

Coder 商业化产品可能是有用的，但最好的东西往往是免费的，空气，水，Vim。


≡ 目录视图

≡ 摘要视图

RSS 订阅

wangeen's technique blog

个人资料



wangeen

访问：459772次
积分：7093
等级：


BLOG 6

排名：第2014名


原创：232篇 转载：155篇
译文：2篇 评论：30条

文章搜索


博客专栏



MPI 分布式编程
文章：10篇
阅读：13175



VIM
文章：21篇
阅读：21807



BOOST
文章：11篇
阅读：10325

文章分类

linux (40)

mac (48)

Libraries (9)

Go language (2)

C++ language (39)

Tools (13)

并行计算 (18)

应用 (17)

java (2)

CMake Practice 读书笔记

2013-01-18 23:02 4751人阅读 评论(0) 收藏 举报

版权声明：本文为博主原创文章，未经博主允许不得转载。

具体问题可以参考官方文档：http://www.cmake.org/cmake/help/v2.8.10/cmake.html#section_Commands
http://zh.wikibooks.org/wiki/CMake_%E5%85%A5%E9%96%80/%E5%8A%A0%E5%85%A5%E7%B7%A8%E8%AD%

CMake的历史是随着KDE4流行开来的，
主要特点
1 开源
2 跨平台
3 能管理大项目KDE4就是最好的证明
4 简化构建和编译过程 CMake+make
我们感受到最实在的就是2 3 4这几点。CMake比直接编辑makefile语法简单，容易上手，更容易扩展，更适应大的和多变的工程。
建议
1 如果你使用的项目只有几个文件，直接用makefile就可以了（考虑到跨平台除外）
2 C/C++/Java之外的请不要用cmake
3 如果你的语言有完整的构建体系，比如java的ant请不要用cmake
4 如果项目已经有比较完备的管理体系，没有必要用cmake
5 如果只是qt编程没有必要用cmake，因为qmake就够了
总是合适就好，不要滥用。
问题1： 如何清理cmake的临时文件
cmake 并不支持 make distclean,关于这一点,官方是有明确解释的: 因为 CMakeLists.txt 可以执行脚本并通过脚本生成一些临时文件,但是却没有办法来跟
踪这些临时文件到底是哪些。因此,没有办法提供一个可靠的make distclean方案。 同时,还有另外一个非常重要的提示,就是:我们刚才进行的是内部构建(in-sourcebuild),而cmake强烈推荐的是外部构建(out-of-source build)。所以结论是尽量用out of source。

http://blog.csdn.net/wangeen/article/details/8518549

1/6

python (35)

qt (9)

boost (9)

vim (22)

git (9)

web (17)

windows (1)

开发方法 (1)

photoshop (3)

bashrc (1)

Effective STL (3)

jquery (1)

javascript (1)

put (0)

tkinter (1)

twisted (1)

GDB (4)

django (11)

推荐系统 (1)

Aaron Swartz (1)

web ios (1)

websocket (1)

sqlite (2)

文章存档

2015年11月 (1)

2015年10月 (3)

2015年06月 (1)

2015年04月 (1)

2015年02月 (4)

展开

问题2： 换个地方保存目标二进制

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

问题是,我应该把这两条指令写在工程的 CMakeLists.txt 还是 src 目录下的CMakeLists.txt,把握一个简单的原则,在 哪里 ADD_EXECUTABLE 或 ADD_LIBRARY,如果需要改变目标存放路径,就在那里加入上述的定义。

问题3： 如何实现install

```
INSTALL(TARGETS myrun mylib mystaticlib
        RUNTIME DESTINATION bin
        LIBRARY DESTINATION lib
        ARCHIVE DESTINATION libstatic
)
```

上面的例子会将:

可执行二进制 myrun 安装到\${CMAKE_INSTALL_PREFIX}/bin 目录

动态库 libmylib 安装到\${CMAKE_INSTALL_PREFIX}/lib 目录

静态库 libmystaticlib 安装到\${CMAKE_INSTALL_PREFIX}/libstatic 目录

如果我没有定义 CMAKE_INSTALL_PREFIX 会安装到什么地方?

