shuyong1999的专栏

个人资料

shuyong1999

访问： 245941次
积分： 3496
等级： BLOG 5
排名： 第6349名

原创： 84篇
转载： 157篇
译文： 0篇
评论： 32条

文章搜索

文章分类

Linux-Debian (49)
网站 (6)
PHP-Yii (7)
PHP (4)
Linux C/C++ (18)
VIM (8)
UML (4)
云 (1)
windows (0)
项目管理 (5)
MySql (2)
openCV (1)
PHP phpunit (2)
修灵虚境 (7)
5602项目失败总结 (1)
四民武术社网站创建流程日志 (14)
流媒体 (6)
面试宝典 (0)
2011－12月：闭关修炼宝典 (41)
2011－12月：闭关修炼宝典之面试宝典 (2)
2011－12月：闭关修炼宝典之敏

使用CMake构建项目的简明示例

标签： library include path build properties output

2011-12-28 12:50 2813人阅读 评论(3) 收藏 举报

分类： Linux-Debian（48）

1.需求

[1].使用第三方动/静太库

[2].本身代码部分编译为动/静态库

[3]多项目管理

2.构建一个单独的项目

[1]目的：这个项目将生成可执行文件，动态和静态库。

先在workspace(or anywhere you like)下建立项目的目录(文件)结构:

```
workspace
    |—— HelloWorld
            |—— CMakeLists.txt
            |—— include     Hello.h
            |—— src         Hello.cpp, test.cpp, CMakeLists.txt
            |—— build       （CMake外部构建时的使用的目录, 可任意指定）
```

[2]HelloWorld根目录下的CMakeLists.txt

```
# 声明CMake的最低要求版本
cmake_minimum_required(VERSION 2.8)
# 定义项目(工程)名称,
# 同时定义了以下默认变量:
# PROJECT_SOURCE_DIR 或 HELLO_SOURCE_DIR, 代指CMake开始构建的根目录(通常是项目根目录)
# PROJECT_BINARY_DIR 或 HELLO_BINARY_DIR, 代指CMake的编译目录(即执行cmake命令的目录)
PROJECT(HELLO)
# 添加参与编译的子目录
ADD_SUBDIRECTORY(src)
```

[3]Hello.h

```
#ifndef HELLO_H
#define HELLO_H
class Hello
{
public:
    Hello(){}
    void sayHello();
};
#endif
```

[4]Hello.cpp

```
#include <iostream>
#include "Hello.h"
void Hello::sayHello()
{
    std::cout << "Hello CMake!" << std::endl;
```

```
06. }
```

[5]test.cpp

```
01. #include "Hello.h"
02. int main(int arg, char** argv)
03. {
04.     Hello h;
05.     h.sayHello();
06. }
```

[6]src 下的CMakeLists.txt

```
01. # 添加头文件的查找目录
02. INCLUDE_DIRECTORIES(${PROJECT_SOURCE_DIR}/include)
03. ## 生成各种目标(target)文件: 可执行文件、动态库、静态库
04. # 指定可执行文件的输出目录，输出到bin下面
05. SET(EXECUTABLE_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/bin)
06. # 指定可执行文件名(hello)和相关源文件
07. ADD_EXECUTABLE(hello test.cpp Hello.cpp)
08. # 指定库文件输出路径
09. SET(LIBRARY_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/lib)
10. # 生成动态库
11. # 注意，前面已经使用target文件名hello，这里不能再用。
12. ADD_LIBRARY(hello_so SHARED Hello.cpp)
13. # 设置库输出名为 hello => libhello.so
14. SET_TARGET_PROPERTIES(hello_so PROPERTIES OUTPUT_NAME "hello")
15. # 生成静态库
16. # 注意，前面已经使用target文件名hello，这里不能再用。
17. ADD_LIBRARY(hello_a STATIC Hello.cpp)
18. # 设置输库出名为 hello => libhello.a
19. SET_TARGET_PROPERTIES(hello_a PROPERTIES OUTPUT_NAME "hello")
```

