步步为赢

如果你想要得到你从来没有过的东西，
那么你必须去做你从来没做过的事情

目录视图

个人资料

carl_wang_cn

关注 发私信

访问： 365556次
积分： 3778
等级： BLOG 5
排名： 第5685名

原创： 64篇 转载： 4篇
译文： 1篇 评论： 67条

博客专栏

redis学习笔记
文章：14篇
阅读：51553

STL精粹
文章：5篇
阅读：5403

Lua入门与进阶
文章：14篇
阅读：147846

文章分类

Linux (26)
C++ (6)
MySQL (3)
Lua (14)
Redis (14)
杂七杂八 (1)

cmake使用示例与整理总结

标签： cmake makefile

2014-11-10 00:37 29607人阅读 评论

分类： Linux (25)

目录(?) [+]

本文代码托管于github cmake_demo

cmake中一些预定义变量

- PROJECT_SOURCE_DIR 工程的根目录
- PROJECT_BINARY_DIR 运行cmake命令的目录,通常是${PROJECT_SOURCE_DIR}/build
- CMAKE_INCLUDE_PATH 环境变量,非cmake变量
- CMAKE_LIBRARY_PATH 环境变量
- CMAKE_CURRENT_SOURCE_DIR 当前处理的CMakeLists.txt所在的路径
- CMAKE_CURRENT_BINARY_DIR target编译目录
  使用ADD_SURDIRECTORY(src bin)可以更改此变量的值
  SET(EXECUTABLE_OUTPUT_PATH <新路径>)并不会对此变量有影响,只是改变了最终目标文
- CMAKE_CURRENT_LIST_FILE 输出调用这个变量的CMakeLists.txt的完整路径
- CMAKE_CURRENT_LIST_LINE 输出这个变量所在的行
- CMAKE_MODULE_PATH 定义自己的cmake模块所在的路径
  SET(CMAKE_MODULE_PATH ${PROJECT_SOURCE_DIR}/cmake),然后可以用INCLUDE命令来调用自己的模块
- EXECUTABLE_OUTPUT_PATH 重新定义目标二进制可执行文件的存放位置
- LIBRARY_OUTPUT_PATH 重新定义目标链接库文件的存放位置
- PROJECT_NAME 返回通过PROJECT指令定义的项目名称
- CMAKE_ALLOW_LOOSE_LOOP_CONSTRUCTS 用来控制IF ELSE语句的书写方式

系统信息

- CMAKE_MAJOR_VERSION cmake主版本号,如2.8.6中的2
- CMAKE_MINOR_VERSION cmake次版本号,如2.8.6中的8
- CMAKE_PATCH_VERSION cmake补丁等级,如2.8.6中的6
- CMAKE_SYSTEM 系统名称,例如Linux-2.6.22
- CAMKE_SYSTEM_NAME 不包含版本的系统名,如Linux
- CMAKE_SYSTEM_VERSION 系统版本,如2.6.22
- CMAKE_SYSTEM_PROCESSOR 处理器名称,如i686
- UNIX 在所有的类UNIX平台为TRUE,包括OS X和cygwin
- WIN32 在所有的win32平台为TRUE,包括cygwin

开关选项

- BUILD_SHARED_LIBS 控制默认的库编译方式。如果未进行设置,使用ADD_LIBRARY时又没有指定库类型,默认编
  （可在t3中稍加修改进行验证）

- CMAKE_C_FLAGS 设置C编译选项
- CMAKE_CXX_FLAGS 设置C++编译选项

cmake常用命令

基本语法规则：

- cmake变量使用${}方式取值,但是在IF控制语句中是直接使用变量名
- 环境变量使用$ENV{}方式取值,使用SET(ENV{VAR} VALUE)赋值

- 指令(参数1 参数2...)
  参数使用括弧括起,参数之间使用空格或分号分开。

```
以ADD_EXECUTABLE指令为例：
ADD_EXECUTABLE(hello main.c func.c)或者
ADD_EXECUTABLE(hello main.c;func.c)
```

