

shuyong1999的专栏

目录视图

摘要视图

RSS 订阅

个人资料



shuyong1999

访问: 245941次

积分: 3496

等级: 

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)

修真虚境 (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)

版权声明: 本文为博主原创文章, 未经博主允许不得转载。

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

01. # 声明CMake的最低要求版本

02. cmake\_minimum\_required(VERSION 2.8)

03. # 定义项目(工程)名称,

04. # 同时定义了以下默认变量:

05. # 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)

06. # PROJECT\_BINARY\_DIR 或 HELLO\_BINARY\_DIR, 代指CMake的编译目录(即执行cmake命令的目录)

07. PROJECT(HELLO)

08. # 添加参与编译的子目录

09. ADD\_SUBDIRECTORY(src)

[3]Hello.h

01. #ifndef HELLO\_H

02. #define HELLO\_H

03. class Hello

04. {

05. public:

06.     Hello(){}

07.     void sayHello();

08. };

09. #endif

[4]Hello.cpp

01. #include <iostream>

02. #include "Hello.h"

03. void Hello::sayHello()

04. {

05.     std::cout << "Hello CMake!" << std::endl;

- 捷开发之团队管理系统系列 (0)
- 2011—12月：闭关修炼宝典之敏捷开发之生态系统系列 (5)
- 初入社会 (2)
- TS (0)
- RTP (0)
- GOF设计模式 (1)
- 要做但没做的内容 (17)
- git (8)
- FVWM (1)
- 数学 (0)
- 云平台 (5)
- GUI (1)
- 人工智能 (1)
- Cloudxy (1)
- 项目管理 (0)
- android (5)
- 数学 (0)
- IPD-CMMI项目开发实战 (1)
- Objective-C (0)
- 飞机对抗仿真 (6)
- X264 (1)

- 文章存档
- 2013年02月 (1)
- 2012年12月 (1)
- 2012年11月 (2)
- 2012年10月 (2)
- 2012年09月 (6)
- 展开

- 阅读排行
- git ssh远程登录 (12143)
- Gitolite 构建 Git 服务器端 (8922)
- debian的U盘安装 (8337)
- TS数据结构分析 (6836)
- ffmpeg与TS (6030)
- Linux内存管理机制简介 (5417)
- Mpeg-2的同步及时间恢复 (4851)
- Debian系安装中文字体 (4786)
- ES PES TS (4196)
- HTML的发展历史 (4182)

- 评论排行
- 使用CMake构建项目的简 (3)
- TS数据结构分析 (3)
- windows的云平台技术 (2)
- debian的U盘安装 (2)
- 图像sensor的工作原理 (2)
- HTML的发展历史 (2)
- Audio PCM (2)
- VP UML8.0乱码问题 (2)
- TS 流解码过程 (1)
- UML看的书籍 (1)

- 推荐文章
- \*Android RocooFix 热修复框架
- \* android6.0源码分析之Camera API2.0下的初始化流程分析

```
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. # 设置库输出名为 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:



(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

## 最新评论

图像sensor的工作原理  
hduewenliu: good

Audio PCM  
mengxiangchun0507: 解释的很清楚

windows的云台技术  
qsh\_zh: 云的时代到来啦。

UML看的书籍  
qsh\_zh: 多看书，多实践。

HTML的发展历史  
qsh\_zh: 不管你正在使用的HTML是哪个版本，你已经在使用HTML 5了

HTML的发展历史  
qsh\_zh: 不管你正在使用的HTML是哪个版本，你已经在使用HTML 5了

图像sensor的工作原理  
confidence321: very good

使用CMake构建项目的简明示例  
k雪痕: 收益很多，感谢分享

如何在 linux 下检测内存泄漏  
yuganxxx: 你好，听你这么说，这个工具是很牛B的样子，但是我下载不到，你有吗，能否发一份给我？多谢多谢我的邮箱y...

使用CMake构建项目的简明示例  
qudongtianxia: 好的 写的好

```
06. # 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）
07. # 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。
08. LINK_DIRECTORIES(${ENV{HOME}}/workspace/HelloWorld/lib)
09. # 生成可执行文件到项目的bin目录
10. SET(EXECUTABLE_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/bin)
11. ADD_EXECUTABLE(main main.cpp)
12. # 制定链接的外部Lib
13. TARGET_LINK_LIBRARIES(main libhello.a)
```

最后，进入build目录执行：

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

到bin目录下测试可执行文件：

```
01. $ ./main
02. $ 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.
```

顶

0

踩

0

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连接状态详解

2012-09-28

阅读 335

• VP UML8.0乱码问题

2012-05-02

阅读 829

• yed debian 64位机器不能...

2012-04-08

阅读 757

• Linux 文档编排三剑客LaT...

2012-03-13

阅读 363

• Debian系安装中文字体

2012-03-07

阅读 4784

• TCP keepAlive

2012-09-28

阅读 310

• /bin/sh:can't access tty:job...

2012-05-07

阅读 1556

• PHY芯片

2012-04-12

阅读 3989

• Linux OpenGL开发

2012-03-14

阅读 451

• 下一步的工作计划

2012-03-13

阅读 468

更多文章

猜你在找

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

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服   杂志客服   微博客服   webmaster@csdn.net   400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏乐知网络技术有限公司 提供商务支持  
京 ICP 证 09002463 号 | Copyright © 1999-2014, CSDN.NET, All Rights Reserved

