROJECT_NAME} PROPERTY OBJ_VAR)
```

set_property(TARGET ${PROJECT_NAME} PROPERTY OBJ_VAR "TARGET 001")

get_property(var TARGET ${PROJECT_NAME} PROPERTY OBJ_VAR)

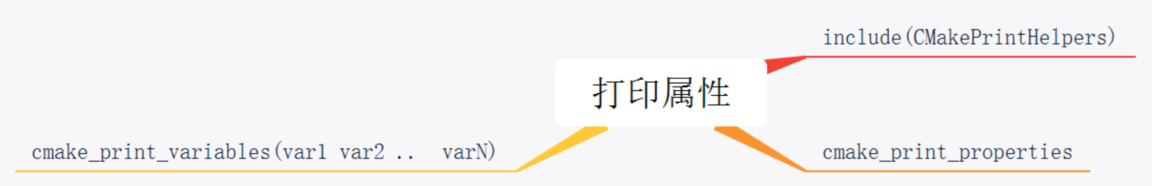
示例



```
set_property(SOURCE main.cpp  PROPERTY COMPILE_DEFINITIONS  ”PARA1=1234”)
```

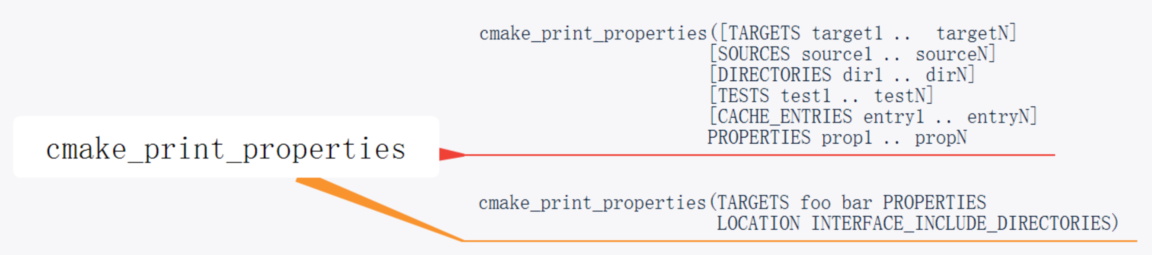
set_property(SOURCE main.cpp PROPERTY COMPILE_DEFINITIONS "PARA1=1234")

3.4. 打印属性


打印属性
include(CMakePrintHelpers)
cmake_print_variables(var1 var2 .. varN)
cmake_print_properties

3.4.1. include(CMakePrintHelpers)

3.4.2. cmake_print_properties



```
cmake_print_properties([TARGETS target1 ..  targetN]
                       [SOURCES source1 .. sourceN]
                       [DIRECTORIES dir1 .. dirN]
                       [TESTS test1 .. testN]
                       [CACHE_ENTRIES entry1 .. entryN]
                       PROPERTIES prop1 .. propN
```

```
cmake_print_properties(TARGETS foo bar PROPERTIES
                        LOCATION INTERFACE_INCLUDE_DIRECTORIES)
```

cmake_print_properties([TARGETS target1 .. targetN]

[SOURCES source1 .. sourceN]

[DIRECTORIES dir1 .. dirN]

[TESTS test1 .. testN]

[CACHE_ENTRIES entry1 .. entryN]

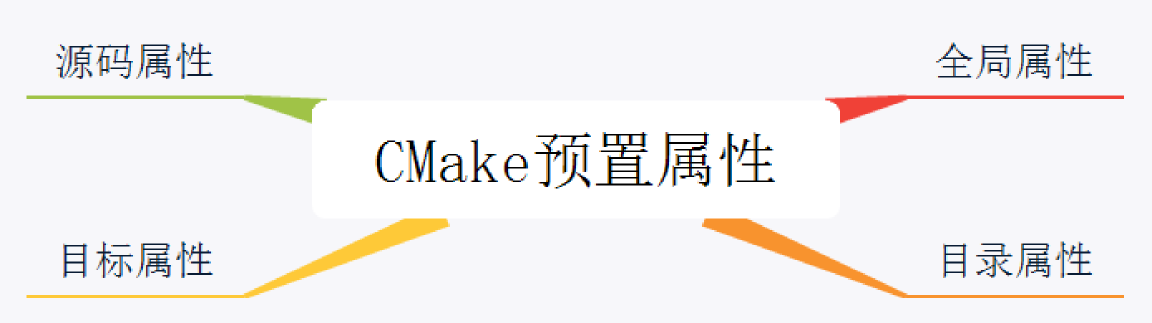
PROPERTIES prop1 .. propN

cmake_print_properties(TARGETS foo bar PROPERTIES

LOCATION INTERFACE_INCLUDE_DIRECTORIES)

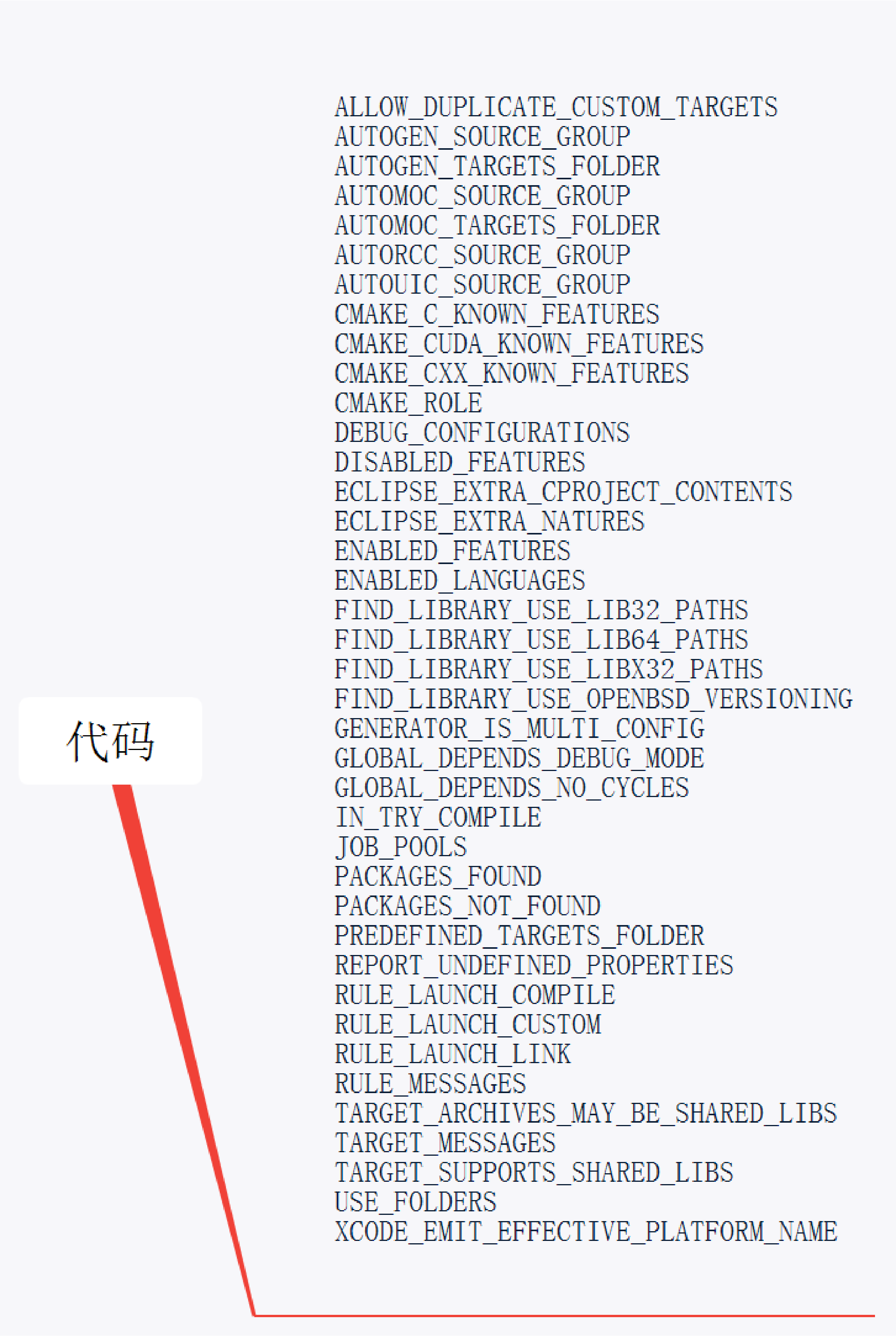
3.4.3. cmake_print_variables(var1 var2 .. varN)

3.5. CMake预置属性



3.5.1. 全局属性

代码



代码

```
ALLOW_DUPLICATE_CUSTOM_TARGETS
AUTOGEN_SOURCE_GROUP
AUTOGEN_TARGETS_FOLDER
AUTOMOC_SOURCE_GROUP
AUTOMOC_TARGETS_FOLDER
AUTORCC_SOURCE_GROUP
AUTOUIC_SOURCE_GROUP
CMAKE_C_KNOWN_FEATURES
CMAKE_CUDA_KNOWN_FEATURES
CMAKE_CXX_KNOWN_FEATURES
CMAKE_ROLE
DEBUG_CONFIGURATIONS
DISABLED_FEATURES
ECLIPSE_EXTRA_CPROJECT_CONTENTS
ECLIPSE_EXTRA_NATURES
ENABLED_FEATURES
ENABLED_LANGUAGES
FIND_LIBRARY_USE_LIB32_PATHS
FIND_LIBRARY_USE_LIB64_PATHS
FIND_LIBRARY_USE_LIBX32_PATHS
FIND_LIBRARY_USE_OPENBSD_VERSIONING
GENERATOR_IS_MULTI_CONFIG
GLOBAL_DEPENDS_DEBUG_MODE
GLOBAL_DEPENDS_NO_CYCLES
IN_TRY_COMPILE
JOB_POOLS
PACKAGES_FOUND
PACKAGES_NOT_FOUND
PREDEFINED_TARGETS_FOLDER
REPORT_UNDEFINED_PROPERTIES
RULE_LAUNCH_COMPILE
RULE_LAUNCH_CUSTOM
RULE_LAUNCH_LINK
RULE_MESSAGES
TARGET_ARCHIVES_MAY_BE_SHARED_LIBS
TARGET_MESSAGES
TARGET_SUPPORTS_SHARED_LIBS
USE_FOLDERS
XCODE_EMIT_EFFECTIVE_PLATFORM_NAME
```

ALLOW_DUPLICATE_CUSTOM_TARGETS

AUTOGEN_SOURCE_GROUP

AUTOGEN_TARGETS_FOLDER

AUTOMOC_SOURCE_GROUP

AUTOMOC_TARGETS_FOLDER

AUTORCC_SOURCE_GROUP

AUTOUIC_SOURCE_GROUP

CMAKE_C_KNOWN_FEATURES

CMAKE_CUDA_KNOWN_FEATURES

CMAKE_CXX_KNOWN_FEATURES

CMAKE_ROLE

DEBUG_CONFIGURATIONS

DISABLED_FEATURES

ECLIPSE_EXTRA_CPROJECT_CONTENTS

ECLIPSE_EXTRA_NATURES

ENABLED_FEATURES

ENABLED_LANGUAGES

FIND_LIBRARY_USE_LIB32_PATHS

FIND_LIBRARY_USE_LIB64_PATHS

FIND_LIBRARY_USE_LIBX32_PATHS

FIND_LIBRARY_USE_OPENBSD_VERSIONING

GENERATOR_IS_MULTI_CONFIG

GLOBAL_DEPENDS_DEBUG_MODE

GLOBAL_DEPENDS_NO_CYCLES

IN_TRY_COMPILE

JOB_POOLS

PACKAGES_FOUND

PACKAGES_NOT_FOUND

PREDEFINED_TARGETS_FOLDER

REPORT_UNDEFINED_PROPERTIES

RULE_LAUNCH_COMPILE

RULE_LAUNCH_CUSTOM

RULE_LAUNCH_LINK

RULE_MESSAGES

TARGET_ARCHIVES_MAY_BE_SHARED_LIBS

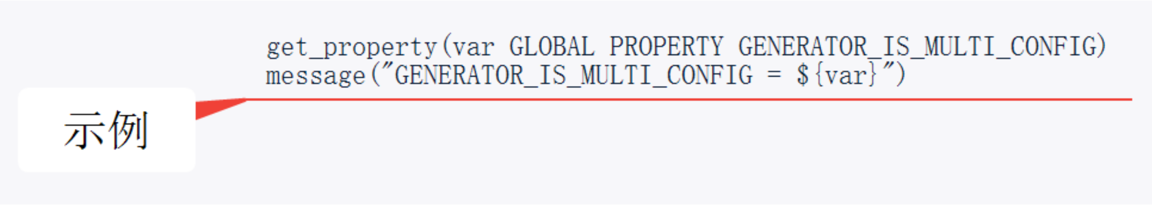
TARGET_MESSAGES

TARGET_SUPPORTS_SHARED_LIBS

USE_FOLDERS

XCODE_EMIT_EFFECTIVE_PLATFORM_NAME

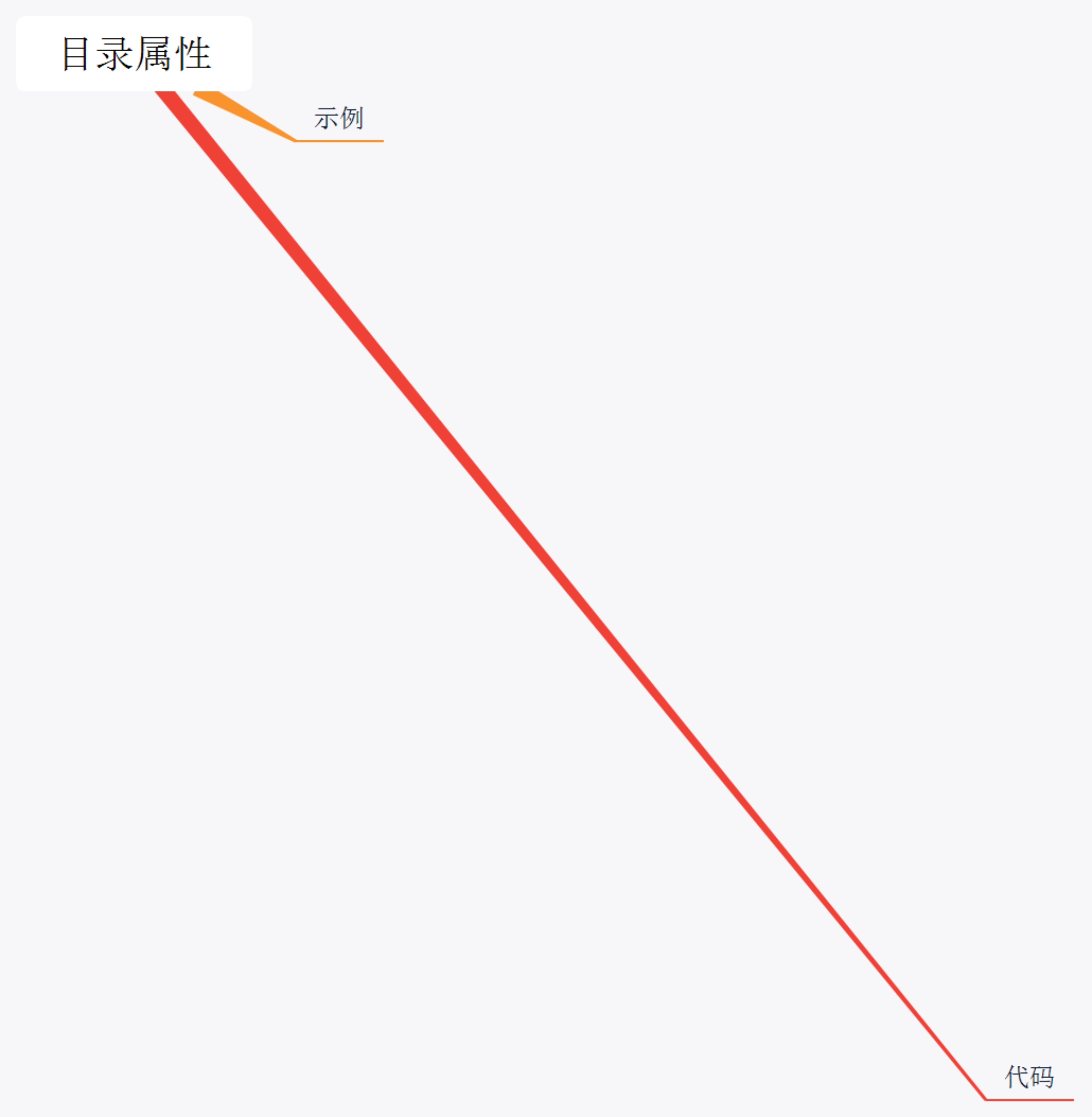
示例



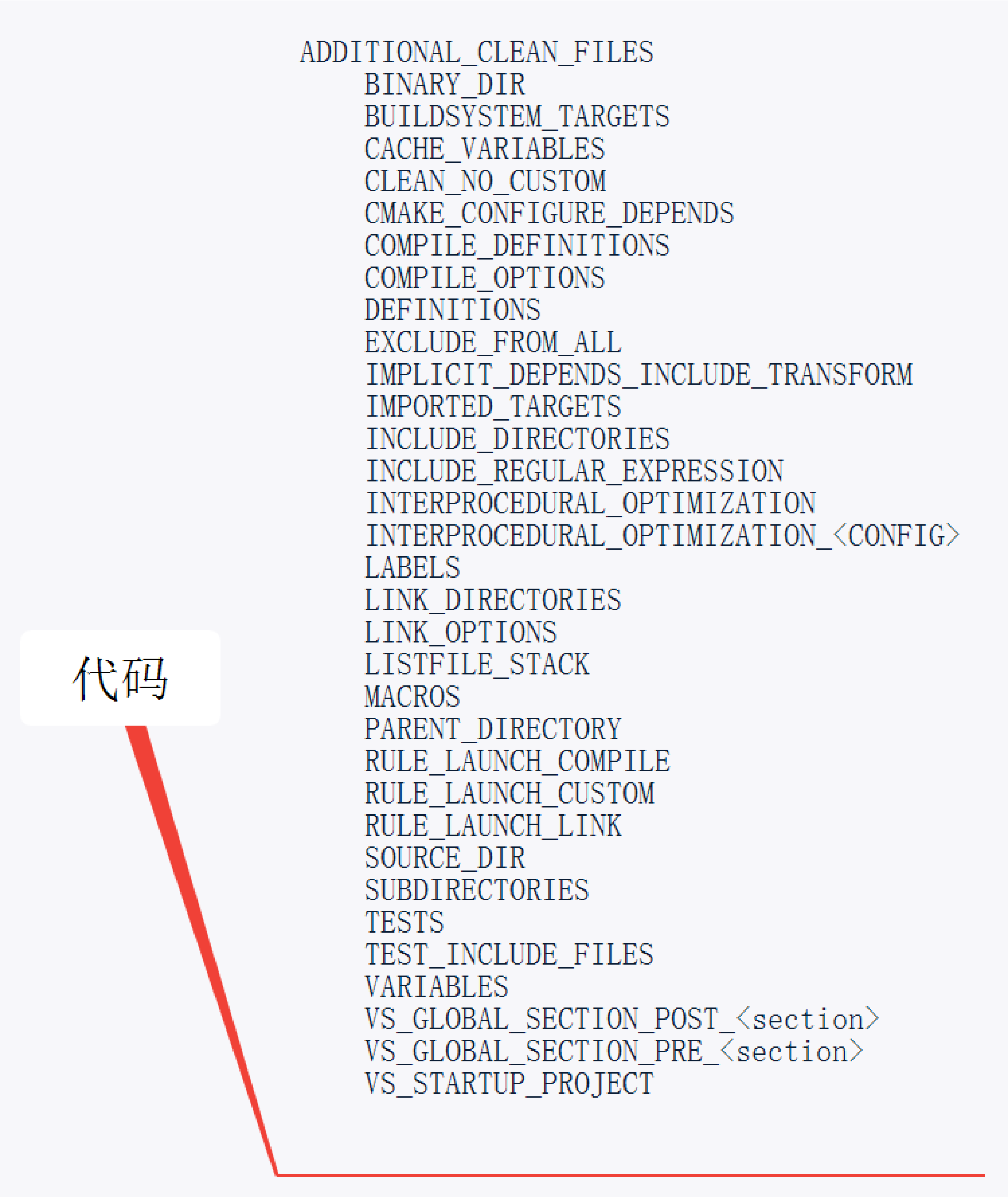
```
get_property(var GLOBAL PROPERTY GENERATOR_IS_MULTI_CONFIG)
message("GENERATOR_IS_MULTI_CONFIG = ${var}")
```

