CMake使用

gaccob

2013 年 9 月 12 日

1. 什么是CMake

CMake是一个跨平台的自动化建构系统，它使用一个名为CMakeLists.txt的文件来描述构建过程，可以产生标准的构建文件，如Unix的Makefile或Windows Visual C++的projects/workspaces. 它的强大之处在于：跨平台，自动化.

2. 常用的语法规则

注释	#
设置变量	set(var, value)
条件判断	if() else() endif()
for循环	foreach(loopvar, arg1, arg2, ...) endforeach()
while循环	while(condition) endwhile(condition)

3. 常用的命令

```
1  # 指定cmake的版本依赖
2  cmake_minimum_required(**)
3
4  # 指定项目名称
5  project(**)
6
7  # 指定头文件的搜索路径，相当于指定gcc的-I参数
8  include_directories
9
10 # 动态链接库或静态链接库的搜索路径，相当于gcc的-L参数
11 link_directories
12
```

```
# 包含子目录
add_subdirectory

# 添加编译参数，例如add_definition(“-Wall -ggdb -O0″)
add_definitions

# 添加链接库，貌似可以不区分是共享库或者静态库，cmake会自动查找
target_link_libraries

# 编译可执行文件
add_executable

# 编译lib
add_library

# 打印日志到终端
message([SEND_ERROR | STATUS | FATAL_ERROR], “***”)
```

4. 常用的内部变量

```
# C编译器
MAKE_C_COMPILER

# C编译选项
CMAKE_C_FLAGS

# C++编译器
CMAKE_CXX_COMPILER

# C++编译选
CMAKE_CXX_FLAGS

# 可执行文件的存放路径
EXECUTABLE_OUTPUT_PATH

# 库文件路径
LIBRARY_OUTPUT_PATH

# build 类型，可以指定Debug或者Release
CMAKE_BUILD_TYPE

# 库类型，可以指定动态库或者静态库（ON/OFF）
BUILD_SHARED_LIBS

# 项目的根目录，可以在在CMakeLists.txt中set设置，也可以cmake时-D指定
CMAKE_SOURCE_DIR
```

5. 其他的一些用法

- 根据OS指定编译选项: if(APPLE); IF(UNIX); if(WIN32)
- 字符串比较: if(** STREQUAL **) ... endif()

6. gbase中的sample

CMakeLists.txt:

```
cmake_minimum_required(VERSION 2.8.10)
project(gbase)
include("${CMAKE_SOURCE_DIR}/gbase.cmake")
```

common.cmake:

```
# 禁止共享库
option(BUILD_SHARED_LIBS "build shared libraries." OFF)

# 编译类型
if (CMAKE_CONFIGURATION_TYPES)
    message(STATUS "build type: ${CMAKE_CONFIGURATION_TYPES}"
        )
elseif (CMAKE_BUILD_TYPE)
    message(STATUS "build type: ${CMAKE_BUILD_TYPE}")
else()
    set(CMAKE_BUILD_TYPE "Debug")
    message(STATUS "build type: ${CMAKE_BUILD_TYPE}")
endif()

# 编译选项（仅gcc、clang、vc）
if ("${CMAKE_CXX_COMPILER_ID}" STREQUAL "GNU")
    set(COMMON_DEBUG "-ggdb -Wall -Werror -pg -O0")
    set(COMMON_RELEASE "-ggdb -Wall -Werror -pg -O1")
    set (COMMON_LINK_LIB z dl pthread m)
elseif ("${CMAKE_CXX_COMPILER_ID}" STREQUAL "Clang")
    set(COMMON_DEBUG "-ggdb -Wall -Werror -pg -O0")
    set(COMMON_RELEASE "-ggdb -Wall -Werror -pg -O1")
    set (COMMON_LINK_LIB z dl pthread m)
elseif ("${CMAKE_CXX_COMPILER_ID}" STREQUAL "MSVC")
    set(COMMON_DEBUG "/Od /MDd")
    set(COMMON_RELEASE "/O2 /MD /D NDEBUG")
endif()
set(CMAKE_CXX_FLAGS_DEBUG "${COMMON_DEBUG}")
set(CMAKE_CXX_FLAGS_RELEASE "${COMMON_RELEASE}")
set(CMAKE_C_FLAGS_DEBUG "${COMMON_DEBUG}")
set(CMAKE_C_FLAGS_RELEASE "${COMMON_RELEASE}")