你可以尝试以下,cmake ..;make;make install,你会发现CMAKE_INSTALL_PREFIX 的默认定义是/usr/local

```
INSTALL(FILES hello.h DESTINATION include/hello)

cmake -DCMAKE_INSTALL_PREFIX=/usr ..

make
make install
```

我们就可以将头文件和共享库安装到系统目录/usr/lib 和/usr/include/hello 中了。

问题4： 库格式的支持

CMake支持三种库

SHARED,动态库

STATIC,静态库

MODULE,在使用 dyld 的系统有效,如果不支持 dyld,则被当作 SHARED 对待。

问题5： 如何解决动态和静态库重名

当我们需要同时建立动态静态库的时候可能会遇到麻烦,因为不能用同一个target名称,解决办法

```
SET_TARGET_PROPERTIES(hello PROPERTIES CLEAN_DIRECT_OUTPUT 1)

SET_TARGET_PROPERTIES(hello_static PROPERTIES CLEAN_DIRECT_OUTPUT
1)
```

问题6： 如何解决动态库的版本的支持（这一点在开发的过程中显得十分有用,可以用一下）

动态库版本号按照规则,动态库是应该包含一个版本号的,我们可以看一下系统的动态库,一般情况是libhello.so.1.2

```
libhello.so ->libhello.so.1
```

```
libhello.so.1->libhello.so.1.2
```

```
SET_TARGET_PROPERTIES(hello PROPERTIES VERSION 1.2 SOVERSION 1)
```

问题7：那如何链接到静态库呢？

方法很简单：

将 `TARGET_LINK_LIBRARIES` 指令修改为：`TARGET_LINK_LIBRARIES(main libhello.a)`

在find之后，应该还有其他办法作者没有讲到。

问题8：如何修改编译选项

`CMAKE_C_FLAGS`

设置 C 编译选项,也可以通过指令 `ADD_DEFINITIONS()`添加。

`CMAKE_CXX_FLAGS`

设置 C++编译选项,也可以通过指令 `ADD_DEFINITIONS()`添加。

问题9：如何添加宏编译

`ADD_DEFINITIONS`

向 C/C++编译器添加-D 定义,比如:

`ADD_DEFINITIONS(-DENABLE_DEBUG -DABC)`,参数之间用空格分割。

如果你的代码中定义了`#ifdef ENABLE_DEBUG #endif`,这个代码块就会生效。

问题10：如何使用**testcase**（过去没有什么体会，看到这个觉得很有用，显然可以提高开发的效率，要应用）

`ADD_TEST` 指令的语法是：

```
ADD_TEST(testname Exename arg1 arg2 ...)
```

```
ADD_TEST(mytest ${PROJECT_BINARY_DIR}/bin/main)ENABLE_TESTING()
```

生成Makefile后,就可以运行**make test**来执行测试了。

问题11：自动添加所有的源代码（避免添加很多源文件的麻烦，有利也有弊）

`AUX_SOURCE_DIRECTORY(dir VARIABLE)`

作用是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,这个指令临时被用来自动构建源文件列表。因为目前 **cmake** 还不能自动发现新添加的源文件。

比如

```
AUX_SOURCE_DIRECTORY(. SRC_LIST)
```

```
ADD_EXECUTABLE(main ${SRC_LIST})
```

问题12：如何调用其他的可执行程序

```
EXEC_PROGRAM(Executable [directory in which to run]  
             [ARGS <arguments to executable>])
```

```
[OUTPUT_VARIABLE <var>]
```

```
[RETURN_VALUE <var>])
```

比如可以运行一些shell的command

问题13: 如何实现对**FILE**的操作

FILE 指令 指令很简单 可以完成一些文件的move delete rename read write

问题14: 如何使用其他编译器比如**icc**

There's two ways we do it. If you are starting with a clean cache (first time you run cmake), if you do:

```
CC=icc FC=ifort cmake /path/to/source
```

It will set it up using Intel compilers.