```
get_property(var GLOBAL PROPERTY GENERATOR_IS_MULTI_CONFIG)
message("GENERATOR_IS_MULTI_CONFIG = ${var}")
```

3.5.2. 目录属性



代码



```
ADDITIONAL_CLEAN_FILES
    BINARY_DIR
    BUILDSYSTEM_TARGETS
    CACHE_VARIABLES
    CLEAN_NO_CUSTOM
    CMAKE_CONFIGURE_DEPENDS
    COMPILE_DEFINITIONS
    COMPILE_OPTIONS
    DEFINITIONS
    EXCLUDE_FROM_ALL
    IMPLICIT_DEPENDS_INCLUDE_TRANSFORM
    IMPORTED_TARGETS
    INCLUDE_DIRECTORIES
    INCLUDE_REGULAR_EXPRESSION
    INTERPROCEDURAL_OPTIMIZATION
    INTERPROCEDURAL_OPTIMIZATION_<CONFIG>
    LABELS
    LINK_DIRECTORIES
    LINK_OPTIONS
    LISTFILE_STACK
    MACROS
    PARENT_DIRECTORY
    RULE_LAUNCH_COMPILE
    RULE_LAUNCH_CUSTOM
    RULE_LAUNCH_LINK
    SOURCE_DIR
    SUBDIRECTORIES
    TESTS
    TEST_INCLUDE_FILES
    VARIABLES
    VS_GLOBAL_SECTION_POST_<section>
    VS_GLOBAL_SECTION_PRE_<section>
    VS_STARTUP_PROJECT
```

ADDITIONAL_CLEAN_FILES

BINARY_DIR

BUILDSYSTEM_TARGETS

CACHE_VARIABLES

CLEAN_NO_CUSTOM

CMAKE_CONFIGURE_DEPENDS

COMPILE_DEFINITIONS

COMPILE_OPTIONS

DEFINITIONS

EXCLUDE_FROM_ALL

IMPLICIT_DEPENDS_INCLUDE_TRANSFORM

IMPORTED_TARGETS

INCLUDE_DIRECTORIES

INCLUDE_REGULAR_EXPRESSION

INTERPROCEDURAL_OPTIMIZATION

INTERPROCEDURAL_OPTIMIZATION_<CONFIG>

LABELS

LINK_DIRECTORIES

LINK_OPTIONS

LISTFILE_STACK

MACROS

PARENT_DIRECTORY

RULE_LAUNCH_COMPILE

RULE_LAUNCH_CUSTOM

RULE_LAUNCH_LINK

SOURCE_DIR

SUBDIRECTORIES

TESTS

TEST_INCLUDE_FILES

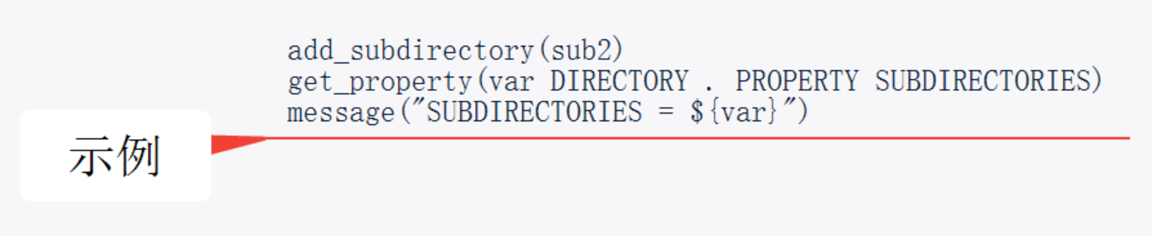
VARIABLES

VS_GLOBAL_SECTION_POST_<section>

VS_GLOBAL_SECTION_PRE_<section>

VS_STARTUP_PROJECT

示例



```
add_subdirectory(sub2)
get_property(var DIRECTORY . PROPERTY SUBDIRECTORIES)
message(”SUBDIRECTORIES = ${var}”)
```

add_subdirectory(sub2)

get_property(var DIRECTORY . PROPERTY SUBDIRECTORIES)

message("SUBDIRECTORIES = ${var}")

3.5.3. 目标属性

代码