- 指令是大小写无关的,参数和变量是大小写相关的。推荐你全部使用**大写**指令。

部分常用命令列表：

- PROJECT
  PROJECT(projectname [CXX] [C] [Java])
  指定工程名称,并可指定工程支持的语言。支持语言列表可忽略,默认支持所有语言
- SET
  SET(VAR [VALUE] [CACHE TYPE DOCSTRING [FORCE]])
  定义变量(可以定义多个VALUE,如SET(SRC_LIST main.c util.c reactor.c))
- MESSAGE
  MESSAGE([SEND_ERROR | STATUS | FATAL_ERROR] “message to display” ...)
  向终端输出用户定义的信息或变量的值
  SEND_ERROR, 产生错误,生成过程被跳过
  STATUS, 输出前缀为—的信息
  FATAL_ERROR, 立即终止所有cmake过程
- ADD_EXECUTABLE
  ADD_EXECUTABLE(bin_file_name ${SRC_LIST})
  生成可执行文件
- ADD_LIBRARY
  ADD_LIBRARY(libname [SHARED | STATIC | MODULE] [EXCLUDE_FROM_ALL] SRC_LIST)
  生成动态库或静态库
  SHARED 动态库
  STATIC 静态库
  MODULE 在使用dyld的系统有效,若不支持dyld,等同于SHARED
  EXCLUDE_FROM_ALL 表示该库不会被默认构建
- SET_TARGET_PROPERTIES
  设置输出的名称,设置动态库的版本和API版本
- CMAKE_MINIMUM_REQUIRED
  CMAKE_MINIMUM_REQUIRED(VERSION version_number [FATAL_ERROR])
  声明CMake的版本要求
- ADD_SUBDIRECTORY
  ADD_SUBDIRECTORY(src_dir [binary_dir] [EXCLUDE_FROM_ALL])
  向当前工程添加存放源文件的子目录,并可以指定中间二进制和目标二进制的存放位置
  EXCLUDE_FROM_ALL含义：将这个目录从编译过程中排除
- SUBDIRS
  deprecated,不再推荐使用
  (hello sample)相当于分别写ADD_SUBDIRECTORY(hello),ADD_SUBDIRECTORY(sample)
- INCLUDE_DIRECTORIES
  INCLUDE_DIRECTORIES([AFTER | BEFORE] [SYSTEM] dir1 dir2 ...)
  向工程添加多个特定的头文件搜索路径,路径之间用空格分隔,如果路径包含空格,可以使用双引号将它括起来,默认的件搜索路径的后面。有如下两种方式可以控制搜索路径添加的位置：
  - CMAKE_INCLUDE_DIRECTORIES_BEFORE,通过SET这个cmake变量为on,可以将添加的头文件搜索路径放
  - 通过AFTER或BEFORE参数,也可以控制是追加还是置前
- LINK_DIRECTORIES
  LINK_DIRECTORIES(dir1 dir2 ...)
  添加非标准的共享库搜索路径
- TARGET_LINK_LIBRARIES
  TARGET_LINK_LIBRARIES(target lib1 lib2 ...)
  为target添加需要链接的共享库
- ADD_DEFINITIONS
  向C/C++编译器添加-D定义
  ADD_DEFINITIONS(-DENABLE_DEBUG -DABC),参数之间用空格分隔
- ADD_DEPENDENCIES
  ADD_DEPENDENCIES(target-name depend-target1 depend-target2 ...)
  定义target依赖的其他target,确保target在构建之前,其依赖的target已经构建完毕

- AUX_SOURCE_DIRECTORY
  AUX_SOURCE_DIRECTORY(dir VAR)
  发现一个目录下所有的源代码文件并将列表存储在一个变量中
  把当前目录下的所有源码文件名赋给变量DIR_HELLO_SRCS

- EXEC_PROGRAM
  EXEC_PROGRAM(Executable [dir where to run] [ARGS <args>][OUTPUT_VARIABLE <var>] [RETURN_VAL
  用于在指定目录运行某个程序（默认为当前CMakeLists.txt所在目录）,通过ARGS添加参数,通过OUTPUT_VARIA
  获取输出和返回值,如下示例

```
# 在src中运行ls命令,在src/CMakeLists.txt添加
EXEC_PROGRAM(ls ARGS "*.c" OUTPUT_VARIABLE LS_OUTPUT RETURN_VALUE LS_RVALUE)
IF (not LS_RVALUE)
    MESSAGE(STATUS "ls result: " ${LS_OUTPUT}) # 缩进仅为美观,语法无要求
ENDIF(not LS_RVALUE)
```