```

```
# 递归包含工程定义 *.cmake文件
macro(COMMON_PROJECT)
set(COMMON_PROJECT_FILTER "*.cmake")
foreach(basedir ${ARGV})
    file(GLOB_RECURSE COMMON_PROJECT_FILES "${basedir}/${COMMON_PROJECT_FILTER}")
    foreach(project_file ${COMMON_PROJECT_FILES})
        message(STATUS "project file found -- ${project_file}")
        include("${project_file}")
    endforeach()
endforeach()
endmacro(COMMON_PROJECT)

# 颜色回显
function(CommonEcho)
    # ${ARGV}, ${ARGN}
    set(ECHO_WITH_COLOR_CMD "echo")
    set(ECHO_WITH_COLOR_CMD_DP "")
    if(UNIX OR CYGWIN OR APPLE)
        set(TAG_RED     "\\033[31;1m")
        set(TAG_GREEN   "\\033[32;1m")
        set(TAG_YELLOW  "\\033[33;1m")
        set(TAG_BLUE    "\\033[34;1m")
        set(TAG_PURPLE  "\\033[35;1m")
        set(TAG_CYAN    "\\033[36;1m")
        set(TAG_RESET   "\\033[;0m")
        set(ECHO_WITH_COLOR_CMD_DP "-e")
    elseif(WIN32)
        set(TAG_RED     "")
        set(TAG_GREEN   "")
        set(TAG_YELLOW  "")
        set(TAG_BLUE    "")
        set(TAG_PURPLE  "")
        set(TAG_CYAN    "")
        set(TAG_RESET   "")
    endif()

    set(ECHO_WITH_COLOR_PREFIX "")
    set(ECHO_WITH_COLOR_SUFFIX "")
    set(ECHO_WITH_COLOR_FLAG "false")
    foreach (msg IN LISTS ARGV)
        if ( "${msg}" STREQUAL "COLOR" )
            set(ECHO_WITH_COLOR_FLAG "true")
        elseif( "${ECHO_WITH_COLOR_FLAG}" STREQUAL "true" )
            set(ECHO_WITH_COLOR_FLAG "false")
            if ("${msg}" STREQUAL "RED")
                set(ECHO_WITH_COLOR_PREFIX "${TAG_RED}")
                set(ECHO_WITH_COLOR_SUFFIX "${TAG_RESET}")
```

```
79              elseif ("${msg}" STREQUAL "GREEN")
80                  set(ECHO_WITH_COLOR_PREFIX "${TAG_GREEN}")
81                  set(ECHO_WITH_COLOR_SUFFIX "${TAG_RESET}")
82              elseif ("${msg}" STREQUAL "YELLOW")
83                  set(ECHO_WITH_COLOR_PREFIX "${TAG_YELLOW}")
84                  set(ECHO_WITH_COLOR_SUFFIX "${TAG_RESET}")
85              elseif ("${msg}" STREQUAL "BLUE")
86                  set(ECHO_WITH_COLOR_PREFIX "${TAG_BLUE}")
87                  set(ECHO_WITH_COLOR_SUFFIX "${TAG_RESET}")
88              elseif ("${msg}" STREQUAL "PURPLE")
89                  set(ECHO_WITH_COLOR_PREFIX "${TAG_PURPLE}")
90                  set(ECHO_WITH_COLOR_SUFFIX "${TAG_RESET}")
91              elseif ("${msg}" STREQUAL "CYAN")
92                  set(ECHO_WITH_COLOR_PREFIX "${TAG_CYAN}")
93                  set(ECHO_WITH_COLOR_SUFFIX "${TAG_RESET}")
94              else ()
95                  message(WARNING "EchoWithColor ${msg} not
                        supported.")
96              endif()
97          else()
98              execute_process(COMMAND ${ECHO_WITH_COLOR_CMD} ${
                    ECHO_WITH_COLOR_CMD_DP} "${
                    ECHO_WITH_COLOR_PREFIX}${msg}${
                    ECHO_WITH_COLOR_SUFFIX}")
99          endif()
100     endforeach()
101
102 endfunction(CommonEcho)
103
104 # 编译平台
105 if ("${CMAKE_CXX_SIZEOF_DATA_PTR}" STREQUAL "4")
106     CommonEcho(COLOR RED "-- platform ${CMAKE_CXX_PLATFORM_ID
            } 32")
107 elseif ("${CMAKE_CXX_SIZEOF_DATA_PTR}" STREQUAL "8")
108     CommonEcho(COLOR RED "-- platform ${CMAKE_CXX_PLATFORM_ID
            } 64")
109 else()
110     CommonEcho(COLOR RED "-- platform ${CMAKE_CXX_PLATFORM_ID
            } ??")
111 endif()
```

gbase.cmake:

```
1 include("${CMAKE_SOURCE_DIR}/common.cmake")
2
3 # 包含头文件
4 include_directories("${CMAKE_SOURCE_DIR}")
5
6 set(GBASE_DIR_CORE "${CMAKE_SOURCE_DIR}/core")
```

```
7  set(GBASE_DIR_DS "${CMAKE_SOURCE_DIR}/ds")
8  set(GBASE_DIR_NET "${CMAKE_SOURCE_DIR}/net")
9  set(GBASE_DIR_UTIL "${CMAKE_SOURCE_DIR}/util")
10 set(GBASE_DIR_TEST "${CMAKE_SOURCE_DIR}/test")
11
12 set(GBASE_LIB gbase)
13
14 # 链接选项
15 set (GBASE_LIB_LINK ${COMMON_LINK_LIB})
16
17 # 编译lib的源文件
18 aux_source_directory(${GBASE_DIR_CORE} GBASE_SOURCE)
19 aux_source_directory(${GBASE_DIR_DS} GBASE_SOURCE)
20 aux_source_directory(${GBASE_DIR_NET} GBASE_SOURCE)
21 aux_source_directory(${GBASE_DIR_UTIL} GBASE_SOURCE)
22 foreach(GBASE_SOURCE_FILE ${GBASE_SOURCE})
23     CommonEcho(COLOR CYAN "===> source: ${GBASE_SOURCE_FILE}"
       )
24 endforeach()
25
26 # 编译lib
27 add_library(${GBASE_LIB} ${GBASE_SOURCE})
28
29 # 递归增加test
30 file(GLOB GBASE_TEST_DIRS ${GBASE_DIR_TEST}/*test*)
31 foreach(GBASE_TEST_DIR ${GBASE_TEST_DIRS})
32     CommonEcho(COLOR CYAN "===> directory: ${GBASE_TEST_DIR}"
       )
33     add_subdirectory(${GBASE_TEST_DIR})
34 endforeach()
```