最后，进入build目录执行命令: cmake <项目根目录>
如果执行cmake命令的目录与项目根目录相同，称为内部编译，这时CMake生成的中间文件会与项目代码混合，不推荐。
否则，称为外部编译，所有中间文件会生成在执行cmake命令的目录下。
cmake执行完后，会生成Makefile, 直接make， 会在项目下生成bin和lib目录及目标文件。

```
01. cmake ..
02. make
```

通常我们不需要生成所有类型的目标文件，构建时根据需要选择。
3. 构建多个项目，使用外部项目提供的库文件。
在前面的HelloWorld项目中，生成了lib文件，就可以通过头文件和lib文件发布给其他项目了。
我们创建一个Test项目来使用HelloWorld的生成的库，如下:

```
workspace
    |—— HelloWorld
    |—— Test
            |—— CMakeLists.txt
            |—— src       main.cpp CMakeLists.txt
            |—— build
```

（1）Test目录下的CMakeLists.txt

```
01. cmake_minimum_required(VERSION 2.8)
02. PROJECT(TEST)
03. ADD_SUBDIRECTORY(src)
```

2) main.cpp

```
01. #include "Hello.h"
02. int main(int arg, char** argv)
03. {
04.     Hello h;
05.     h.sayHello();
06. }
```

(3) src下的CMakeLists.txt

```
01. # 显示系统的HOME环境变量的值
02. MESSAGE(STATUS $ENV{HOME})
03. # 指定头文件查找目录
04. # 注意，这里指定绝对路径。
05. INCLUDE_DIRECTORIES($ENV{HOME}/workspace/HelloWorld/include)
```

*Android_GestureDetector手势滑动使用

*Android MaterialList源码解析

*Android开源框架Universal-Image-Loader基本介绍及使用

*Android官方开发文档Training系列课程中文版：创建自定义View之View的创建


```
# 指定库文件查找目录（不能只指定头文件，也需要连接到库文件）
# 注意，这里指定绝对路径，也可通过设置系统环境变量LD_LIBRARY_PATH来指定。
LINK_DIRECTORIES($ENV{HOME}/workspace/HelloWorld/lib)
# 生成可执行文件到项目的bin目录
SET(EXECUTABLE_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/bin)
ADD_EXECUTABLE(main main.cpp)
# 制定链接的外部Lib
TARGET_LINK_LIBRARIES(main libhello.a)
```

最后，进入build目录执行:

```
cmake ..
make
```

到bin目录下测试可执行文件:

```
$ ./main
$ Hello CMake!
```

3. 在多个项目的情况下，使用自定义的**Find<ProjName>.cmake**模块

在workspace下增加两个目录(项目)，CMakeModules和TestFindModule如下：

workspace
|—— HelloWorld
|—— Test
|—— CMakeModules FindHELLO.cmake (存放各子项目的Find<Proj>.cmake 定义)
|—— TestFindModule (功能同Test子项目，不过使用Find<>.cmake模块来查找、链接HelloWorld项目的头文件和库)
|—— CMakeLists.txt
|—— src main.cpp，CMakeLists.txt
|—— build