- INCLUDE
  INCLUDE(file [OPTIONAL]) 用来载入CMakeLists.txt文件
  INCLUDE(module [OPTIONAL])用来载入预定义的cmake模块
  OPTIONAL参数的左右是文件不存在也不会产生错误
  可以载入一个文件,也可以载入预定义模块（模块会在CMAKE_MODULE_PATH指定的路径进行搜索）
  载入的内容将在处理到INCLUDE语句时直接执行

- FIND_
  - FIND_FILE(<VAR> name path1 path2 ...)
    VAR变量代表找到的文件全路径,包含文件名
  - FIND_LIBRARY(<VAR> name path1 path2 ...)
    VAR变量代表找到的库全路径,包含库文件名

```
FIND_LIBRARY(libX X11 /usr/lib)
IF (NOT libx)
    MESSAGE(FATAL_ERROR "libX not found")
ENDIF(NOT libX)
```

  - FIND_PATH(<VAR> name path1 path2 ...)
    VAR变量代表包含这个文件的路径
  - FIND_PROGRAM(<VAR> name path1 path2 ...)
    VAR变量代表包含这个程序的全路径
  - FIND_PACKAGE(<name> [major.minor] [QUIET] [NO_MODULE] [[REQUIRED | COMPONENTS] [cor
    用来调用预定义在CMAKE_MODULE_PATH下的Find<name>.cmake模块,你也可以自己定义Find<name
    模块,通过SET(CMAKE_MODULE_PATH dir)将其放入工程的某个目录供工程使用

- IF
  语法：

```
IF (expression)
    COMMAND1(ARGS ...)
    COMMAND2(ARGS ...)
    ...
ELSE (expression)
    COMMAND1(ARGS ...)
    COMMAND2(ARGS ...)
    ...
ENDIF (expression) # 一定要有ENDIF与IF对应
```

  IF (expression), expression不为：空,0,N,NO,OFF,FALSE,NOTFOUND或<var>_NOTFOUND,为真
  IF (not exp), 与上面相反
  IF (var1 AND var2)
  IF (var1 OR var2)
  IF (COMMAND cmd) 如果cmd确实是命令并可调用,为真
  IF (EXISTS dir) IF (EXISTS file) 如果目录或文件存在,为真
  IF (file1 IS_NEWER_THAN file2),当file1比file2新,或file1/file2中有一个不存在时为真,文件名需使用全路径
  IF (IS_DIRECTORY dir) 当dir是目录时,为真
  IF (DEFINED var) 如果变量被定义,为真
  IF (var MATCHES regex) 此处var可以用var名,也可以用${var}
  IF (string MATCHES regex)

```
当给定的变量或者字符串能够匹配正则表达式regex时为真。比如:
IF ("hello" MATCHES "ell")
    MESSAGE("true")
ENDIF ("hello" MATCHES "ell")
```

数字比较表达式
IF (variable LESS number)
IF (string LESS number)
IF (variable GREATER number)
IF (string GREATER number)
IF (variable EQUAL number)
IF (string EQUAL number)

按照字母表顺序进行比较
IF (variable STRLESS string)
IF (string STRLESS string)
IF (variable STRGREATER string)
IF (string STRGREATER string)
IF (variable STREQUAL string)
IF (string STREQUAL string)

```
一个小例子,用来判断平台差异:
IF (WIN32)
    MESSAGE(STATUS “This is windows.”)
ELSE (WIN32)
    MESSAGE(STATUS “This is not windows”)
ENDIF (WIN32)
上述代码用来控制在不同的平台进行不同的控制,但是,阅读起来却并不是那么舒服,ELSE(WIN32)之类的语句很容易引起歧义
可以SET(CMAKE_ALLOW_LOOSE_LOOP_CONSTRUCTS ON)
这时候就可以写成:
IF (WIN32)
ELSE ()
ENDIF ()
配合ELSEIF使用,可能的写法是这样:
IF (WIN32)
    #do something related to WIN32
ELSEIF (UNIX)
    #do something related to UNIX
ELSEIF(APPLE)
    #do something related to APPLE
ENDIF (WIN32)
```

- WHILE
  语法：

```
WHILE(condition)
    COMMAND1(ARGS ...)
    COMMAND2(ARGS ...)
    ...
ENDWHILE(condition)
```