```
ADDITIONAL_CLEAN_FILES
AIX_EXPORT_ALL_SYMBOLS
ALIAS_GLOBAL
ALIASED_TARGET
ANDROID_ANT_ADDITIONAL_OPTIONS
ANDROID_API
ANDROID_API_MIN
ANDROID_ARCH
ANDROID_ASSETS_DIRECTORIES
ANDROID_GUI
ANDROID_JAR_DEPENDENCIES
ANDROID_JAR_DIRECTORIES
ANDROID_JAVA_SOURCE_DIR
ANDROID_NATIVE_LIB_DEPENDENCIES
ANDROID_NATIVE_LIB_DIRECTORIES
ANDROID_PROCESS_MAX
ANDROID_PROGUARD
ANDROID_PROGUARD_CONFIG_PATH
ANDROID_SECURE_PROPS_PATH
ANDROID_SKIP_ANT_STEP
ANDROID_STL_TYPE
ARCHIVE_OUTPUT_DIRECTORY
ARCHIVE_OUTPUT_DIRECTORY_<CONFIG>
ARCHIVE_OUTPUT_NAME
ARCHIVE_OUTPUT_NAME_<CONFIG>
AUTOGEN_BUILD_DIR
AUTOGEN_ORIGIN_DEPENDS
AUTOGEN_PARALLEL
AUTOGEN_TARGET_DEPENDS
AUTOMOC
AUTOMOC_COMPILER_PREDEFINES
AUTOMOC_DEPEND_FILTERS
AUTOMOC_EXECUTABLE
AUTOMOC_MACRO_NAMES
AUTOMOC_MOC_OPTIONS
AUTOMOC_PATH_PREFIX
AUTORCC
AUTORCC_EXECUTABLE
AUTORCC_OPTIONS
AUTOUIC
AUTOUIC_EXECUTABLE
AUTOUIC_OPTIONS
AUTOUIC_SEARCH_PATHS
BINARY_DIR
BUILD_RPATH
BUILD_RPATH_USE_ORIGIN
BUILD_WITH_INSTALL_NAME_DIR
BUILD_WITH_INSTALL_RPATH
BUNDLE
BUNDLE_EXTENSION
C_EXTENSIONS
C_STANDARD
C_STANDARD_REQUIRED
COMMON_LANGUAGE_RUNTIME
COMPATIBLE_INTERFACE_BOOL
COMPATIBLE_INTERFACE_NUMBER_MAX
COMPATIBLE_INTERFACE_NUMBER_MIN
COMPATIBLE_INTERFACE_STRING
COMPILE_DEFINITIONS
COMPILE_FEATURES
COMPILE_FLAGS
COMPILE_OPTIONS
COMPILE_PDB_NAME
COMPILE_PDB_NAME_<CONFIG>
COMPILE_PDB_OUTPUT_DIRECTORY
COMPILE_PDB_OUTPUT_DIRECTORY_<CONFIG>
<CONFIG>_OUTPUT_NAME
<CONFIG>_POSTFIX
CROSSCOMPILING_EMULATOR
CUDA_ARCHITECTURES
CUDA_EXTENSIONS
CUDA_PTX_COMPILATION
CUDA_RESOLVE_DEVICE_SYMBOLS
CUDA_RUNTIME_LIBRARY
CUDA_SEPARABLE_COMPILATION
CUDA_STANDARD
CUDA_STANDARD_REQUIRED
CXX_EXTENSIONS
CXX_STANDARD
CXX_STANDARD_REQUIRED
DEBUG_POSTFIX
DEFINE_SYMBOL
DEPLOYMENT_ADDITIONAL_FILES
DEPLOYMENT_REMOTE_DIRECTORY
DEPRECATION
DISABLE_PRECOMPILE_HEADERS
DOTNET_SDK
DOTNET_TARGET_FRAMEWORK
DOTNET_TARGET_FRAMEWORK_VERSION
EchoString
ENABLE_EXPORTS
EXCLUDE_FROM_ALL
EXCLUDE_FROM_DEFAULT_BUILD
EXCLUDE_FROM_DEFAULT_BUILD_<CONFIG>
EXPORT_COMPILE_COMMANDS
EXPORT_NAME
EXPORT_PROPERTIES
FOLDER
Fortran_BUILDING_INSTRINSIC_MODULES
Fortran_FORMAT
Fortran_MODULE_DIRECTORY
Fortran_PREPROCESS
FRAMEWORK
FRAMEWORK_MULTI_CONFIG_POSTFIX_<CONFIG>
FRAMEWORK_VERSION
GENERATOR_FILE_NAME
GHS_INTEGRITY_APP
GHS_NO_SOURCE_GROUP_FILE
GNUtoMS
HAS_CXX
HEADER_DIRS
HEADER_DIRS_<NAME>
HEADER_SET
HEADER_SET_<NAME>
HEADER_SETS
HIP_ARCHITECTURES
HIP_EXTENSIONS
HIP_STANDARD
HIP_STANDARD_REQUIRED
IMPLICIT_DEPENDS_INCLUDE_TRANSFORM
IMPORTED
IMPORTED_COMMON_LANGUAGE_RUNTIME
IMPORTED_CONFIGURATIONS
IMPORTED_GLOBAL
IMPORTED_IMPLIB
IMPORTED_IMPLIB_<CONFIG>
IMPORTED_LIBNAME
IMPORTED_LIBNAME_<CONFIG>
IMPORTED_LINK_DEPENDENT_LIBRARIES
IMPORTED_LINK_DEPENDENT_LIBRARIES_<CONFIG>
IMPORTED_LINK_INTERFACE_LANGUAGES
IMPORTED_LINK_INTERFACE_LANGUAGES_<CONFIG>
IMPORTED_LINK_INTERFACE_LIBRARIES
IMPORTED_LINK_INTERFACE_LIBRARIES_<CONFIG>
IMPORTED_LINK_INTERFACE_MULTIPLICITY
IMPORTED_LINK_INTERFACE_MULTIPLICITY_<CONFIG>
IMPORTED_LOCATION
IMPORTED_LOCATION_<CONFIG>
IMPORTED_NO_SONAME
IMPORTED_NO_SONAME_<CONFIG>
IMPORTED_NO_SYSTEM
IMPORTED_OBJECTS
IMPORTED_OBJECTS_<CONFIG>
IMPORTED_SONAME
IMPORTED_SONAME_<CONFIG>
IMPORT_PREFIX
IMPORT_SUFFIX
INCLUDE_DIRECTORIES
INSTALL_NAME_DIR
INSTALL_REMOVE_ENVIRONMENT_RPATH
INSTALL_RPATH
INSTALL_RPATH_USE_LINK_PATH
INTERFACE_AUTOUIC_OPTIONS
INTERFACE_COMPILE_DEFINITIONS
INTERFACE_COMPILE_FEATURES
INTERFACE_COMPILE_OPTIONS
INTERFACE_HEADER_SETS
INTERFACE_INCLUDE_DIRECTORIES
INTERFACE_LINK_DEPENDS
INTERFACE_LINK_DIRECTORIES
INTERFACE_LINK_LIBRARIES
INTERFACE_LINK_OPTIONS
INTERFACE_POSITION_INDEPENDENT_CODE
INTERFACE_PRECOMPILE_HEADERS
INTERFACE_SOURCES
INTERFACE_SYSTEM_INCLUDE_DIRECTORIES
INTERPROCEDURAL_OPTIMIZATION
INTERPROCEDURAL_OPTIMIZATION_<CONFIG>
IOS_INSTALL_COMBINED
ISPC_HEADER_DIRECTORY
ISPC_HEADER_SUFFIX
ISPC_INSTRUCTION_SETS
JOB_POOL_COMPILE
JOB_POOL_LINK
JOB_POOL_PRECOMPILE_HEADER
LABELS
<LANG>_CLANG_TIDY
<LANG>_COMPILER_LAUNCHER
<LANG>_CPPCHECK
<LANG>_CPPLINT
<LANG>_EXTENSIONS
<LANG>_INCLUDE_WHAT_YOU_USE
<LANG>_LINKER_LAUNCHER
<LANG>_STANDARD
<LANG>_STANDARD_REQUIRED
<LANG>_VISIBILITY_PRESET
LIBRARY_OUTPUT_DIRECTORY
LIBRARY_OUTPUT_DIRECTORY_<CONFIG>
LIBRARY_OUTPUT_NAME
LIBRARY_OUTPUT_NAME_<CONFIG>
LINK_DEPENDS
LINK_DEPENDS_NO_SHARED
LINK_DIRECTORIES
LINK_FLAGS
LINK_FLAGS_<CONFIG>
LINK_INTERFACE_LIBRARIES
LINK_INTERFACE_LIBRARIES_<CONFIG>
LINK_INTERFACE_MULTIPLICITY
LINK_INTERFACE_MULTIPLICITY_<CONFIG>
LINK_LIBRARIES
LINK_LIBRARIES_ONLY_TARGETS
LINK_OPTIONS
LINK_SEARCH_END_STATIC
LINK_SEARCH_START_STATIC
LINK_WHAT_YOU_USE
LINKER_LANGUAGE
LOCATION
LOCATION_<CONFIG>
MACHO_COMPATIBILITY_VERSION
MACHO_CURRENT_VERSION
MACOSX_BUNDLE
MACOSX_BUNDLE_INFO_PLIST
MACOSX_FRAMEWORK_INFO_PLIST
MACOSX_RPATH
MANUALLY_ADDED_DEPENDENCIES
MAP_IMPORTED_CONFIG_<CONFIG>
MSVC_RUNTIME_LIBRARY
NAME
NO_SONAME
NO_SYSTEM_FROM_IMPORTED
OBJC_EXTENSIONS
OBJC_STANDARD
OBJC_STANDARD_REQUIRED
OBJCXX_EXTENSIONS
OBJCXX_STANDARD
OBJCXX_STANDARD_REQUIRED
OPTIMIZE_DEPENDENCIES
OSX_ARCHITECTURES
OSX_ARCHITECTURES_<CONFIG>
OUTPUT_NAME
OUTPUT_NAME_<CONFIG>
PCH_WARN_INVALID
PCH_INSTANTIATE_TEMPLATES
PDB_NAME
PDB_NAME_<CONFIG>
PDB_OUTPUT_DIRECTORY
PDB_OUTPUT_DIRECTORY_<CONFIG>
POSITION_INDEPENDENT_CODE
PRECOMPILE_HEADERS
PRECOMPILE_HEADERS_REUSE_FROM
PREFIX
PRIVATE_HEADER
PROJECT_LABEL
PUBLIC_HEADER
RESOURCE
RULE_LAUNCH_COMPILE
RULE_LAUNCH_CUSTOM
RULE_LAUNCH_LINK
RUNTIME_OUTPUT_DIRECTORY
RUNTIME_OUTPUT_DIRECTORY_<CONFIG>
RUNTIME_OUTPUT_NAME
RUNTIME_OUTPUT_NAME_<CONFIG>
SKIP_BUILD_RPATH
SOURCE_DIR
SOURCES
SOVERSION
STATIC_LIBRARY_FLAGS
STATIC_LIBRARY_FLAGS_<CONFIG>
STATIC_LIBRARY_OPTIONS
SUFFIX
Swift_DEPENDENCIES_FILE
Swift_LANGUAGE_VERSION
Swift_MODULE_DIRECTORY
Swift_MODULE_NAME
TYPE
UNITY_BUILD
UNITY_BUILD_BATCH_SIZE
UNITY_BUILD_CODE_AFTER_INCLUDE
UNITY_BUILD_CODE_BEFORE_INCLUDE
UNITY_BUILD_MODE
UNITY_BUILD_UNIQUE_ID
VERSION
VISIBILITY_INLINES_HIDDEN
VS_CONFIGURATION_TYPE
VS_DEBUGGER_COMMAND
VS_DEBUGGER_COMMAND_ARGUMENTS
VS_DEBUGGER_ENVIRONMENT
VS_DEBUGGER_WORKING_DIRECTORY
VS_DESKTOP_EXTENSIONS_VERSION
VS_DOTNET_DOCUMENTATION_FILE
VS_DOTNET_REFERENCE_<refname>
VS_DOTNET_REFERENCEPROP_<refname>_TAG_<tagname>
VS_DOTNET_REFERENCES
VS_DOTNET_REFERENCES_COPY_LOCAL
VS_DOTNET_TARGET_FRAMEWORK_VERSION
VS_DPI_AWARE
VS_GLOBAL_KEYWORD
VS_GLOBAL_PROJECT_TYPES
VS_GLOBAL_ROOTNAMESPACE
VS_GLOBAL_<variable>
VS_IOT_EXTENSIONS_VERSION
VS_IOT_STARTUP_TASK
VS_JUST_MY_CODE_DEBUGGING
VS_KEYWORD
VS_MOBILE_EXTENSIONS_VERSION
VS_NO_SOLUTION_DEPLOY
VS_PACKAGE_REFERENCES
VS_PLATFORM_TOOLSET
VS_PROJECT_IMPORT
VS_SCC_AUXPATH
VS_SCC_LOCALPATH
VS_SCC_PROJECTNAME
VS_SCC_PROVIDER
VS_SDK_REFERENCES
VS_SOLUTION_DEPLOY
VS_SOURCE_SETTINGS_<tool>
VS_USER_PROPS
VS_WINDOWS_TARGET_PLATFORM_MIN_VERSION
VS_WINRT_COMPONENT
VS_WINRT_EXTENSIONS
VS_WINRT_REFERENCES
WIN32_EXECUTABLE
WINDOWS_EXPORT_ALL_SYMBOLS
XCODE_ATTRIBUTE_<an-attribute>
XCODE_EMBED_FRAMEWORKS_CODE_SIGN_ON_COPY
XCODE_EMBED_FRAMEWORKS_REMOVE_HEADERS_ON_COPY
XCODE_EMBED_<type>
XCODE_EMBED_<type>_CODE_SIGN_ON_COPY
XCODE_EMBED_<type>_PATH
XCODE_EMBED_<type>_REMOVE_HEADERS_ON_COPY
XCODE_EXPLICIT_FILE_TYPE
XCODE_GENERATE_SCHEME
XCODE_LINK_BUILD_PHASE_MODE
XCODE_PRODUCT_TYPE
XCODE_SCHEME_ADDRESS_SANITIZER
XCODE_SCHEME_ADDRESS_SANITIZER_USE_AFTER_RETURN
XCODE_SCHEME_ARGUMENTS
XCODE_SCHEME_DEBUG_AS_ROOT
XCODE_SCHEME_DEBUG_DOCUMENT_VERSIONING
XCODE_SCHEME_ENABLE_GPU_FRAME_CAPTURE_MODE
XCODE_SCHEME_DISABLE_MAIN_THREAD_CHECKER
XCODE_SCHEME_DYNAMIC_LIBRARY_LOADS
XCODE_SCHEME_DYNAMIC_LINKER_API_USAGE
XCODE_SCHEME_ENVIRONMENT
XCODE_SCHEME_EXECUTABLE
XCODE_SCHEME_GUARD_MALLOC
XCODE_SCHEME_MAIN_THREAD_CHECKER_STOP
XCODE_SCHEME_MALLOC_GUARD_EDGES
XCODE_SCHEME_MALLOC_SCRIBBLE
XCODE_SCHEME_MALLOC_STACK
XCODE_SCHEME_THREAD_SANITIZER
XCODE_SCHEME_THREAD_SANITIZER_STOP
XCODE_SCHEME_UNDEFINED_BEHAVIOUR_SANITIZER
XCODE_SCHEME_UNDEFINED_BEHAVIOUR_SANITIZER_STOP
XCODE_SCHEME_WORKING_DIRECTORY
XCODE_SCHEME_ZOMBIE_OBJECTS
XCTEST
```

ADDITIONAL_CLEAN_FILES

AIX_EXPORT_ALL_SYMBOLS

ALIAS_GLOBAL

ALIASED_TARGET

ANDROID_ANT_ADDITIONAL_OPTIONS

ANDROID_API

ANDROID_API_MIN

ANDROID_ARCH

ANDROID_ASSETS_DIRECTORIES

ANDROID_GUI

ANDROID_JAR_DEPENDENCIES

ANDROID_JAR_DIRECTORIES

ANDROID_JAVA_SOURCE_DIR

ANDROID_NATIVE_LIB_DEPENDENCIES

ANDROID_NATIVE_LIB_DIRECTORIES

ANDROID_PROCESS_MAX

ANDROID_PROGUARD

ANDROID_PROGUARD_CONFIG_PATH

ANDROID_SECURE_PROPS_PATH

ANDROID_SKIP_ANT_STEP

ANDROID_STL_TYPE

ARCHIVE_OUTPUT_DIRECTORY

ARCHIVE_OUTPUT_DIRECTORY_<CONFIG>

ARCHIVE_OUTPUT_NAME

ARCHIVE_OUTPUT_NAME_<CONFIG>

AUTOGEN_BUILD_DIR

AUTOGEN_ORIGIN_DEPENDS

AUTOGEN_PARALLEL

AUTOGEN_TARGET_DEPENDS

AUTOMOC

AUTOMOC_COMPILER_PREDEFINES

AUTOMOC_DEPEND_FILTERS

AUTOMOC_EXECUTABLE

AUTOMOC_MACRO_NAMES

AUTOMOC_MOC_OPTIONS

AUTOMOC_PATH_PREFIX

AUTORCC

AUTORCC_EXECUTABLE

AUTORCC_OPTIONS

AUTOUIC

AUTOUIC_EXECUTABLE

AUTOUIC_OPTIONS

AUTOUIC_SEARCH_PATHS

BINARY_DIR

BUILD_RPATH

BUILD_RPATH_USE_ORIGIN

BUILD_WITH_INSTALL_NAME_DIR

BUILD_WITH_INSTALL_RPATH

BUNDLE

BUNDLE_EXTENSION

C_EXTENSIONS

C_STANDARD

C_STANDARD_REQUIRED

COMMON_LANGUAGE_RUNTIME

COMPATIBLE_INTERFACE_BOOL

COMPATIBLE_INTERFACE_NUMBER_MAX

COMPATIBLE_INTERFACE_NUMBER_MIN

COMPATIBLE_INTERFACE_STRING

COMPILE_DEFINITIONS

COMPILE_FEATURES

COMPILE_FLAGS

COMPILE_OPTIONS

COMPILE_PDB_NAME

COMPILE_PDB_NAME_<CONFIG>

COMPILE_PDB_OUTPUT_DIRECTORY

COMPILE_PDB_OUTPUT_DIRECTORY_<CONFIG>

<CONFIG>_OUTPUT_NAME

<CONFIG>_POSTFIX

CROSSCOMPILING_EMULATOR

CUDA_ARCHITECTURES

CUDA_EXTENSIONS

CUDA_PTX_COMPILATION

CUDA_RESOLVE_DEVICE_SYMBOLS

CUDA_RUNTIME_LIBRARY

CUDA_SEPARABLE_COMPILATION

CUDA_STANDARD

CUDA_STANDARD_REQUIRED

CXX_EXTENSIONS

CXX_STANDARD

CXX_STANDARD_REQUIRED

DEBUG_POSTFIX

DEFINE_SYMBOL

DEPLOYMENT_ADDITIONAL_FILES

DEPLOYMENT_REMOTE_DIRECTORY

DEPRECATION

DISABLE_PRECOMPILE_HEADERS

DOTNET_SDK

DOTNET_TARGET_FRAMEWORK

DOTNET_TARGET_FRAMEWORK_VERSION

EchoString

ENABLE_EXPORTS

EXCLUDE_FROM_ALL

EXCLUDE_FROM_DEFAULT_BUILD

EXCLUDE_FROM_DEFAULT_BUILD_<CONFIG>

EXPORT_COMPILE_COMMANDS

EXPORT_NAME

EXPORT_PROPERTIES

FOLDER

Fortran_BUILDING_INSTRINSIC_MODULES

Fortran_FORMAT

Fortran_MODULE_DIRECTORY

Fortran_PREPROCESS

FRAMEWORK

FRAMEWORK_MULTI_CONFIG_POSTFIX_<CONFIG>

FRAMEWORK_VERSION

GENERATOR_FILE_NAME

GHS_INTEGRITY_APP

GHS_NO_SOURCE_GROUP_FILE

GNUtoMS

HAS_CXX

HEADER_DIRS

HEADER_DIRS_<NAME>

HEADER_SET

HEADER_SET_<NAME>

HEADER_SETS

HIP_ARCHITECTURES

HIP_EXTENSIONS

HIP_STANDARD

HIP_STANDARD_REQUIRED

IMPLICIT_DEPENDS_INCLUDE_TRANSFORM

IMPORTED

IMPORTED_COMMON_LANGUAGE_RUNTIME

IMPORTED_CONFIGURATIONS

IMPORTED_GLOBAL

IMPORTED_IMPLIB

IMPORTED_IMPLIB_<CONFIG>

IMPORTED_LIBNAME

IMPORTED_LIBNAME_<CONFIG>

IMPORTED_LINK_DEPENDENT_LIBRARIES

IMPORTED_LINK_DEPENDENT_LIBRARIES_<CONFIG>

IMPORTED_LINK_INTERFACE_LANGUAGES

IMPORTED_LINK_INTERFACE_LANGUAGES_<CONFIG>

IMPORTED_LINK_INTERFACE_LIBRARIES

IMPORTED_LINK_INTERFACE_LIBRARIES_<CONFIG>

IMPORTED_LINK_INTERFACE_MULTIPLICITY

IMPORTED_LINK_INTERFACE_MULTIPLICITY_<CONFIG>

IMPORTED_LOCATION

IMPORTED_LOCATION_<CONFIG>

IMPORTED_NO_SONAME

IMPORTED_NO_SONAME_<CONFIG>

IMPORTED_NO_SYSTEM

IMPORTED_OBJECTS

IMPORTED_OBJECTS_<CONFIG>

IMPORTED_SONAME

IMPORTED_SONAME_<CONFIG>

IMPORT_PREFIX

IMPORT_SUFFIX

INCLUDE_DIRECTORIES

INSTALL_NAME_DIR

INSTALL_REMOVE_ENVIRONMENT_RPATH

INSTALL_RPATH

INSTALL_RPATH_USE_LINK_PATH

INTERFACE_AUTOUIC_OPTIONS

INTERFACE_COMPILE_DEFINITIONS

INTERFACE_COMPILE_FEATURES

INTERFACE_COMPILE_OPTIONS

INTERFACE_HEADER_SETS

INTERFACE_INCLUDE_DIRECTORIES

INTERFACE_LINK_DEPENDS

INTERFACE_LINK_DIRECTORIES

INTERFACE_LINK_LIBRARIES

INTERFACE_LINK_OPTIONS

INTERFACE_POSITION_INDEPENDENT_CODE

INTERFACE_PRECOMPILE_HEADERS

INTERFACE_SOURCES

INTERFACE_SYSTEM_INCLUDE_DIRECTORIES

INTERPROCEDURAL_OPTIMIZATION

INTERPROCEDURAL_OPTIMIZATION_<CONFIG>

IOS_INSTALL_COMBINED

ISPC_HEADER_DIRECTORY

ISPC_HEADER_SUFFIX

ISPC_INSTRUCTION_SETS

JOB_POOL_COMPILE

JOB_POOL_LINK

JOB_POOL_PRECOMPILE_HEADER

LABELS

<LANG>_CLANG_TIDY

<LANG>_COMPILER_LAUNCHER

<LANG>_CPPCHECK

<LANG>_CPPLINT

<LANG>_EXTENSIONS

<LANG>_INCLUDE_WHAT_YOU_USE

<LANG>_LINKER_LAUNCHER

<LANG>_STANDARD

<LANG>_STANDARD_REQUIRED

<LANG>_VISIBILITY_PRESET

LIBRARY_OUTPUT_DIRECTORY

LIBRARY_OUTPUT_DIRECTORY_<CONFIG>

LIBRARY_OUTPUT_NAME

LIBRARY_OUTPUT_NAME_<CONFIG>

LINK_DEPENDS

LINK_DEPENDS_NO_SHARED

LINK_DIRECTORIES

LINK_FLAGS

LINK_FLAGS_<CONFIG>

LINK_INTERFACE_LIBRARIES

LINK_INTERFACE_LIBRARIES_<CONFIG>

LINK_INTERFACE_MULTIPLICITY

LINK_INTERFACE_MULTIPLICITY_<CONFIG>

LINK_LIBRARIES

LINK_LIBRARIES_ONLY_TARGETS

LINK_OPTIONS

LINK_SEARCH_END_STATIC

LINK_SEARCH_START_STATIC

LINK_WHAT_YOU_USE

LINKER_LANGUAGE

LOCATION

LOCATION_<CONFIG>

MACHO_COMPATIBILITY_VERSION

MACHO_CURRENT_VERSION

MACOSX_BUNDLE

MACOSX_BUNDLE_INFO_PLIST

MACOSX_FRAMEWORK_INFO_PLIST

MACOSX_RPATH

MANUALLY_ADDED_DEPENDENCIES

MAP_IMPORTED_CONFIG_<CONFIG>

MSVC_RUNTIME_LIBRARY

NAME

NO_SONAME

NO_SYSTEM_FROM_IMPORTED

OBJC_EXTENSIONS

OBJC_STANDARD

OBJC_STANDARD_REQUIRED

OBJCXX_EXTENSIONS

OBJCXX_STANDARD

OBJCXX_STANDARD_REQUIRED

OPTIMIZE_DEPENDENCIES

OSX_ARCHITECTURES

OSX_ARCHITECTURES_<CONFIG>

OUTPUT_NAME

OUTPUT_NAME_<CONFIG>

PCH_WARN_INVALID

PCH_INSTANTIATE_TEMPLATES

PDB_NAME

PDB_NAME_<CONFIG>

PDB_OUTPUT_DIRECTORY

PDB_OUTPUT_DIRECTORY_<CONFIG>

POSITION_INDEPENDENT_CODE

PRECOMPILE_HEADERS

PRECOMPILE_HEADERS_REUSE_FROM

PREFIX

PRIVATE_HEADER

PROJECT_LABEL

PUBLIC_HEADER

RESOURCE

RULE_LAUNCH_COMPILE

RULE_LAUNCH_CUSTOM

RULE_LAUNCH_LINK

RUNTIME_OUTPUT_DIRECTORY

RUNTIME_OUTPUT_DIRECTORY_<CONFIG>

RUNTIME_OUTPUT_NAME

RUNTIME_OUTPUT_NAME_<CONFIG>

SKIP_BUILD_RPATH

SOURCE_DIR

SOURCES

SOVERSION

STATIC_LIBRARY_FLAGS

STATIC_LIBRARY_FLAGS_<CONFIG>

STATIC_LIBRARY_OPTIONS

SUFFIX

Swift_DEPENDENCIES_FILE

Swift_LANGUAGE_VERSION

Swift_MODULE_DIRECTORY

Swift_MODULE_NAME

TYPE

UNITY_BUILD

UNITY_BUILD_BATCH_SIZE

UNITY_BUILD_CODE_AFTER_INCLUDE

UNITY_BUILD_CODE_BEFORE_INCLUDE

UNITY_BUILD_MODE

UNITY_BUILD_UNIQUE_ID

VERSION

VISIBILITY_INLINES_HIDDEN

VS_CONFIGURATION_TYPE

VS_DEBUGGER_COMMAND

VS_DEBUGGER_COMMAND_ARGUMENTS

VS_DEBUGGER_ENVIRONMENT

VS_DEBUGGER_WORKING_DIRECTORY

VS_DESKTOP_EXTENSIONS_VERSION

VS_DOTNET_DOCUMENTATION_FILE

VS_DOTNET_REFERENCE_<refname>

VS_DOTNET_REFERENCEPROP_<refname>_TAG_<tagname>

VS_DOTNET_REFERENCES

VS_DOTNET_REFERENCES_COPY_LOCAL

VS_DOTNET_TARGET_FRAMEWORK_VERSION

VS_DPI_AWARE

VS_GLOBAL_KEYWORD

VS_GLOBAL_PROJECT_TYPES

VS_GLOBAL_ROOTNAMESPACE

VS_GLOBAL_<variable>

VS_IOT_EXTENSIONS_VERSION

VS_IOT_STARTUP_TASK

VS_JUST_MY_CODE_DEBUGGING

VS_KEYWORD

VS_MOBILE_EXTENSIONS_VERSION

VS_NO_SOLUTION_DEPLOY

VS_PACKAGE_REFERENCES

VS_PLATFORM_TOOLSET

VS_PROJECT_IMPORT

VS_SCC_AUXPATH

VS_SCC_LOCALPATH

VS_SCC_PROJECTNAME

VS_SCC_PROVIDER

VS_SDK_REFERENCES

VS_SOLUTION_DEPLOY

VS_SOURCE_SETTINGS_<tool>

VS_USER_PROPS

VS_WINDOWS_TARGET_PLATFORM_MIN_VERSION

VS_WINRT_COMPONENT

VS_WINRT_EXTENSIONS

VS_WINRT_REFERENCES

WIN32_EXECUTABLE

WINDOWS_EXPORT_ALL_SYMBOLS

XCODE_ATTRIBUTE_<an-attribute>

XCODE_EMBED_FRAMEWORKS_CODE_SIGN_ON_COPY

XCODE_EMBED_FRAMEWORKS_REMOVE_HEADERS_ON_COPY

XCODE_EMBED_<type>

XCODE_EMBED_<type>_CODE_SIGN_ON_COPY

XCODE_EMBED_<type>_PATH

XCODE_EMBED_<type>_REMOVE_HEADERS_ON_COPY

XCODE_EXPLICIT_FILE_TYPE

XCODE_GENERATE_SCHEME

XCODE_LINK_BUILD_PHASE_MODE

XCODE_PRODUCT_TYPE

XCODE_SCHEME_ADDRESS_SANITIZER

XCODE_SCHEME_ADDRESS_SANITIZER_USE_AFTER_RETURN

XCODE_SCHEME_ARGUMENTS

XCODE_SCHEME_DEBUG_AS_ROOT

XCODE_SCHEME_DEBUG_DOCUMENT_VERSIONING

XCODE_SCHEME_ENABLE_GPU_FRAME_CAPTURE_MODE

XCODE_SCHEME_DISABLE_MAIN_THREAD_CHECKER

XCODE_SCHEME_DYNAMIC_LIBRARY_LOADS

XCODE_SCHEME_DYNAMIC_LINKER_API_USAGE

XCODE_SCHEME_ENVIRONMENT

XCODE_SCHEME_EXECUTABLE

XCODE_SCHEME_GUARD_MALLOC

XCODE_SCHEME_MAIN_THREAD_CHECKER_STOP

XCODE_SCHEME_MALLOC_GUARD_EDGES

XCODE_SCHEME_MALLOC_SCRIBBLE

XCODE_SCHEME_MALLOC_STACK

XCODE_SCHEME_THREAD_SANITIZER

XCODE_SCHEME_THREAD_SANITIZER_STOP

XCODE_SCHEME_UNDEFINED_BEHAVIOUR_SANITIZER

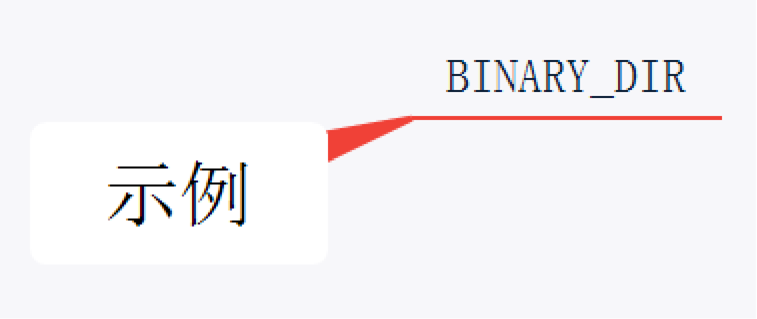
XCODE_SCHEME_UNDEFINED_BEHAVIOUR_SANITIZER_STOP

XCODE_SCHEME_WORKING_DIRECTORY

XCODE_SCHEME_ZOMBIE_OBJECTS

XCTEST

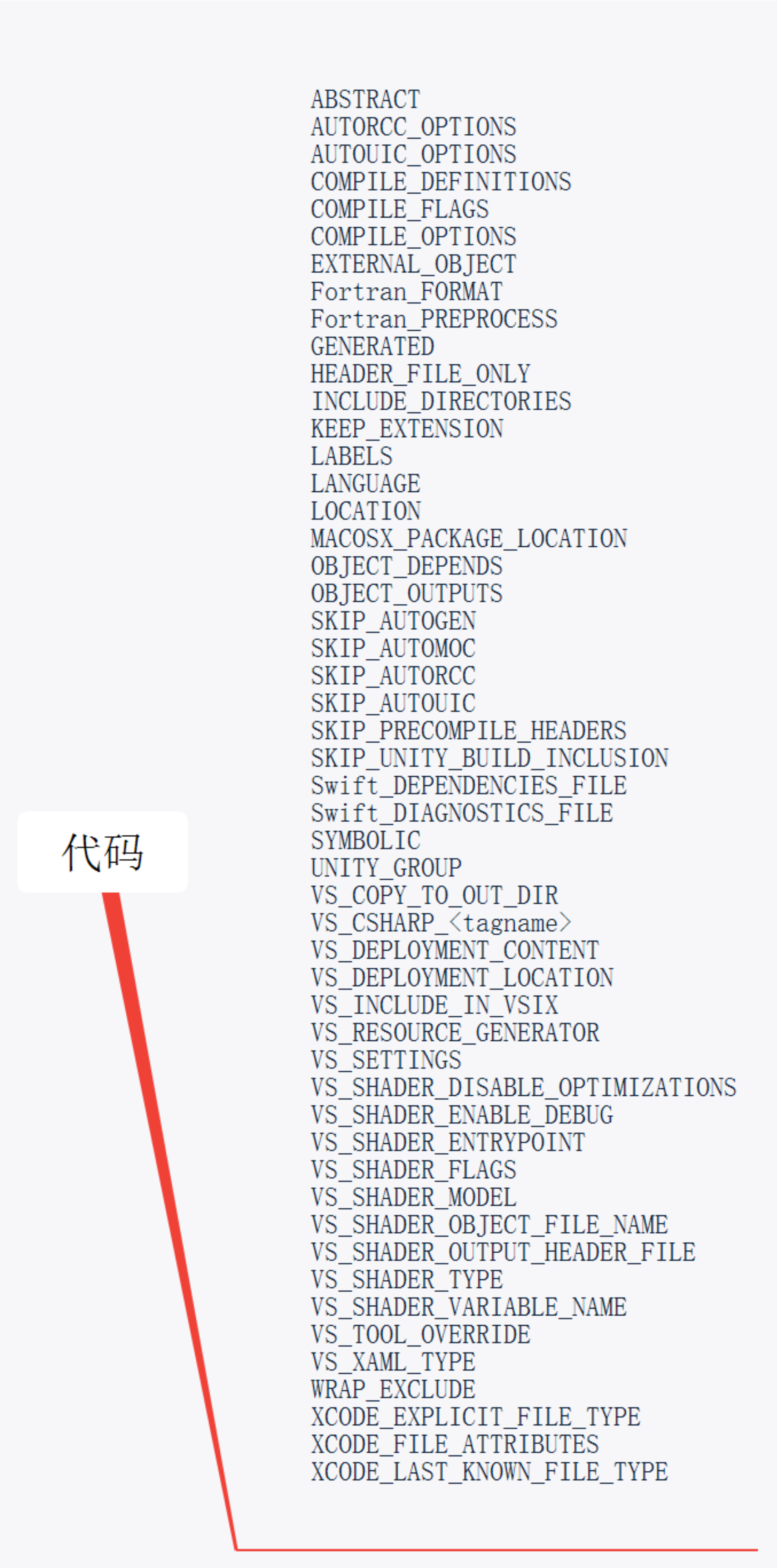
示例



BINARY_DIR

3.5.4. 源码属性

代码