（1）CMakeModules目录下的FindHELLO.cmake

```
MESSAGE(STATUS $ENV{HOME}/workspace/HelloWorld)
FIND_PATH(HELLO_INCLUDE_DIR Hello.h $ENV{HOME}/workspace/HelloWorld/include)


#FIND_LIBRARY(HELLO_LIBRARY NAMES hello PATHS $ENV{HOME}/workspace/HelloWorld/lib)
FIND_LIBRARY(HELLO_LIBRARY hello $ENV{HOME}/workspace/HelloWorld/lib)


IF (HELLO_INCLUDE_DIR)
  MESSAGE(STATUS "FOUND HELLO.H  ${HELLO_INCLUDE_DIR}")
ENDIF (HELLO_INCLUDE_DIR)


IF (HELLO_LIBRARY)
  MESSAGE(STATUS "FOUND HELLO LIB  ${HELLO_LIBRARY}")
ENDIF (HELLO_LIBRARY)


IF (HELLO_INCLUDE_DIR AND HELLO_LIBRARY)
  SET(HELLO_FOUND TRUE)
ENDIF (HELLO_INCLUDE_DIR AND HELLO_LIBRARY)


IF (HELLO_FOUND)
  IF (NOT HELLO_FIND_QUIETLY)
    MESSAGE(STATUS "Found Hello: ${HELO_LIBRARY}")
  ENDIF (NOT HELLO_FIND_QUIETLY)
ELSE (HELLO_FOUND)
  IF (HELLO_FIND_REQUIRED)
    MESSAGE(FATAL_ERROR "Could not find hello library")
  ENDIF (HELLO_FIND_REQUIRED)
ENDIF (HELLO_FOUND)
```

（2）TestFindModule目录下的CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8)

PROJECT(TEST)

# CMAKE_MODULE_PATH定义必须放到前面

SET(CMAKE_MODULE_PATH $ENV{HOME}/workspace/CMakeModules)

#MESSAGE(STATUS $ENV{HOME}/workspace/CMakeModules)
```

```
ADD_SUBDIRECTORY(src)
```

(2) TestFindModule目录src下的main.cpp

```
#include <Hello.h>
int main(int arg, char** argv)
{
    Hello h;
    h.sayHello();
}
```

(3) TestFindModule目录src下的CMakeLists.txt

```
# FIND_PACKAGE(<name>) 用来调用预定义在 CMAKE_MODULE_PATH 下的 Find<name>.cmake 模块
FIND_PACKAGE(HELLO)
IF (HELLO_FOUND)
    SET(EXECUTABLE_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/bin)
    ADD_EXECUTABLE(hello main.cpp)
    INCLUDE_DIRECTORIES(${HELLO_INCLUDE_DIR})
    TARGET_LINK_LIBRARIES(hello ${HELLO_LIBRARY})
ENDIF (HELLO_FOUND)
```

最后，进入build目录执行:

```
cmake ..
make
```

到bin目录下测试可执行文件:

```
$ ./main
$ Hello CMake!
```

10.

上一篇 CMake，新的KDE构建系统(转载)
下一篇 瑜伽

我的同类文章

Linux-Debian（48）

• 二，三，四层交换机的区别	2012-10-10 阅读 442	• TCP keepAlive	2012-09-28 阅读 310
• TCP连接状态详解	2012-09-28 阅读 335	• /bin/sh:can't access tty;job…	2012-05-07 阅读 1556
• VP UML8.0乱码问题	2012-05-02 阅读 829	• PHY芯片	2012-04-12 阅读 3989
• yed debian 64位机器不能…	2012-04-08 阅读 757	• Linux OpenGL开发	2012-03-14 阅读 451
• Linux 文档编排三剑客LaT…	2012-03-13 阅读 363	• 下一步的工作计划	2012-03-13 阅读 468
• Debian系安装中文字体	2012-03-07 阅读 4784		

更多文章

猜你在找

软考项目管理知识实战（上）	cmake使用示例与整理总结
软件测试基础	cmake使用示例与整理总结
零基础学HTML 5实战开发(第一季)	cmake使用示例与整理总结
2016软考系统集成项目管理工程师视频教程精讲 基础	cmake使用示例与整理总结
微信平台二次开发入门	cmake使用示例与整理总结

查看评论

3楼 k雪痕 2015-02-03 20:39发表

收益很多，感谢分享

2楼 qudongtianxia 2014-03-11 17:42发表

好的 写的好

1楼 qudongtianxia 2014-03-11 17:42发表

好 写的好

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker
OpenStack VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC
WAP jQuery BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML
LBS Unity Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra
CloudStack FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App
SpringSide Maemo Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP
HBase Pure Solr Angular Cloud Foundry Redis Scala Django Bootstrap