  其真假判断条件可以参考IF指令

- FOREACH
  FOREACH指令的使用方法有三种形式：
  1. 列表
     语法：

```
FOREACH(loop_var arg1 arg2 ...)
     COMMAND1(ARGS ...)
     COMMAND2(ARGS ...)
 ...
ENDFOREACH(loop_var)
```

     示例：

```
AUX_SOURCE_DIRECTORY(. SRC_LIST)
FOREACH(F ${SRC_LIST})
     MESSAGE(${F})
ENDFOREACH(F)
```

  2. 范围
     语法：

```
FOREACH(loop_var RANGE total)
    COMMAND1(ARGS ...)
    COMMAND2(ARGS ...)
    ...
ENDFOREACH(loop_var)
```

示例：

```
从0到total以1为步进
FOREACH(VAR RANGE 10)
   MESSAGE(${VAR})
ENDFOREACH(VAR)
输出：
012345678910
```

3. 范围和步进
语法：

```
FOREACH(loop_var RANGE start stop [step])
    COMMAND1(ARGS ...)
    COMMAND2(ARGS ...)
    ...
ENDFOREACH(loop_var)
```

从start开始到stop结束,以step为步进,
注意：直到遇到ENDFOREACH指令,整个语句块才会得到真正的执行。

```
FOREACH(A RANGE 5 15 3)
    MESSAGE(${A})
ENDFOREACH(A)
输出：
581114
```

cmake中如何生成动态库和静态库

参考ADD_LIBRARY和SET_TARGET_PROPERTIES用法

t3示例

cmake中如何使用动态库和静态库（查找库的路径）

参考INCLUDE_DIRECTORIES, LINK_DIRECTORIES, TARGET_LINK_LIBRARIES用法

t4示例使用动态库或静态库

t5示例如何使用cmake预定义的cmake模块(以FindCURL.cmake为例演示)

t6示例如何使用自定义的cmake模块(编写了自定义的FindHELLO.cmake)

注意读t5和t6的CMakeLists.txt和FindHELLO.cmake中的注释部分

cmake中如何指定生成文件的输出路径

- 如上ADD_SUBDIRECTORY的时候指定目标二进制文件输出路径（推荐使用下面这种）
- 使用SET命令重新定义EXECUTABLE_OUTPUT_PATH和LIBRARY_OUTPUT_PATH变量来指定最终的二进制文件的

```
SET(EXECUTABLE_OUTPUT_PATH ${PROJECT_BINARY_DIR}/bin)
SET(LIBRARY_OUTPUT_PATH ${PROJECT_BINARY_DIR}/lib)
```

上面的两条命令通常紧跟ADD_EXECUTABLE和ADD_LIBRARY,与其写在同一个CMakeLists.txt即可

cmake中如何增加编译选项

使用变量CMAKE_C_FLAGS添加C编译选项

使用变量CMAKE_CXX_FLAGS添加C++编译选项

使用ADD_DEFINITION添加

cmake中如何增加头文件路径

参考INCLUDE_DIRECTORIES命令用法

cmake中如何在屏幕上打印信息

参考MESSAGE用法

cmake中如何给变量赋值

参考SET和AUX_SOURCE_DIRECTORY用法

建议：在Project根目录先建立build,然后在build文件夹内运行cmake .. ，这样就不会污染源代码, 如果不想要这些自动生的删除build文件夹就可以

- 上一篇 Effective STL: 不同容器删除元素的方法
- 下一篇 Linux下各种查找命令（find, grep, which, whereis, locate）

我的同类文章

Linux（25）

- ftp 命令行操作 常用命令 2014-11-11 阅读 783
- 进程和线程的区别 2014-08-04 阅读 481
- shell日常使用整理 2014-06-15 阅读 1457
- setjmp, longjmp用法简介 2013-07-14 阅读 1550
- 正则表达式 进阶（二）-- 回溯引用、前... 2013-05-26 阅读 4930
- Linux下各种查找命令（find, grep, whi... 2014-11-
- 进程与线程的一个简单解释 2014-08-
- 关于C语言的隐式类型转换 2014-04-
- linux下踢出已登录用户 2013-07-
- 正则表达式 进阶（一）-- 匹配多连续字... 2013-05-

更多文章

参考知识库

猜你在找

查看评论