```
ABSTRACT
AUTORCC_OPTIONS
AUTOUIC_OPTIONS
COMPILE_DEFINITIONS
COMPILE_FLAGS
COMPILE_OPTIONS
EXTERNAL_OBJECT
Fortran_FORMAT
Fortran_PREPROCESS
GENERATED
HEADER_FILE_ONLY
INCLUDE_DIRECTORIES
KEEP_EXTENSION
LABELS
LANGUAGE
LOCATION
MACOSX_PACKAGE_LOCATION
OBJECT_DEPENDS
OBJECT_OUTPUTS
SKIP_AUTOGEN
SKIP_AUTOMOC
SKIP_AUTORCC
SKIP_AUTOUIC
SKIP_PRECOMPILE_HEADERS
SKIP_UNITY_BUILD_INCLUSION
Swift_DEPENDENCIES_FILE
Swift_DIAGNOSTICS_FILE
SYMBOLIC
UNITY_GROUP
VS_COPY_TO_OUT_DIR
VS_CSHARP_<tagname>
VS_DEPLOYMENT_CONTENT
VS_DEPLOYMENT_LOCATION
VS_INCLUDE_IN_VSIX
VS_RESOURCE_GENERATOR
VS_SETTINGS
VS_SHADER_DISABLE_OPTIMIZATIONS
VS_SHADER_ENABLE_DEBUG
VS_SHADER_ENTRYPOINT
VS_SHADER_FLAGS
VS_SHADER_MODEL
VS_SHADER_OBJECT_FILE_NAME
VS_SHADER_OUTPUT_HEADER_FILE
VS_SHADER_TYPE
VS_SHADER_VARIABLE_NAME
VS_TOOL_OVERRIDE
VS_XAML_TYPE
WRAP_EXCLUDE
XCODE_EXPLICIT_FILE_TYPE
XCODE_FILE_ATTRIBUTES
XCODE_LAST_KNOWN_FILE_TYPE
```

ABSTRACT

AUTORCC_OPTIONS

AUTOUIC_OPTIONS

COMPILE_DEFINITIONS

COMPILE_FLAGS

COMPILE_OPTIONS

EXTERNAL_OBJECT

Fortran_FORMAT

Fortran_PREPROCESS

GENERATED

HEADER_FILE_ONLY

INCLUDE_DIRECTORIES

KEEP_EXTENSION

LABELS

LANGUAGE

LOCATION

MACOSX_PACKAGE_LOCATION

OBJECT_DEPENDS

OBJECT_OUTPUTS

SKIP_AUTOGEN

SKIP_AUTOMOC

SKIP_AUTORCC

SKIP_AUTOUIC

SKIP_PRECOMPILE_HEADERS

SKIP_UNITY_BUILD_INCLUSION

Swift_DEPENDENCIES_FILE

Swift_DIAGNOSTICS_FILE

SYMBOLIC

UNITY_GROUP

VS_COPY_TO_OUT_DIR

VS_CSHARP_<tagname>

VS_DEPLOYMENT_CONTENT

VS_DEPLOYMENT_LOCATION

VS_INCLUDE_IN_VSIX

VS_RESOURCE_GENERATOR

VS_SETTINGS

VS_SHADER_DISABLE_OPTIMIZATIONS

VS_SHADER_ENABLE_DEBUG

VS_SHADER_ENTRYPOINT

VS_SHADER_FLAGS

VS_SHADER_MODEL

VS_SHADER_OBJECT_FILE_NAME

VS_SHADER_OUTPUT_HEADER_FILE

VS_SHADER_TYPE

VS_SHADER_VARIABLE_NAME

VS_TOOL_OVERRIDE

VS_XAML_TYPE

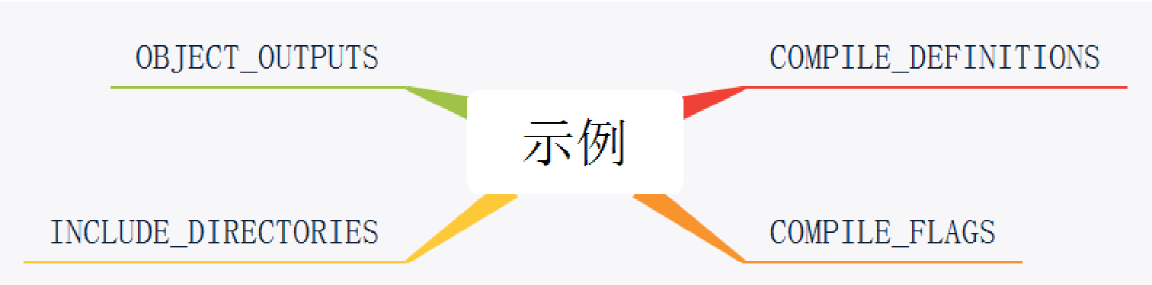
WRAP_EXCLUDE

XCODE_EXPLICIT_FILE_TYPE

XCODE_FILE_ATTRIBUTES

XCODE_LAST_KNOWN_FILE_TYPE

示例



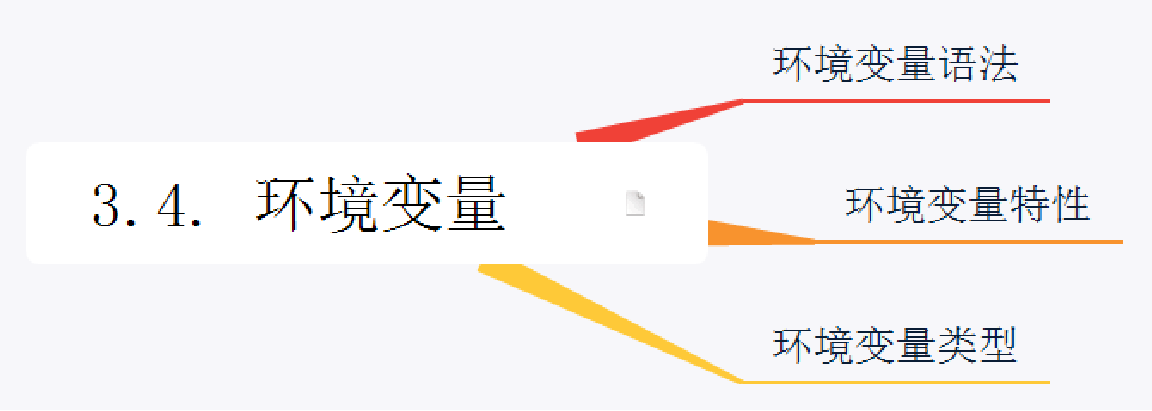
COMPILE_DEFINITIONS

COMPILE_FLAGS

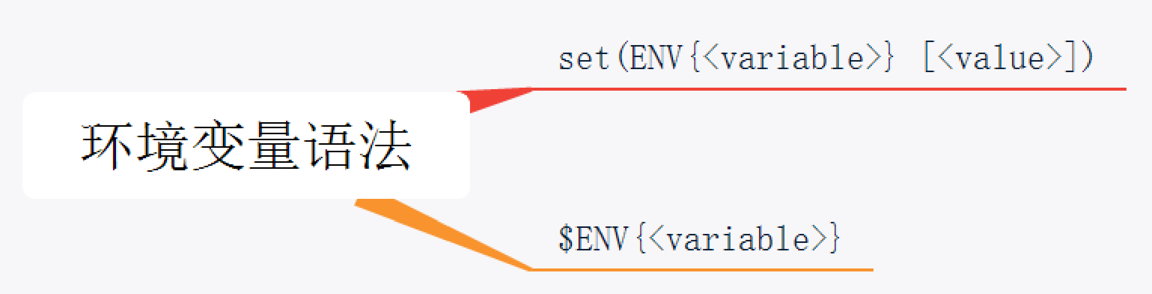
INCLUDE_DIRECTORIES

OBJECT_OUTPUTS

4. 3.4. 环境变量



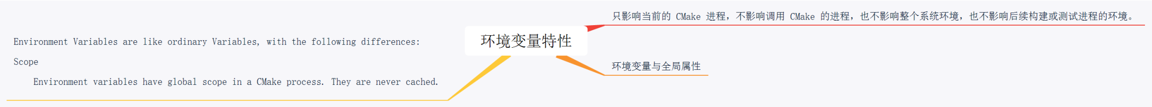
4.1. 环境变量语法



4.1.1. set(ENV{<variable>} [<value>])

4.1.2. $ENV{<variable>}

4.2. 环境变量特性



只影响当前的 CMake 进程，不影响调用 CMake 的进程，也不影响整个系统环境，也不影响后续构建或测试进程的环境。

Environment Variables are like ordinary Variables, with the following differences:

Scope

Environment variables have global scope in a CMake process. They are never cached.

4.2.1. 只影响当前的 CMake 进程，不影响调用 CMake 的进程，也不影响整个系统环境，也不影响后续构建或测试进程的环境。

4.2.2. 环境变量与全局属性

基本类似 全局属性可以加说明

环境变量访问简单

4.2.3. Environment Variables are like ordinary Variables, with the following differences:

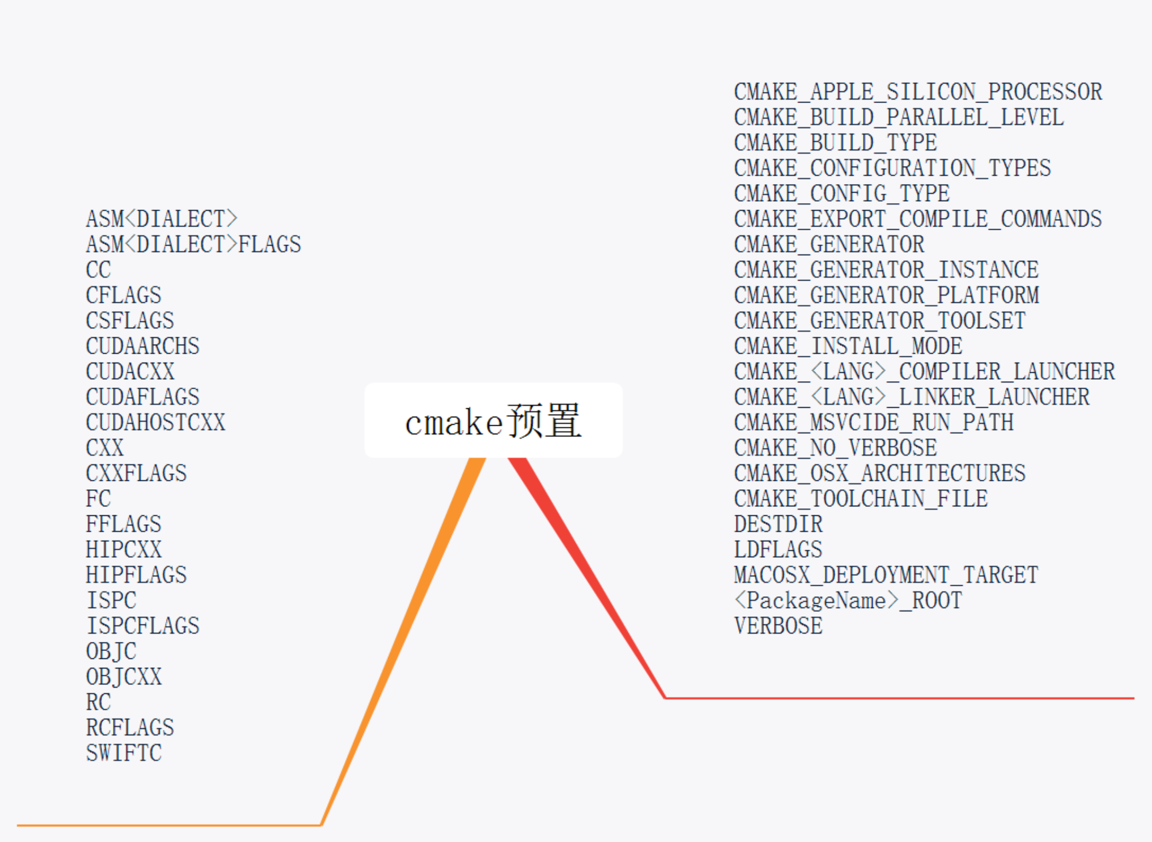
Scope

Environment variables have global scope in a CMake process. They are never cached.

4.3. 环境变量类型

环境变量类型
cmake预置
系统变量
自定义环境变量

4.3.1. cmake预置



CMAKE_APPLE_SILICON_PROCESSOR

CMAKE_BUILD_PARALLEL_LEVEL

CMAKE_BUILD_TYPE

CMAKE_CONFIGURATION_TYPES

CMAKE_CONFIG_TYPE

CMAKE_EXPORT_COMPILE_COMMANDS

CMAKE_GENERATOR

CMAKE_GENERATOR_INSTANCE

CMAKE_GENERATOR_PLATFORM

CMAKE_GENERATOR_TOOLSET

CMAKE_INSTALL_MODE

CMAKE_<LANG>_COMPILER_LAUNCHER

CMAKE_<LANG>_LINKER_LAUNCHER

CMAKE_MSVCIDE_RUN_PATH

CMAKE_NO_VERBOSE

CMAKE_OSX_ARCHITECTURES

CMAKE_TOOLCHAIN_FILE

DESTDIR

LDFLAGS

MACOSX_DEPLOYMENT_TARGET

<PackageName>_ROOT

VERBOSE

ASM<DIALECT>

ASM<DIALECT>FLAGS

CC

CFLAGS

CSFLAGS

CUDAARCHS

CUDACXX

CUDAFLAGS

CUDAHOSTCXX

CXX

CXXFLAGS

FC

FFLAGS

HIPCXX

HIPFLAGS

ISPC

ISPCFLAGS

OBJC

OBJCXX

RC

RCFLAGS

SWIFTC

4.3.2. 自定义环境变量

4.3.3. 系统变量

5. 3.5 cmake math数学运算

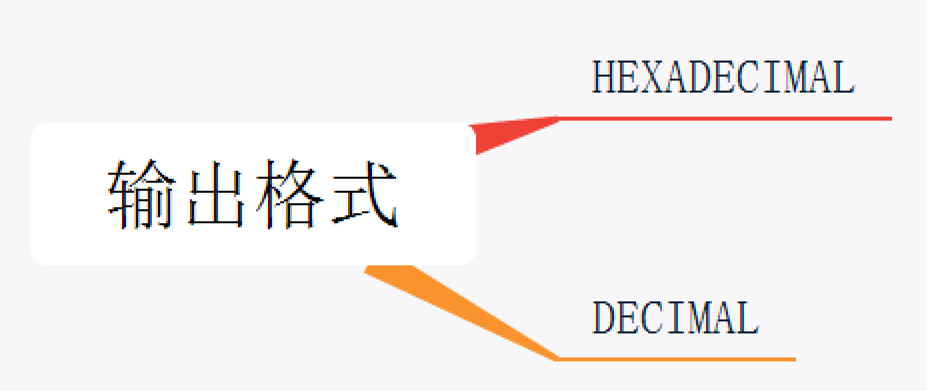
输出格式
math(EXPR <variable> "<expression>" [OUTPUT_FORMAT <format>])
3.5 cmake math数学运算
结果必须是64位有符号整数
"5 * (10 + 13)". 支持 +, -, *, /, %, |, &, ^, ~, <<, >>

5.1. math(EXPR <variable> "<expression>" [OUTPUT_FORMAT <format>])

5.2. "5 * (10 + 13)". 支持 +, -, *, /, %, |, &, ^, ~, <<, >>

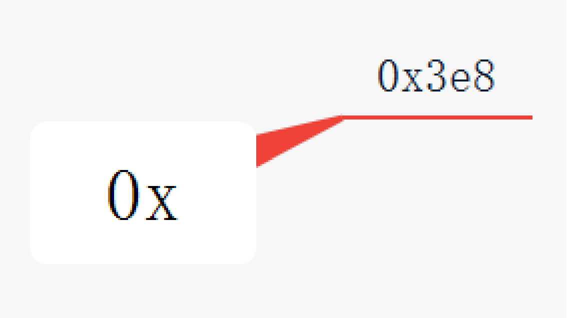
5.3. 结果必须是64位有符号整数

5.4. 输出格式



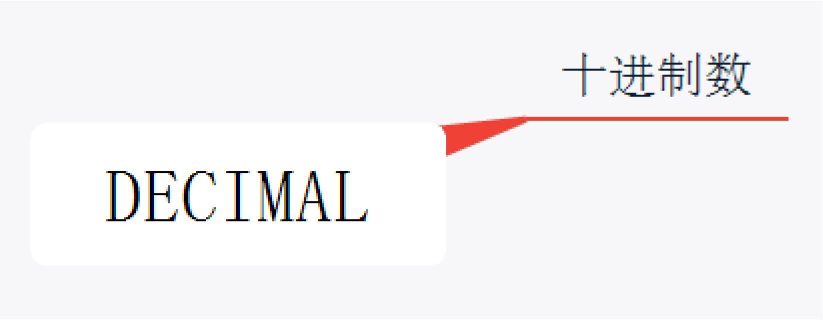
5.4.1. HEXADECIMAL

0x



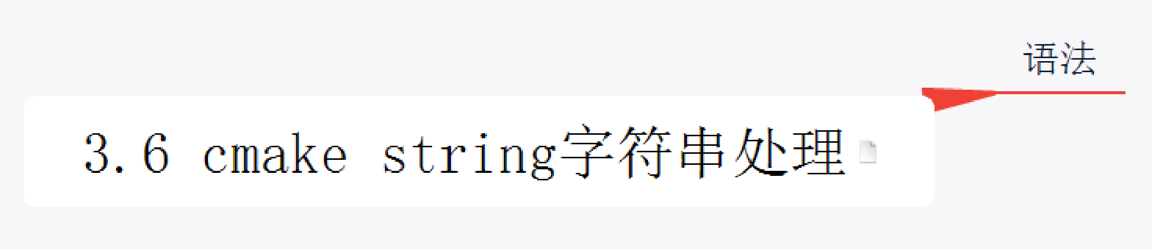
0x3e8

5.4.2. DECIMAL

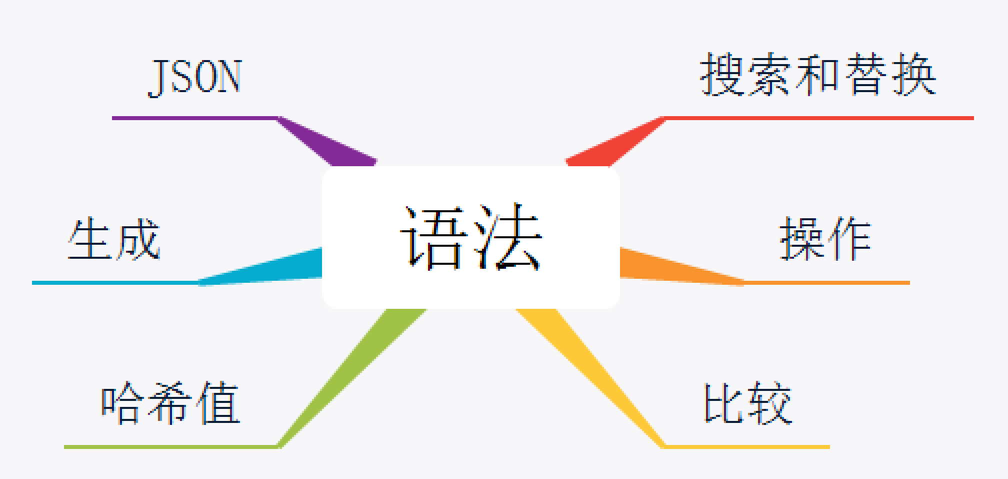


十进制数

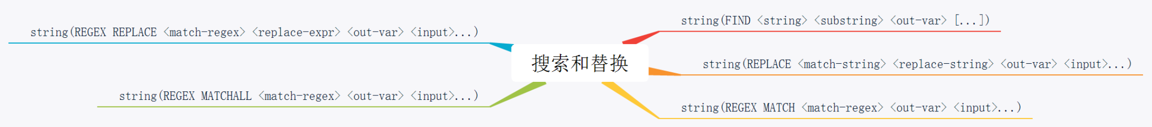
6. 3.6 cmake string字符串处理



6.1. 语法



6.1.1. 搜索和替换


string(REGEX REPLACE <match-regex> <replace-expr> <out-var> <input>...)
string(FIND <string> <substring> <out-var> [...])
搜索和替换
string(REPLACE <match-string> <replace-string> <out-var> <input>...)
string(REGEX MATCHALL <match-regex> <out-var> <input>...)
string(REGEX MATCH <match-regex> <out-var> <input>...)

string(FIND <string> <substring> <out-var> [...])

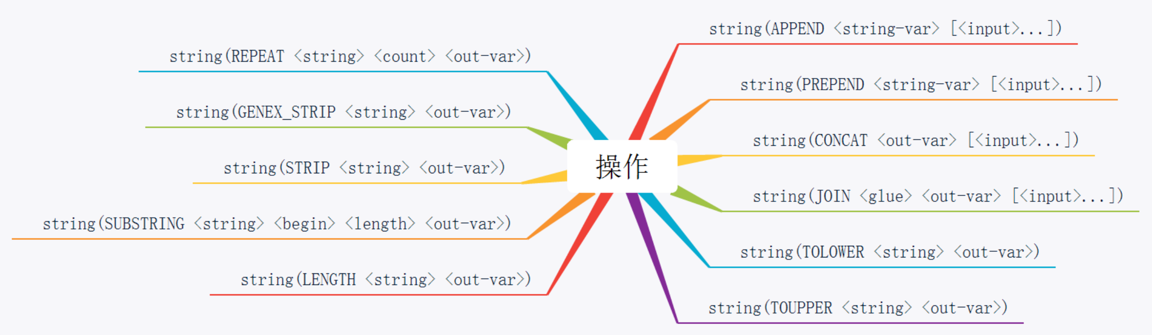
string(REPLACE <match-string> <replace-string> <out-var> <input>...)

string(REGEX MATCH <match-regex> <out-var> <input>...)

string(REGEX MATCHALL <match-regex> <out-var> <input>...)

string(REGEX REPLACE <match-regex> <replace-expr> <out-var> <input>...)

6.1.2. 操作


操作
string(APPEND <string-var> [<input>...])
string(PREPEND <string-var> [<input>...])
string(CONCAT <out-var> [<input>...])
string(JOIN <glue> <out-var> [<input>...])
string(TOLOWER <string> <out-var>)
string(TOUPPER <string> <out-var>)
string(REPEAT <string> <count> <out-var>)
string(GENEX_STRIP <string> <out-var>)
string(STRIP <string> <out-var>)
string(SUBSTRING <string> <begin> <length> <out-var>)
string(LENGTH <string> <out-var>)

string(APPEND <string-var> [<input>...])

string(PREPEND <string-var> [<input>...])

string(CONCAT <out-var> [<input>...])

string(JOIN <glue> <out-var> [<input>...])

string(TOLOWER <string> <out-var>)

string(TOUPPER <string> <out-var>)

string(LENGTH <string> <out-var>)

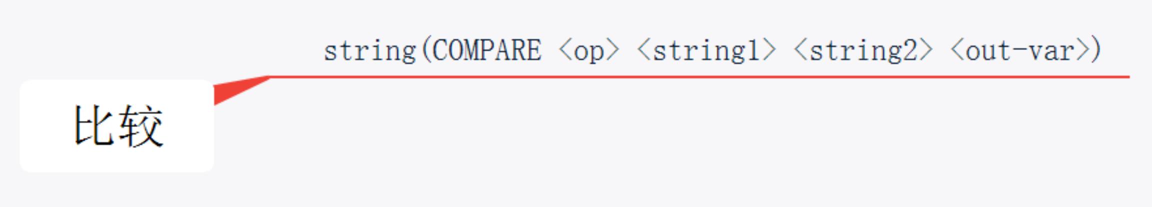
string(SUBSTRING <string> <begin> <length> <out-var>)

string(STRIP <string> <out-var>)

string(GENEX_STRIP <string> <out-var>)

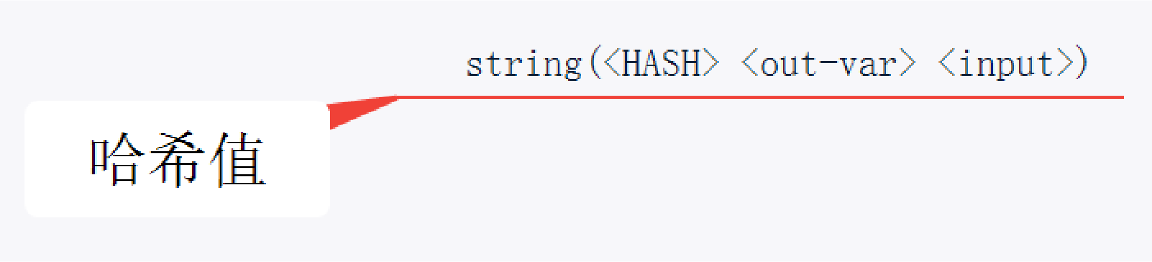
string(REPEAT <string> <count> <out-var>)

6.1.3. 比较



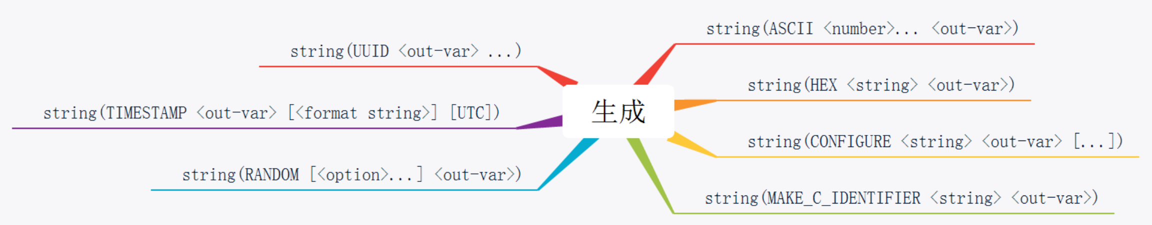
string(COMPARE <op> <string1> <string2> <out-var>)

6.1.4. 哈希值



string(<HASH> <out-var> <input>)

6.1.5. 生成


生成
string(UUID <out-var> ...)
string(TIMESTAMP <out-var> [<format string>] [UTC])
string(RANDOM [<option>...] <out-var>)
string(ASCII <number>... <out-var>)
string(HEX <string> <out-var>)
string(CONFIGURE <string> <out-var> [...])
string(MAKE_C_IDENTIFIER <string> <out-var>)

string(ASCII <number>... <out-var>)

string(HEX <string> <out-var>)

string(CONFIGURE <string> <out-var> [...])

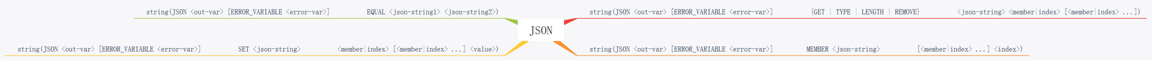
string(MAKE_C_IDENTIFIER <string> <out-var>)

string(RANDOM [<option>...] <out-var>)

string(TIMESTAMP <out-var> [<format string>] [UTC])

string(UUID <out-var> ...)

6.1.6. JSON


JSON
string(JSON <out-var> [ERROR_VARIABLE <error-var>] EQUAL <json-string1> <json-string2>)
string(JSON <out-var> [ERROR_VARIABLE <error-var>] SET <json-string> <member|index> [<member|index> ...] <value>)
string(JSON <out-var> [ERROR_VARIABLE <error-var>] {GET | TYPE | LENGTH | REMOVE} <json-string> <member|index> [<member|index> ...])
string(JSON <out-var> [ERROR_VARIABLE <error-var>] MEMBER <json-string> [<member|index> ...] <index>)

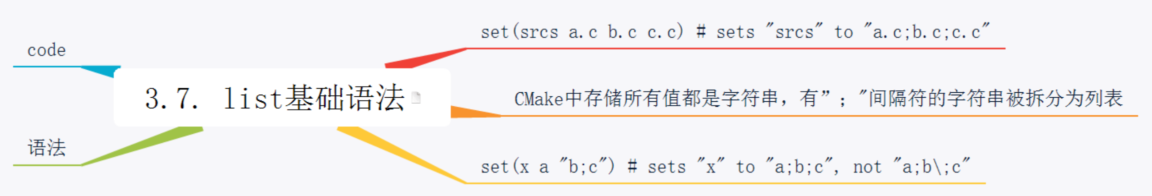
string(JSON <out-var> [ERROR_VARIABLE <error-var>] {GET | TYPE | LENGTH | REMOVE} <json-string> <member|index> [<member|index> ...])

string(JSON <out-var> [ERROR_VARIABLE <error-var>] MEMBER <json-string> [<member|index> ...] <index>)

string(JSON <out-var> [ERROR_VARIABLE <error-var>] SET <json-string> <member|index> [<member|index> ...] <value>)

string(JSON <out-var> [ERROR_VARIABLE <error-var>] EQUAL <json-string1> <json-string2>)

7. 3.7. list基础语法



3.7. list基础语法

code

语法

set(srcs a.c b.c c.c) # sets ″srcs″ to ″a.c;b.c;c.c″

CMake中存储所有值都是字符串，有″；″间隔符的字符串被拆分为列表

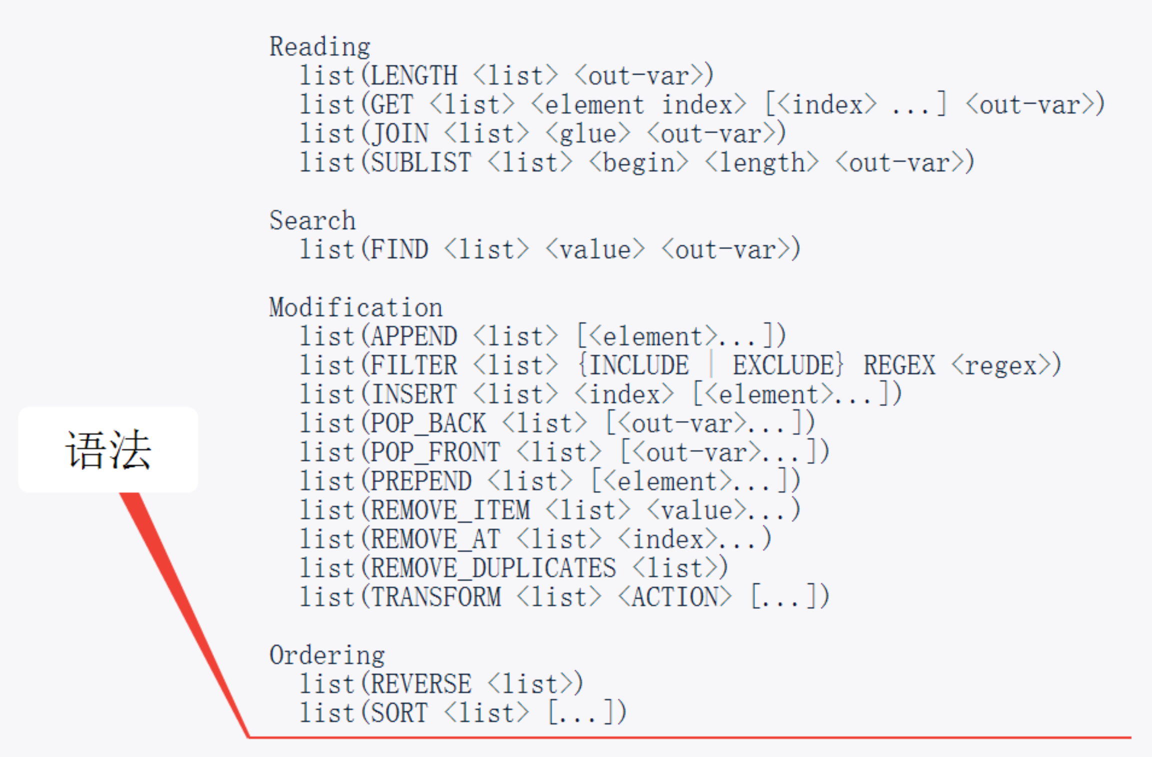
set(x a ″b;c″) # sets ″x″ to ″a;b;c″, not ″a;b\;c″

7.1. set(srcs a.c b.c c.c) # sets "srcs" to "a.c;b.c;c.c"

7.2. CMake中存储所有值都是字符串，有”；"间隔符的字符串被拆分为列表

7.3. set(x a "b;c") # sets "x" to "a;b;c", not "a;b\;c"

7.4. 语法



```
Reading
  list(LENGTH <list> <out-var>)
  list(GET <list> <element index> [<index> ...] <out-var>)
  list(JOIN <list> <glue> <out-var>)
  list(SUBLIST <list> <begin> <length> <out-var>)

Search
  list(FIND <list> <value> <out-var>)

Modification
  list(APPEND <list> [<element>...])
  list(FILTER <list> {INCLUDE | EXCLUDE} REGEX <regex>)
  list(INSERT <list> <index> [<element>...])
  list(POP_BACK <list> [<out-var>...])
  list(POP_FRONT <list> [<out-var>...])
  list(PREPEND <list> [<element>...])
  list(REMOVE_ITEM <list> <value>...)
  list(REMOVE_AT <list> <index>...)
  list(REMOVE_DUPLICATES <list>)
  list(TRANSFORM <list> <ACTION> [...])

Ordering
  list(REVERSE <list>)
  list(SORT <list> [...])
```

7.4.1. Reading

list(LENGTH <list> <out-var>)

list(GET <list> <element index> [<index> ...] <out-var>)

list(JOIN <list> <glue> <out-var>)

list(SUBLIST <list> <begin> <length> <out-var>)

Search

list(FIND <list> <value> <out-var>)

Modification

list(APPEND <list> [<element>...])

list(FILTER <list> {INCLUDE | EXCLUDE} REGEX <regex>)

list(INSERT <list> <index> [<element>...])

list(POP_BACK <list> [<out-var>...])

list(POP_FRONT <list> [<out-var>...])

list(PREPEND <list> [<element>...])

list(REMOVE_ITEM <list> <value>...)

list(REMOVE_AT <list> <index>...)

list(REMOVE_DUPLICATES <list>)

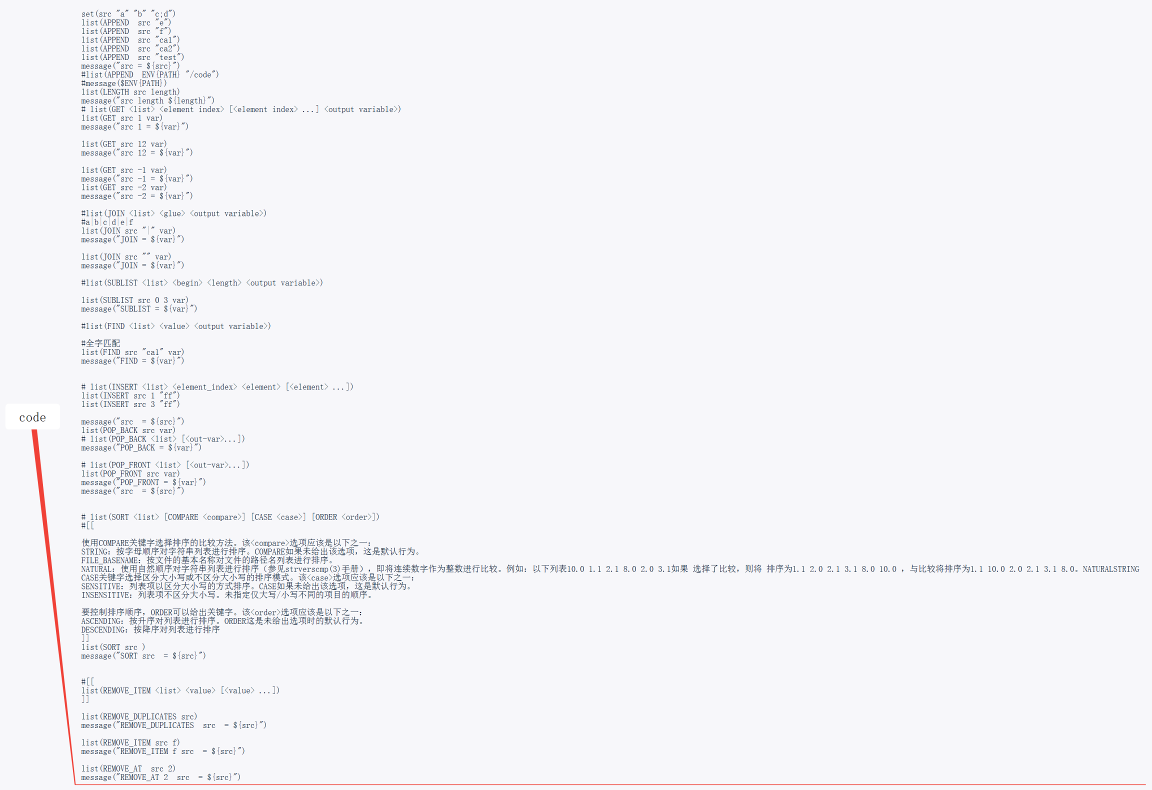
list(TRANSFORM <list> <ACTION> [...])

Ordering

list(REVERSE <list>)

list(SORT <list> [...])

7.5. code



```
set(src "a" "b" "c;d")
list(APPEND  src "e")
list(APPEND  src "f")
list(APPEND  src "ca1")
list(APPEND  src "ca2")
list(APPEND  src "test")
message("src = ${src}")
#list(APPEND  ENV{PATH} "/code")
#message($ENV{PATH})
list(LENGTH src length)
message("src length ${length}")
# list(GET <list> <element index> [<element index> ...] <output variable>)
list(GET src 1 var)
message("src 1 = ${var}")

list(GET src 12 var)
message("src 12 = ${var}")

list(GET src -1 var)
message("src -1 = ${var}")
list(GET src -2 var)
message("src -2 = ${var}")

#list(JOIN <list> <glue> <output variable>)
#a|b|c|d|e|f
list(JOIN src "|" var)
message("JOIN = ${var}")

list(JOIN src "" var)
message("JOIN = ${var}")

#list(SUBLIST <list> <begin> <length> <output variable>)

list(SUBLIST src 0 3 var)
message("SUBLIST = ${var}")

#list(FIND <list> <value> <output variable>)

#全字匹配
list(FIND src "ca1" var)
message("FIND = ${var}")


# list(INSERT <list> <element_index> <element> [<element> ...])
list(INSERT src 1 "ff")
list(INSERT src 3 "ff")

message("src  = ${src}")
list(POP_BACK src var)
# list(POP_BACK <list> [<out-var>...])
message("POP_BACK = ${var}")

# list(POP_FRONT <list> [<out-var>...])
list(POP_FRONT src var)
message("POP_FRONT = ${var}")
message("src  = ${src}")


# list(SORT <list> [COMPARE <compare>] [CASE <case>] [ORDER <order>])
#[[

使用COMPARE关键字选择排序的比较方法。该<compare>选项应该是以下之一：
STRING：按字母顺序对字符串列表进行排序。COMPARE如果未给出该选项，这是默认行为。
FILE_BASENAME：按文件的基本名称对文件的路径名列表进行排序。
NATURAL：使用自然顺序对字符串列表进行排序（参见strverscmp(3)手册），即将连续数字作为整数进行比较。例如：以下列表10.0 1.1 2.1 8.0 2.0 3.1如果 选择了比较，则将 排序为1.1 2.0 2.1 3.1 8.0 10.0 ，与比较将排序为1.1 10.0 2.0 2.1 3.1 8.0。NATURALSTRING
CASE关键字选择区分大小写或不区分大小写的排序模式。该<case>选项应该是以下之一：
SENSITIVE：列表项以区分大小写的方式排序。CASE如果未给出该选项，这是默认行为。
INSENSITIVE：列表项不区分大小写。未指定仅大写/小写不同的项目的顺序。

要控制排序顺序，ORDER可以给出关键字。该<order>选项应该是以下之一：
ASCENDING：按升序对列表进行排序。ORDER这是未给出选项时的默认行为。
DESCENDING：按降序对列表进行排序
]]
list(SORT src )
message("SORT src  = ${src}")


#[[
list(REMOVE_ITEM <list> <value> [<value> ...])
]]

list(REMOVE_DUPLICATES src)
message("REMOVE_DUPLICATES  src  = ${src}")

list(REMOVE_ITEM src f)
message("REMOVE_ITEM f src  = ${src}")

list(REMOVE_AT  src 2)
message("REMOVE_AT 2  src  = ${src}")
```

7.5.1. set(src "a" "b" "c;d")

list(APPEND src "e")

list(APPEND src "f")

list(APPEND src "ca1")

list(APPEND src "ca2")

list(APPEND src "test")

```
message("src = ${src}")
#list(APPEND  ENV{PATH} "/code")
#message($ENV{PATH})
list(LENGTH src length)
message("src length ${length}")
# list(GET <list> <element index> [<element index> ...] <output variable>)
list(GET src 1 var)
message("src 1 = ${var}")

list(GET src 12 var)
message("src 12 = ${var}")

list(GET src -1 var)
message("src -1 = ${var}")
list(GET src -2 var)
message("src -2 = ${var}")

#list(JOIN <list> <glue> <output variable>)
#a|b|c|d|e|f
list(JOIN src "|" var)
message("JOIN = ${var}")

list(JOIN src "" var)
message("JOIN = ${var}")

#list(SUBLIST <list> <begin> <length> <output variable>)

list(SUBLIST src 0 3 var)
message("SUBLIST = ${var}")

#list(FIND <list> <value> <output variable>)
```

```
#全字匹配
list(FIND src "ca1" var)
message("FIND = ${var}")


# list(INSERT <list> <element_index> <element> [<element> ...])
list(INSERT src 1 "ff")
list(INSERT src 3 "ff")

message("src  = ${src}")
list(POP_BACK src var)
# list(POP_BACK <list> [<out-var>...])
message("POP_BACK = ${var}")

# list(POP_FRONT <list> [<out-var>...])
list(POP_FRONT src var)
message("POP_FRONT = ${var}")
message("src  = ${src}")


# list(SORT <list> [COMPARE <compare>] [CASE <case>] [ORDER <order>])
#[[

使用COMPARE关键字选择排序的比较方法。该<compare>选项应该是以下之一：
STRING：按字母顺序对字符串列表进行排序。COMPARE如果未给出该选项，这是默认行为。
FILE_BASENAME：按文件的基本名称对文件的路径名列表进行排序。
NATURAL：使用自然顺序对字符串列表进行排序（参见strverscmp(3)手册），
```

即将连续数字作为整数进行比较。例如：以下列表10.0 1.1 2.1 8.0 2.0 3.1如果选择了比较，则将 排序为1.1 2.0 2.1 3.1 8.0 10.0 ，与比较将排序为1.1 10.0 2.0 2.1 3.1 8.0。NATURALSTRING

CASE关键字选择区分大小写或不区分大小写的排序模式。该<case>选项应该是以下之一：

SENSITIVE：列表项以区分大小写的方式排序。CASE如果未给出该选项，这是默认行为。

INSENSITIVE：列表项不区分大小写。未指定仅大写/小写不同的项目的顺序。

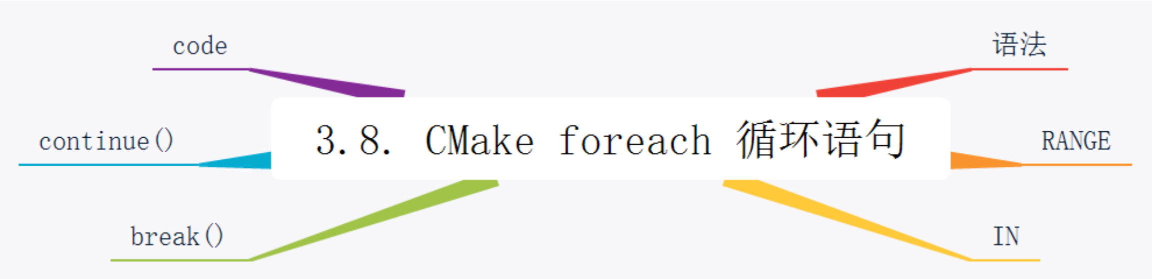
要控制排序顺序，ORDER可以给出关键字。该<order>选项应该是以下之一：

ASCENDING：按升序对列表进行排序。ORDER这是未给出选项时的默认行为。

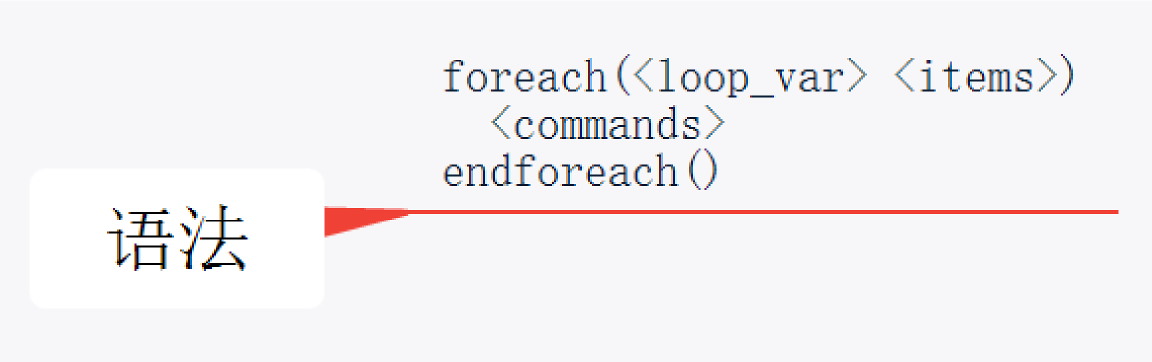
DESCENDING：按降序对列表进行排序

```
]]
list(SORT src )
message("SORT src  = ${src}")



#[[
list(REMOVE_ITEM <list> <value> [<value> ...])
]]

list(REMOVE_DUPLICATES src)
message("REMOVE_DUPLICATES  src  = ${src}")

list(REMOVE_ITEM src f)
message("REMOVE_ITEM f src  = ${src}")

list(REMOVE_AT  src 2)
message("REMOVE_AT 2  src  = ${src}")
```

8. 3.8. CMake foreach 循环语句



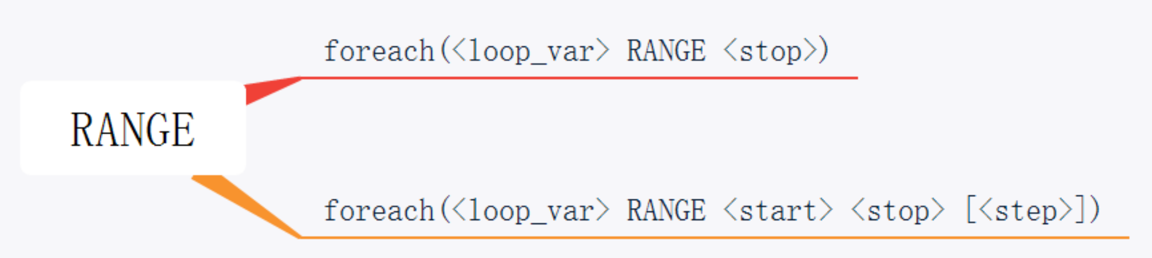
8.1. 语法



```
foreach(<loop_var> <items>)
  <commands>
endforeach()
```

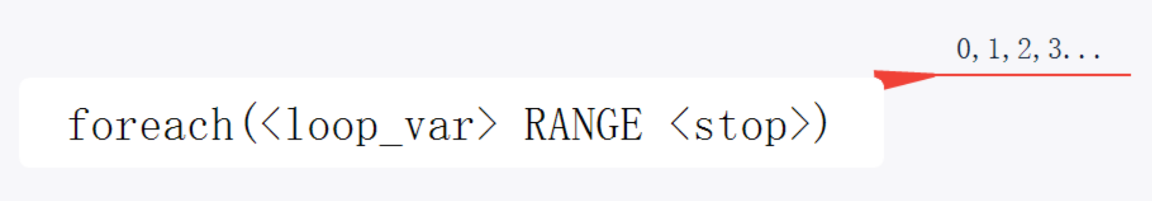
8.1.1. foreach(<loop_var> <items>)
<commands>
endforeach()

8.2. RANGE



```
foreach(<loop_var> RANGE <stop>)
foreach(<loop_var> RANGE <start> <stop> [<step>])
```

8.2.1. foreach(<loop_var> RANGE <stop>)

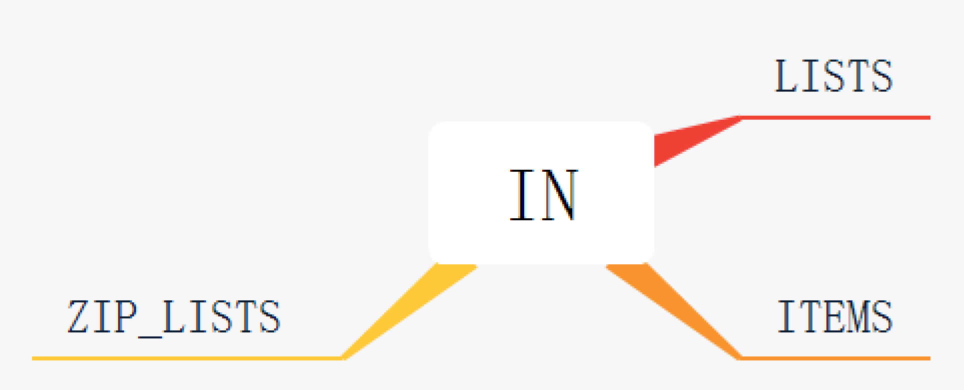


```
foreach(<loop_var> RANGE <stop>)
```

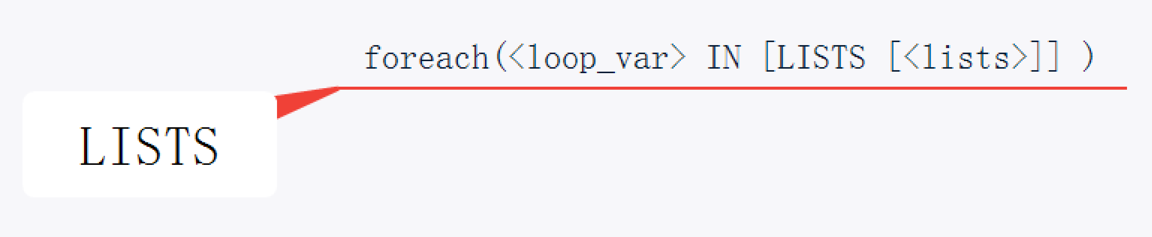
0,1,2,3...

8.2.2. foreach(<loop_var> RANGE <start> <stop> [<step>])

8.3. IN

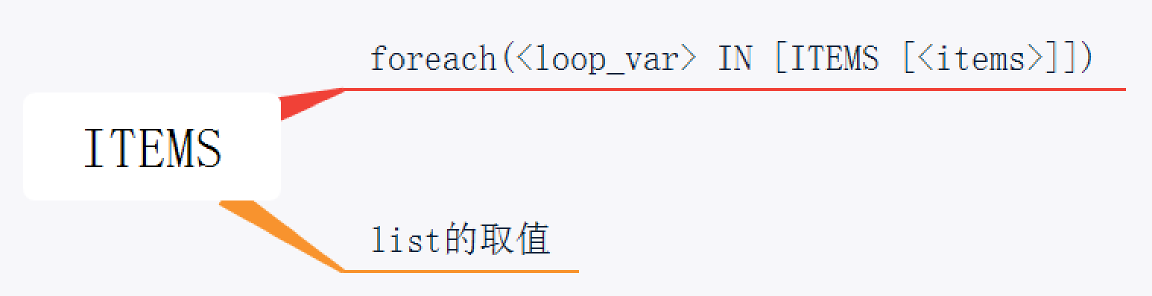


8.3.1. LISTS



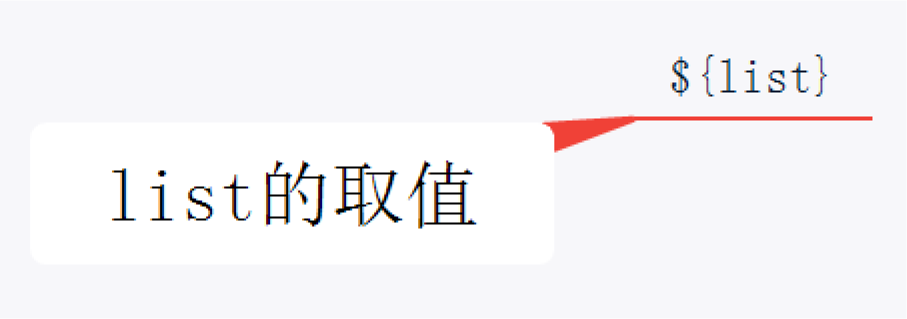
foreach(<loop_var> IN [LISTS [<lists>]])

8.3.2. ITEMS



foreach(<loop_var> IN [ITEMS [<items>]])

list的取值

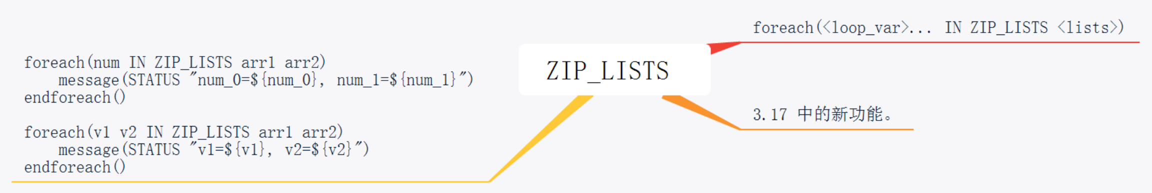


list的取值

${list}

${list}

8.3.3. ZIP_LISTS



```
foreach(num IN ZIP_LISTS arr1 arr2)
    message(STATUS ″num_0=${num_0}, num_1=${num_1}″)
endforeach()

foreach(v1 v2 IN ZIP_LISTS arr1 arr2)
    message(STATUS ″v1=${v1}, v2=${v2}″)
endforeach()
```

ZIP_LISTS

foreach(<loop_var>... IN ZIP_LISTS <lists>)

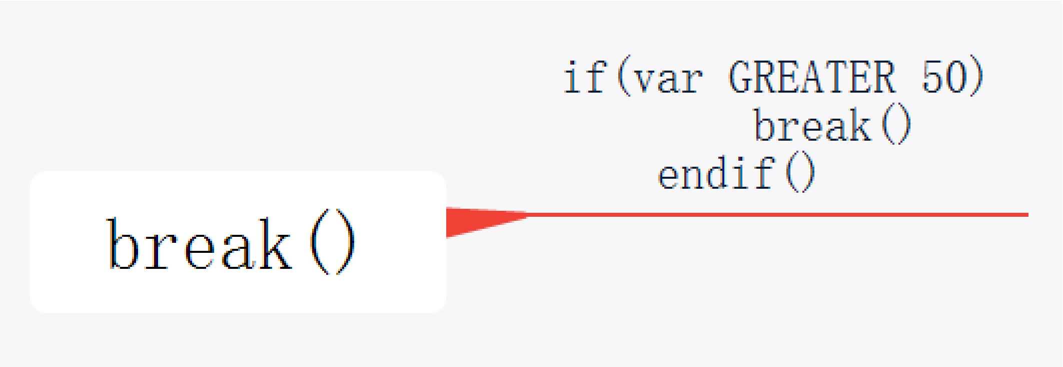
3.17 中的新功能。

foreach(<loop_var>... IN ZIP_LISTS <lists>)

3.17 中的新功能。

```
foreach(num IN ZIP_LISTS arr1 arr2)
  message(STATUS "num_0=${num_0}, num_1=${num_1}")
endforeach()


foreach(v1 v2 IN ZIP_LISTS arr1 arr2)
  message(STATUS "v1=${v1}, v2=${v2}")
endforeach()
```

8.4. break()

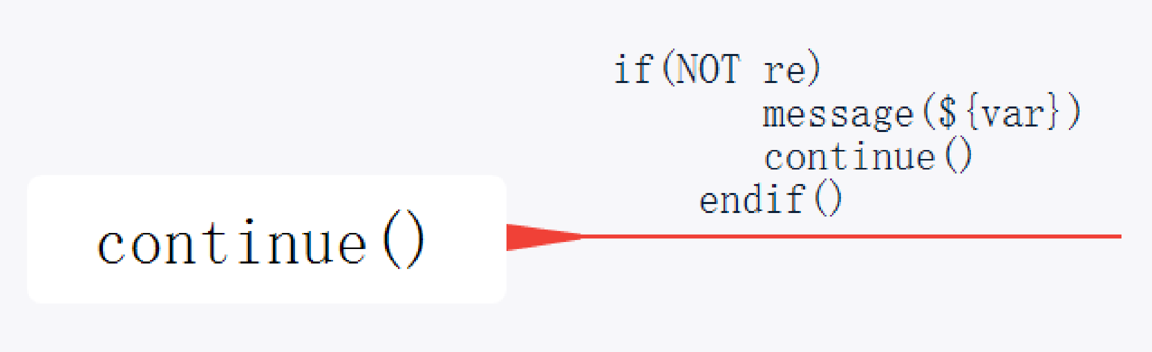


8.4.1. if(var GREATER 50)

break()

endif()

8.5. continue()



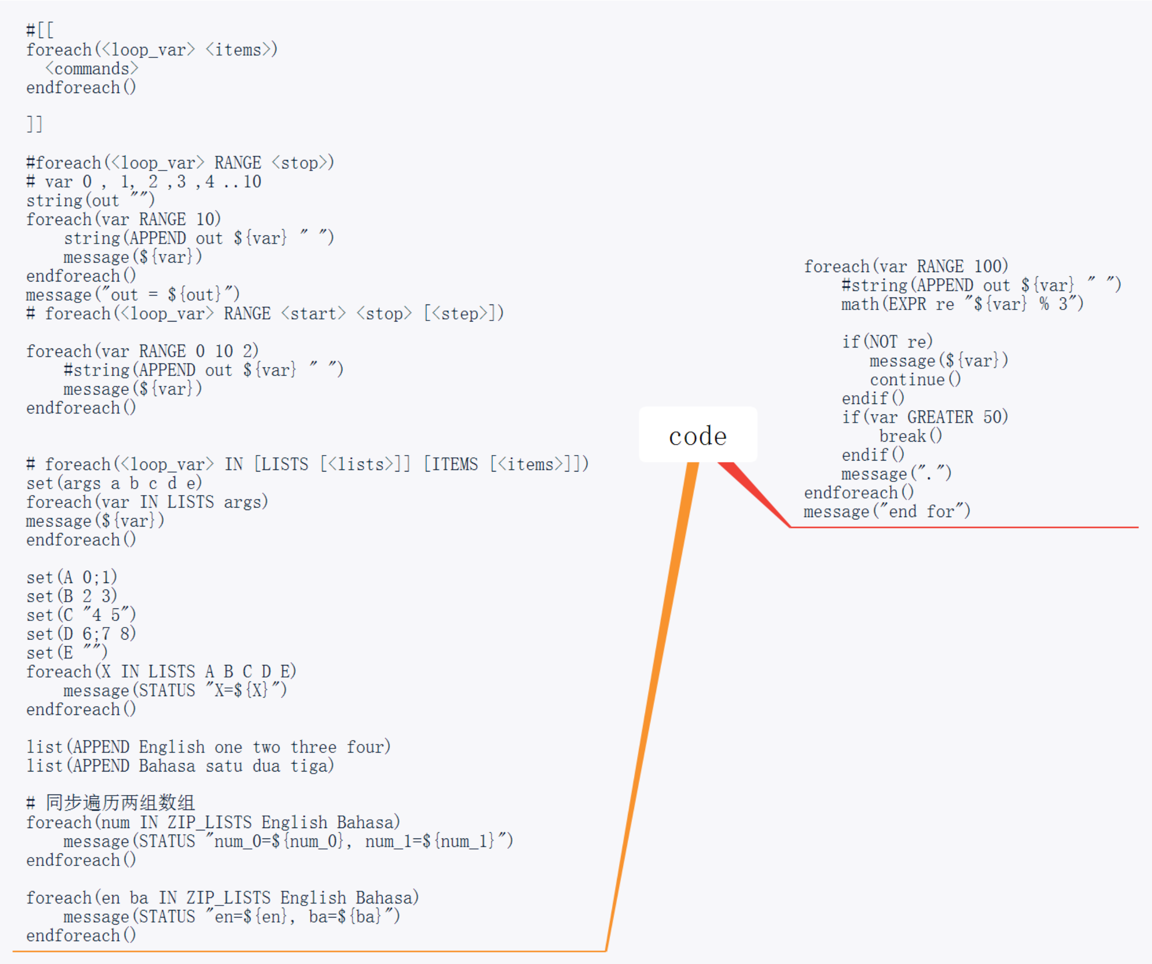
8.5.1. if(NOT re)

message(${var})

continue()

endif()

8.6. code



```
#[[
foreach(<loop_var> <items>)
  <commands>
endforeach()

]]

#foreach(<loop_var> RANGE <stop>)
# var 0 , 1, 2 ,3 ,4 ..10
string(out "")
foreach(var RANGE 10)
    string(APPEND out ${var} " ")
    message(${var})
endforeach()
message("out = ${out}")
# foreach(<loop_var> RANGE <start> <stop> [<step>])

foreach(var RANGE 0 10 2)
    #string(APPEND out ${var} " ")
    message(${var})
endforeach()


# foreach(<loop_var> IN [LISTS [<lists>]] [ITEMS [<items>]])
set(args a b c d e)
foreach(var IN LISTS args)
message(${var})
endforeach()

set(A 0;1)
set(B 2 3)
set(C "4 5")
set(D 6;7 8)
set(E "")
foreach(X IN LISTS A B C D E)
    message(STATUS "X=${X}")
endforeach()

list(APPEND English one two three four)
list(APPEND Bahasa satu dua tiga)

# 同步遍历两组数组
foreach(num IN ZIP_LISTS English Bahasa)
    message(STATUS "num_0=${num_0}， num_1=${num_1}")
endforeach()

foreach(en ba IN ZIP_LISTS English Bahasa)
    message(STATUS "en=${en}， ba=${ba}")
endforeach()
```

```
foreach(var RANGE 100)
    #string(APPEND out ${var} " ")
    math(EXPR re "${var} % 3")

    if(NOT re)
       message(${var})
       continue()
    endif()
    if(var GREATER 50)
        break()
    endif()
    message(".")
endforeach()
message("end for")
```

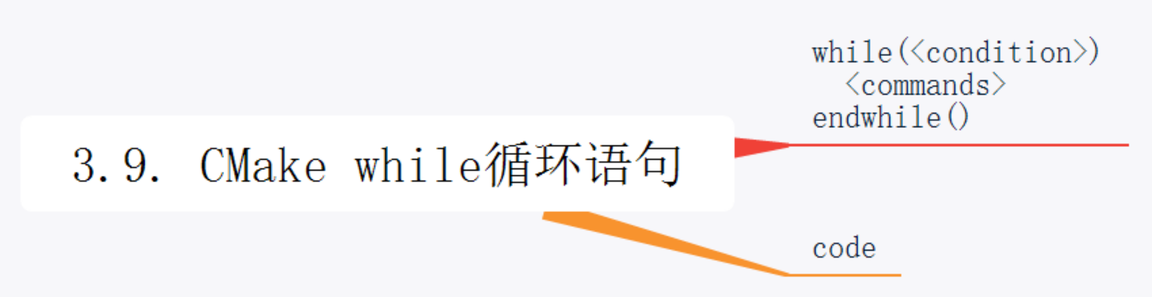
8.6.1. foreach(var RANGE 100)

```
  #string(APPEND out ${var} " ")
  math(EXPR re "${var} % 3")


  if(NOT re)
    message(${var})
    continue()
  endif()
  if(var GREATER 50)
     break()
  endif()
  message(".")
```

```
endforeach()
message("end for")

8.6.2. #[[
foreach(<loop_var> <items>)
  <commands>
endforeach()

]]

#foreach(<loop_var> RANGE <stop>)
# var 0 , 1, 2 ,3 ,4 ..10
string(out "")
foreach(var RANGE 10)
   string(APPEND out ${var} " ")
   message(${var})
endforeach()
message("out = ${out}")
# foreach(<loop_var> RANGE <start> <stop> [<step>])

foreach(var RANGE 0 10 2)
   #string(APPEND out ${var} " ")
   message(${var})
endforeach()


# foreach(<loop_var> IN [LISTS [<lists>]] [ITEMS [<items>]])
set(args a b c d e)
foreach(var IN LISTS args)
message(${var})
endforeach()
```

```
set(A 0;1)
set(B 2 3)
set(C "4 5")
set(D 6;7 8)
set(E "")
foreach(X IN LISTS A B C D E)
   message(STATUS "X=${X}")
endforeach()

list(APPEND English one two three four)
list(APPEND Bahasa satu dua tiga)

# 同步遍历两组数组
foreach(num IN ZIP_LISTS English Bahasa)
   message(STATUS "num_0=${num_0}, num_1=${num_1}")
endforeach()

foreach(en ba IN ZIP_LISTS English Bahasa)
   message(STATUS "en=${en}, ba=${ba}")
endforeach()
```

9. 3.9. CMake while循环语句



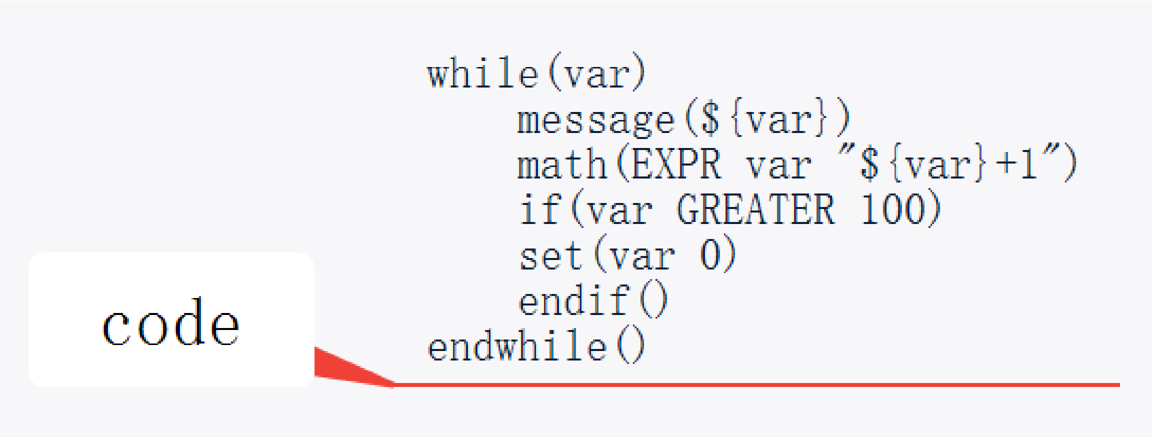
```
while(<condition>)
   <commands>
endwhile()
```

9.1. while(<condition>)

<commands>

endwhile()

9.2. code



```
while(var)
    message(${var})
    math(EXPR var "${var}+1")
    if(var GREATER 100)
    set(var 0)
    endif()
endwhile()
```

9.2.1. while(var)

message(${var})

math(EXPR var "${var}+1")

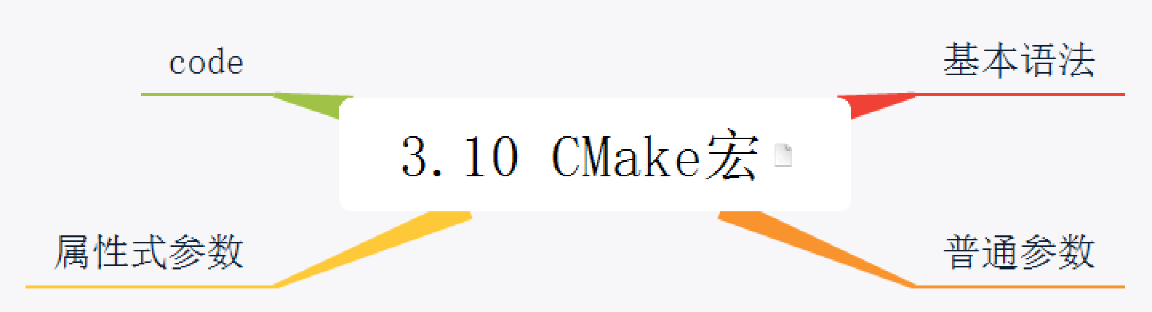
if(var GREATER 100)

set(var 0)

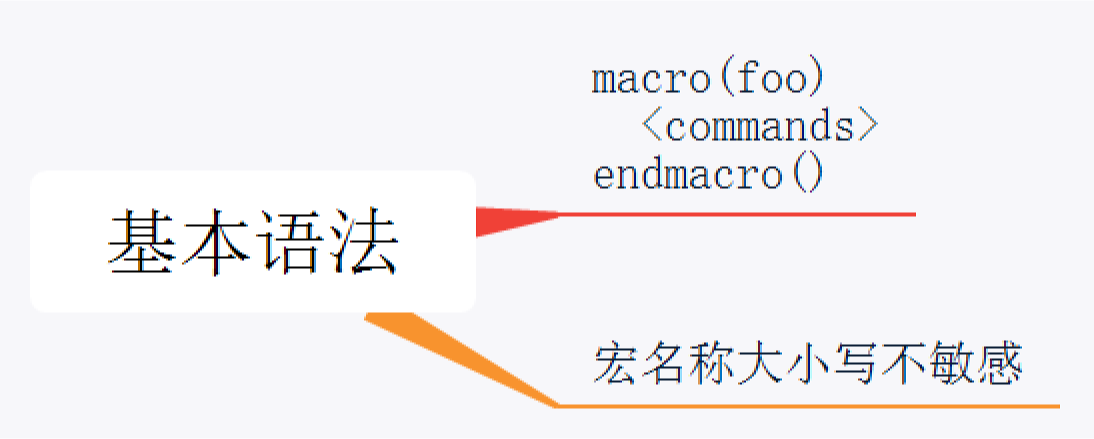
endif()

endwhile()

10. 3.10 CMake宏



10.1. 基本语法

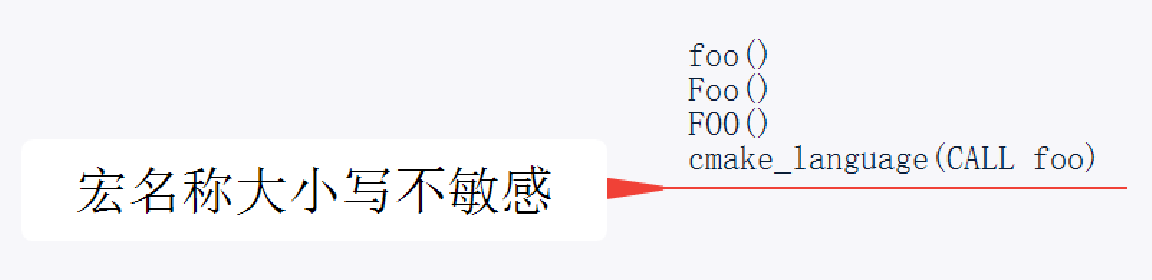


10.1.1. macro(foo)

 <commands>

endmacro()

10.1.2. 宏名称大小写不敏感



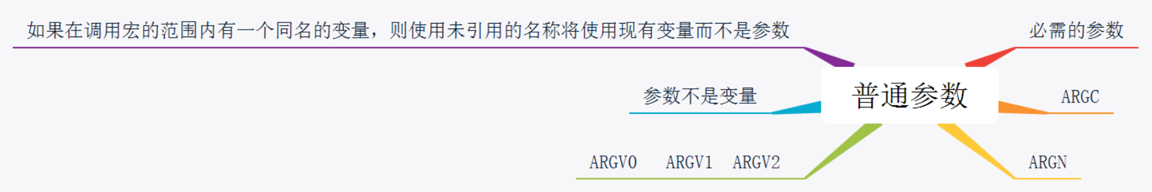
foo()

Foo()

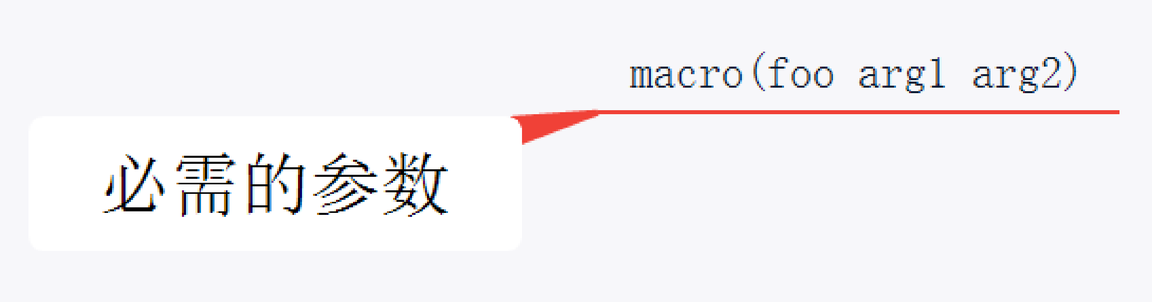
FOO()

cmake_language(CALL foo)

10.2. 普通参数


普通参数
如果在调用宏的范围内有一个同名的变量，则使用未引用的名称将使用现有变量而不是参数
必需的参数
参数不是变量
ARGC
ARGV0 ARGV1 ARGV2
ARGN

10.2.1. 必需的参数



macro(foo arg1 arg2)

10.2.2. ARGC

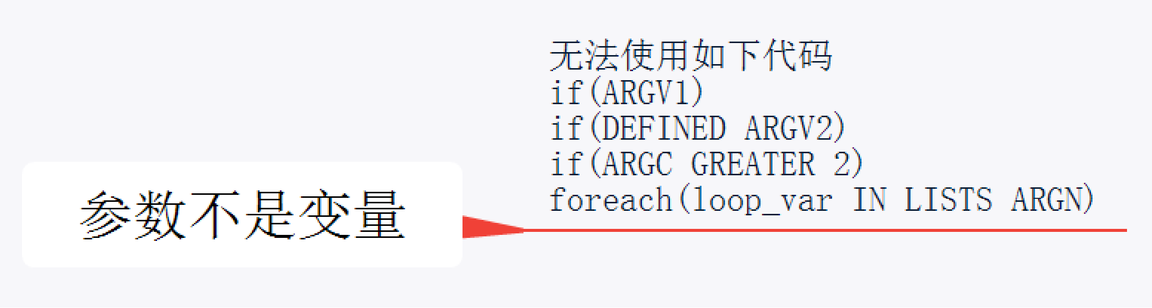
参数个数

10.2.3. ARGN

参数数组

10.2.4. ARGV0 ARGV1 ARGV2

10.2.5. 参数不是变量



无法使用如下代码

if(ARGV1)

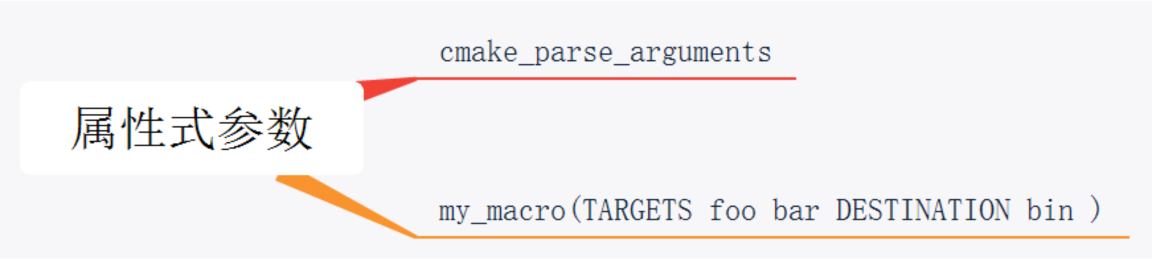
if(DEFINED ARGV2)

if(ARGC GREATER 2)

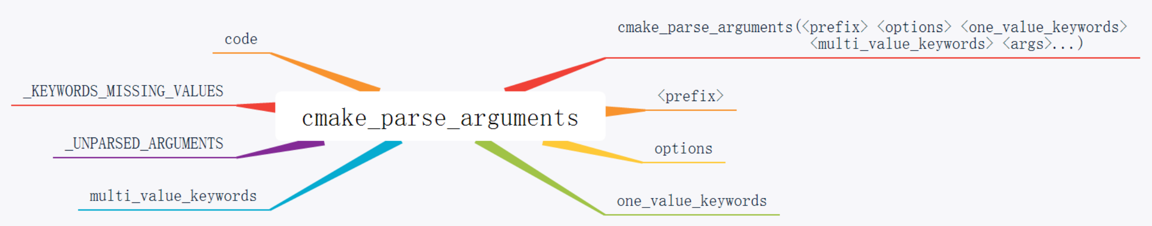
foreach(loop_var IN LISTS ARGN)

10.2.6. 如果在调用宏的范围内有一个同名的变量，则使用未引用的名称将使用现有变量而不是参数

10.3. 属性式参数

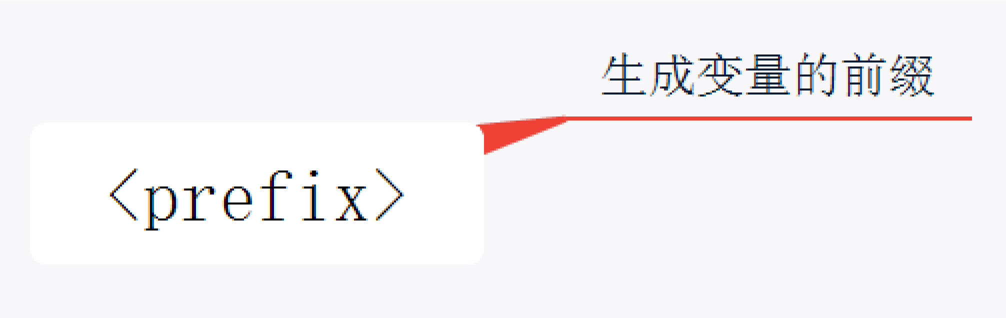


10.3.1. cmake_parse_arguments



cmake_parse_arguments(<prefix> <options> <one_value_keywords>
<multi_value_keywords> <args>...)

<prefix>

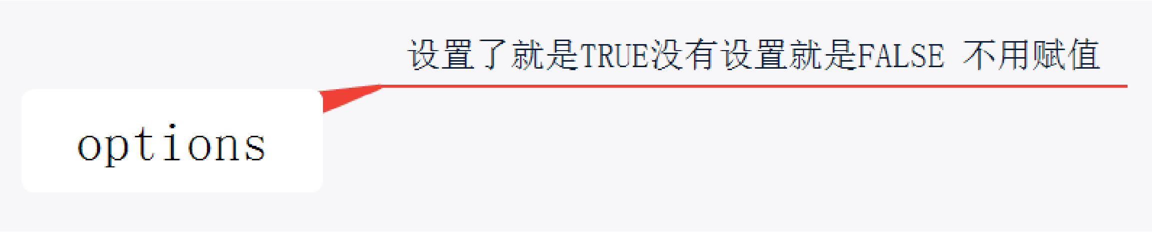


生成变量的前缀

```
<prefix>
```

生成变量的前缀

options

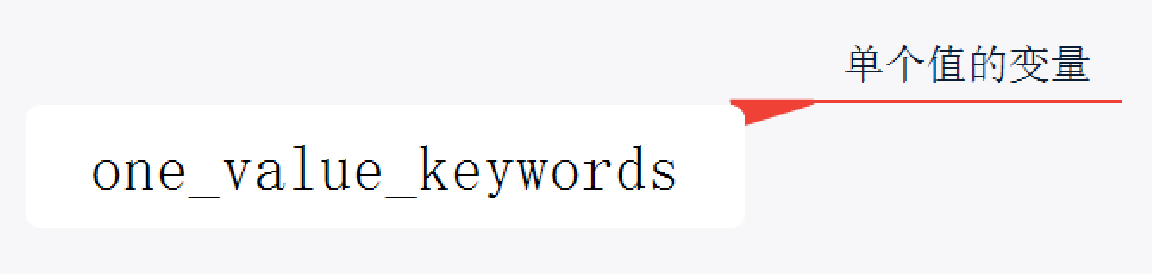


设置了就是TRUE没有设置就是FALSE 不用赋值

```
options
```

设置了就是TRUE没有设置就是FALSE 不用赋值

one_value_keywords

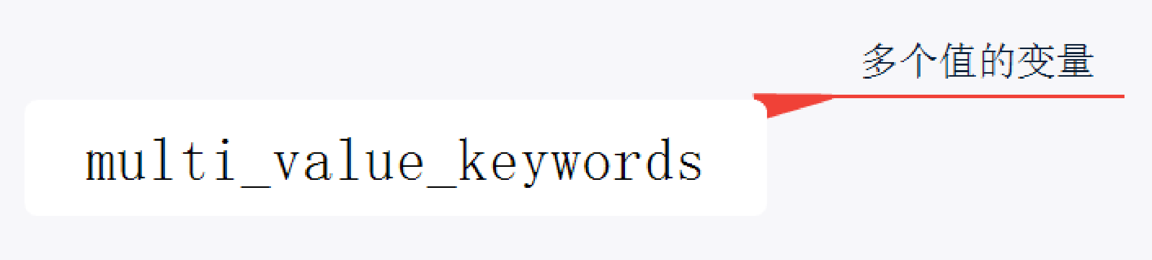


单个值的变量

```
one_value_keywords
```

单个值的变量

multi_value_keywords

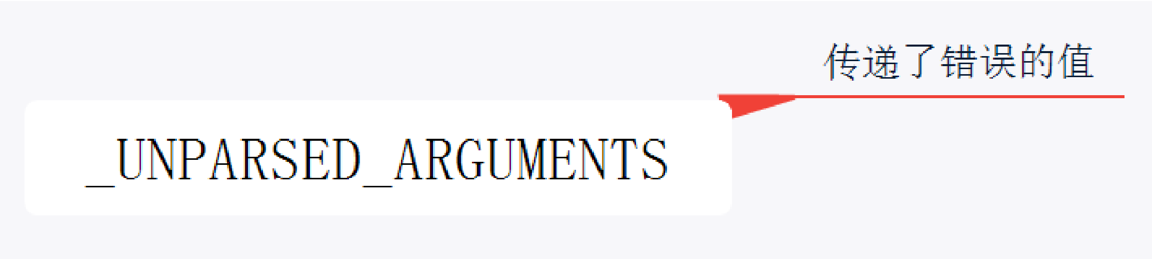


多个值的变量

```
multi_value_keywords
```

多个值的变量

_UNPARSED_ARGUMENTS

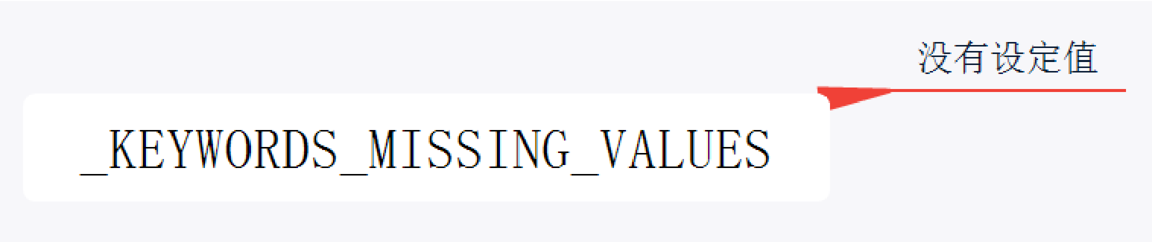


传递了错误的值

```
_UNPARSED_ARGUMENTS
```

传递了错误的值

_KEYWORDS_MISSING_VALUES

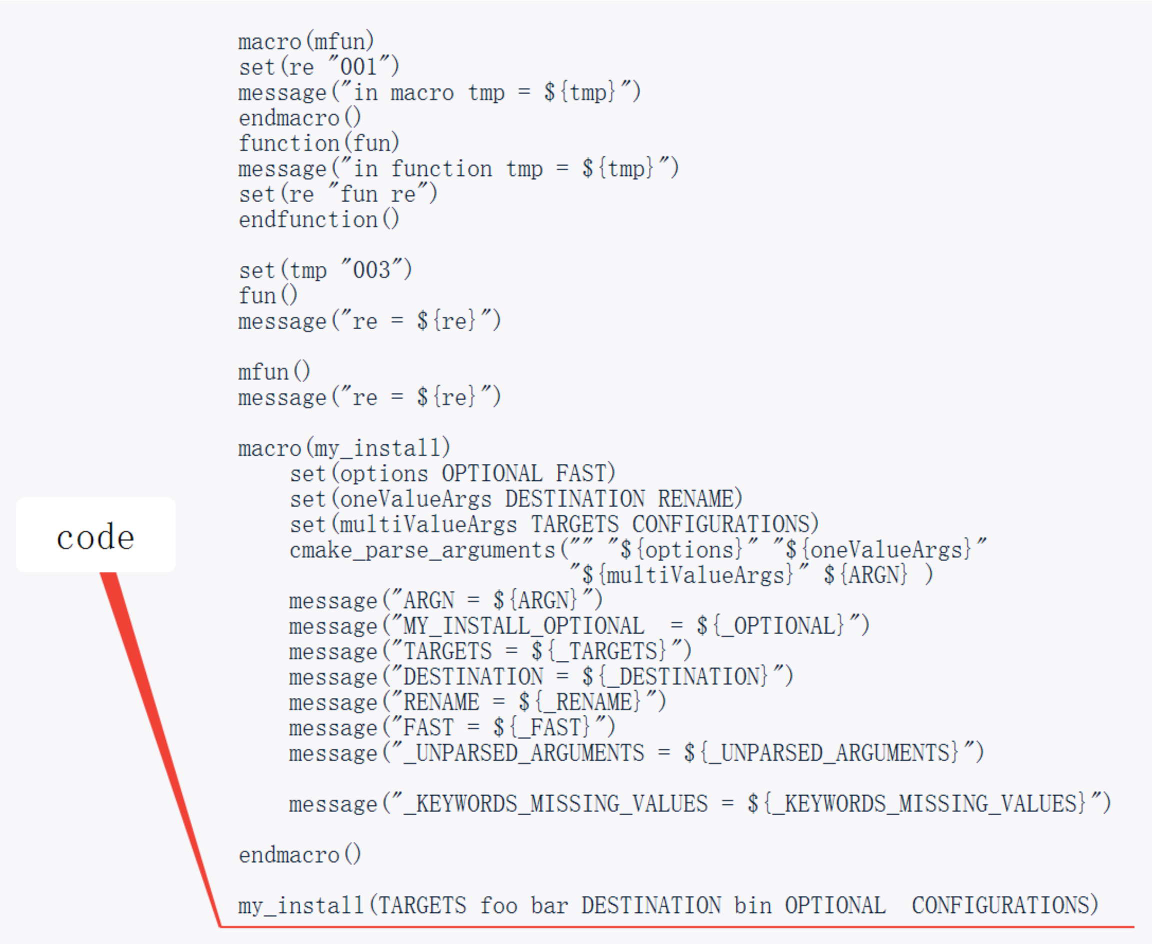


没有设定值

```
_KEYWORDS_MISSING_VALUES
```

没有设定值

code



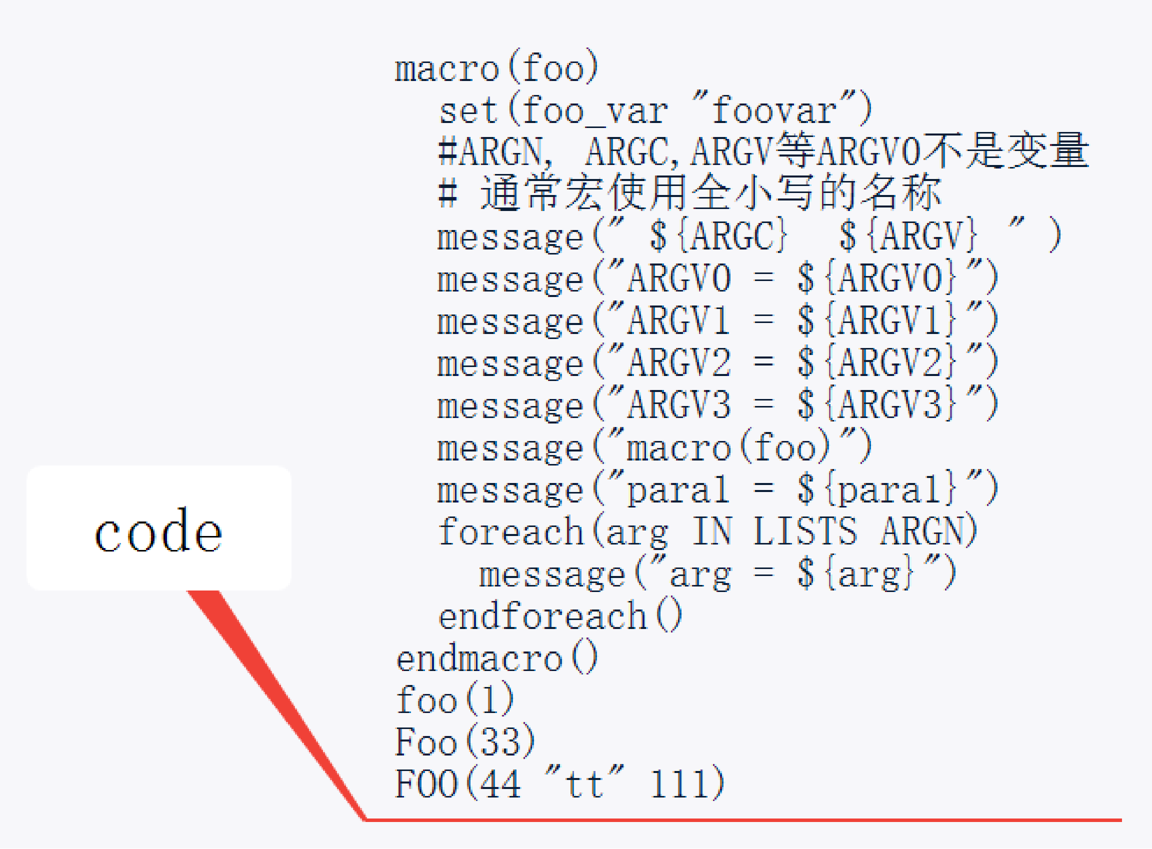
```
macro(mfun)
set(re "001")
message("in macro tmp = ${tmp}")
endmacro()
function(fun)
message("in function tmp = ${tmp}")
set(re "fun re")
endfunction()

set(tmp "003")
fun()
message("re = ${re}")

mfun()
message("re = ${re}")

macro(my_install)
    set(options OPTIONAL FAST)
    set(oneValueArgs DESTINATION RENAME)
    set(multiValueArgs TARGETS CONFIGURATIONS)
    cmake_parse_arguments("" "${options}" "${oneValueArgs}"
                          "${multiValueArgs}" ${ARGN} )
    message("ARGN = ${ARGN}")
    message("MY_INSTALL_OPTIONAL  = ${_OPTIONAL}")
    message("TARGETS = ${_TARGETS}")
    message("DESTINATION = ${_DESTINATION}")
    message("RENAME = ${_RENAME}")
    message("FAST = ${_FAST}")
    message("_UNPARSED_ARGUMENTS = ${_UNPARSED_ARGUMENTS}")

    message("_KEYWORDS_MISSING_VALUES = ${_KEYWORDS_MISSING_VALUES}")

endmacro()

my_install(TARGETS foo bar DESTINATION bin OPTIONAL  CONFIGURATIONS)
```

```
macro(mfun)
set(re "001")
message("in macro tmp = ${tmp}")
endmacro()
function(fun)
message("in function tmp = ${tmp}")
set(re "fun re")
endfunction()

set(tmp "003")
fun()
message("re = ${re}")
```

```
mfun()
message("re = ${re}")

macro(my_install)
  set(options OPTIONAL FAST)
  set(oneValueArgs DESTINATION RENAME)
  set(multiValueArgs TARGETS CONFIGURATIONS)
  cmake_parse_arguments("" "${options}" "${oneValueArgs}"
                        "${multiValueArgs}" ${ARGN} )
  message("ARGN = ${ARGN}")
  message("MY_INSTALL_OPTIONAL  = ${_OPTIONAL}")
  message("TARGETS = ${_TARGETS}")
  message("DESTINATION = ${_DESTINATION}")
  message("RENAME = ${_RENAME}")
  message("FAST = ${_FAST}")
  message("_UNPARSED_ARGUMENTS = ${_UNPARSED_ARGUMENTS}")

  message("_KEYWORDS_MISSING_VALUES =
${_KEYWORDS_MISSING_VALUES}")

endmacro()

my_install(TARGETS foo bar DESTINATION bin OPTIONAL  CONFIGURATIONS)
```

10.3.2. my_macro(TARGETS foo bar DESTINATION bin)

10.4. code



```
macro(foo)
  set(foo_var "foovar")
  #ARGN, ARGC,ARGV等ARGV0不是变量
  # 通常宏使用全小写的名称
  message(" ${ARGC}  ${ARGV}  " )
  message("ARGV0 = ${ARGV0}")
  message("ARGV1 = ${ARGV1}")
  message("ARGV2 = ${ARGV2}")
  message("ARGV3 = ${ARGV3}")
  message("macro(foo)")
  message("para1 = ${para1}")
  foreach(arg IN LISTS ARGN)
    message("arg = ${arg}")
  endforeach()
endmacro()
foo(1)
Foo(33)
FOO(44 "tt" 111)
```

10.4.1. macro(foo)

```
 set(foo_var "foovar")
 #ARGN, ARGC,ARGV等ARGV0不是变量
 # 通常宏使用全小写的名称
 message(" ${ARGC}  ${ARGV} " )
 message("ARGV0 = ${ARGV0}")
 message("ARGV1 = ${ARGV1}")
 message("ARGV2 = ${ARGV2}")
 message("ARGV3 = ${ARGV3}")
 message("macro(foo)")
 message("para1 = ${para1}")
 foreach(arg IN LISTS ARGN)
   message("arg = ${arg}")
 endforeach()
endmacro()
```

foo(1)

Foo(33)

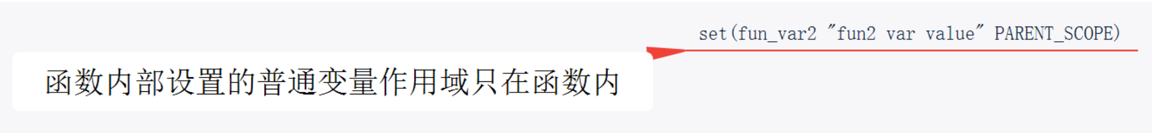
FOO(44 "tt" 111)

11. 3.11 CMake函数

code
函数的参数是变量
3. 11 CMake函数
函数可以用return返回
函数内部设置的普通变量作用域只在函数内

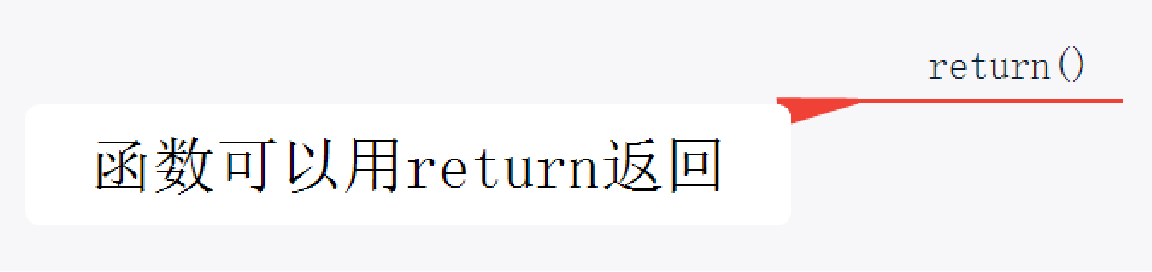
11.1. 函数的参数是变量

11.2. 函数内部设置的普通变量作用域只在函数内

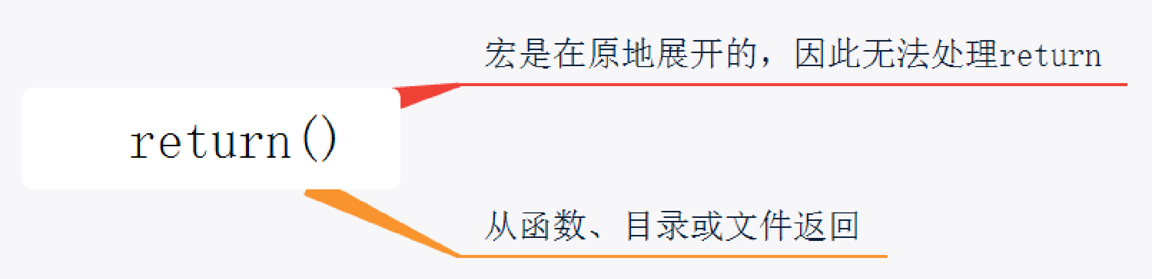

set(fun_var2 "fun2 var value" PARENT_SCOPE)
函数内部设置的普通变量作用域只在函数内

11.2.1. set(fun_var2 "fun2 var value" PARENT_SCOPE)

11.3. 函数可以用return返回



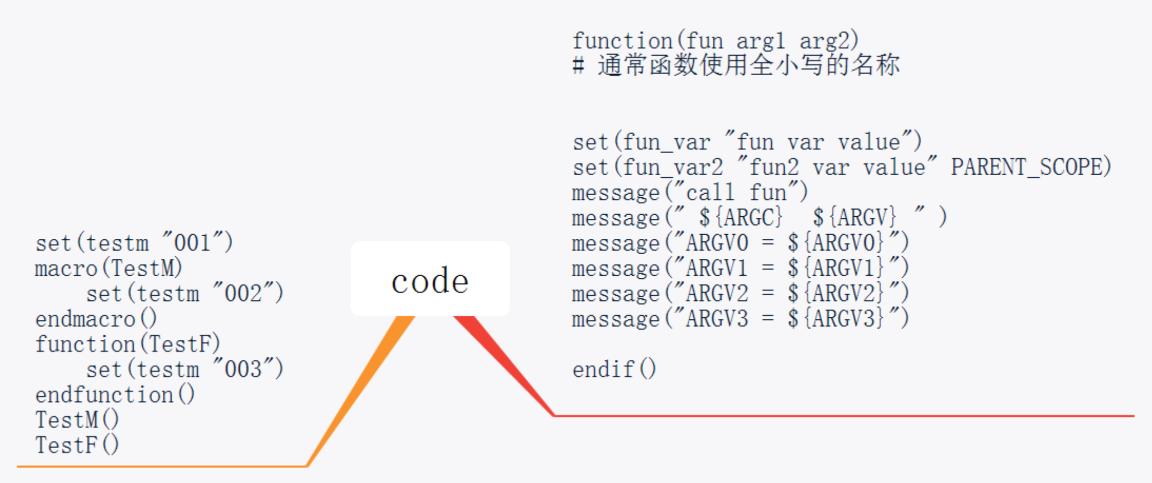
11.3.1. return()



宏是在原地展开的，因此无法处理**return**

从函数、目录或文件返回

11.4. code



```
set(testm "001")
macro(TestM)
    set(testm "002")
endmacro()
function(TestF)
    set(testm "003")
endfunction()
TestM()
TestF()
```

```
function(fun arg1 arg2)
# 通常函数使用全小写的名称


set(fun_var "fun var value")
set(fun_var2 "fun2 var value" PARENT_SCOPE)
message("call fun")
message(" ${ARGC}  ${ARGV} " )
message("ARGV0 = ${ARGV0}")
message("ARGV1 = ${ARGV1}")
message("ARGV2 = ${ARGV2}")
message("ARGV3 = ${ARGV3}")

endif()
```

11.4.1. function(fun arg1 arg2)

# 通常函数使用全小写的名称

set(fun_var "fun var value")

set(fun_var2 "fun2 var value" PARENT_SCOPE)

message("call fun")

message(" ${ARGC} ${ARGV} ")

message("ARGV0 = ${ARGV0}")

message("ARGV1 = ${ARGV1}")

message("ARGV2 = ${ARGV2}")

message("ARGV3 = ${ARGV3}")

endif()

11.4.2. set(testm "001")

macro(TestM)

```
   set(testm "002")
endmacro()
function(TestF)
   set(testm "003")
endfunction()
TestM()
TestF()
```