We also do:

```
include(CMakeForceCompiler)
CMAKE_FORCE_C_COMPILER(icc "Intel C Compiler")
CMAKE_FORCE_CXX_COMPILER(icpc "Intel C++ Compiler")
CMAKE_FORCE_Fortran_COMPILER(ifort "Intel Fortran Compiler")
```

Lastly, when using the Intel compiler, we add:

```
if(${CMAKE_Fortran_COMPILER_ID} STREQUAL "Intel")
    if(NOT CMAKE_Fortran_FLAGS_RELEASE)
        set(CMAKE_Fortran_FLAGS_RELEASE "-O2 -xhost" CACHE STRING "" FORCE)
    endif()
    set(CMAKE_Fortran_FLAGS_DEBUG
        "${CMAKE_Fortran_FLAGS_DEBUG} -check noarg_temp_created -C -traceback" CA\
        CHE STRING "" FORCE)
    set(CMAKE_Fortran_FLAGS_DEBUGHEAVY
        "${CMAKE_Fortran_FLAGS_DEBUG} -check noarg_temp_created -fpe0 -warn align\
        ments -warn declarations -warn general -warn interfaces -warn truncated_source \
        -warn uncalled -warn uninitialized -warn usage -common_args -warn unused -fp-st\
        ack-check -check bounds -check uninit -check format" CACHE STRING "" FORCE)
    mark_as_advanced(CMAKE_Fortran_FLAGS_DEBUGHEAVY)
endif()
```

问题15: 如何支持**cuda**编程

cmake有一套很好的指令来支持这个。

下面是一个写得很好的例子

```
[cpp]
```

```
01. PROJECT( JTianLingBLog )
02. CMAKE_MINIMUM_REQUIRED(VERSION 2.8)
03. INCLUDE_DIRECTORIES(./dependes/windows/include/)
04. LINK_DIRECTORIES(./dependes/windows/lib/)
```

```
05. FILE(GLOB_RECURSE SRC_LIST "*.cpp")
06. FOREACH(src ${SRC_LIST})
07.     IF(NOT ( (src MATCHES ".*2009-10-20.*") OR (src MATCHES ".*2009-9-28.*") ) )
08.         MESSAGE( ${src} Finded)
09.         STRING( REGEX REPLACE ".*/(.*)\\.cpp$" "\\1" prjName ${src} )
10.         MESSAGE( ${prjName} Added)
11.         ADD_EXECUTABLE( ${prjName} ${src} )
12.
13.         IF (NOT (prjName STREQUAL mvarray) )
14.             SET_TARGET_PROPERTIES(${prjName} PROPERTIES WIN32_EXECUTABLE "true")
15.             ENDEF(NOT (prjName STREQUAL mvarray) )
16.
17.         ENDEF(NOT ( (src MATCHES ".*2009-10-20.*") AND (src MATCHES ".*2009-9-28.*") ) )
18.     ENDFOREACH(src)
```

问题16: 明明添加了库，可是cmake为什么不去找，或者修改了确不更新

删除cmakecache文件，再cmake一次就可以了。

问题17: cmake的变量可以跨文件使用吗？

可以， 建议给project bin name之类的常用内容，命名成变量，这样方便以后修改，跨文件使用有一个前提那就是这些变量要定义在add_subdirectory之前，只有这样才能被下面的cmakelists找到。

问题18: cmake如何同时支持32 bit和 64 bit系统？

```
if( CMAKE_SIZEOF_VOID_P EQUAL 8 )
    set( BOOST_LIBRARY "/boost/win64/lib" )
else( CMAKE_SIZEOF_VOID_P EQUAL 8 )
    set( BOOST_LIBRARY "/boost/win32/lib" )
endif( CMAKE_SIZEOF_VOID_P EQUAL 8 )
set( CMAKE_EXE_LINKER_FLAGS ${BOOST_LIBRARY} )
```

顶

0

踩

0

上一篇

STL iterator的自己实现

下一篇

并行计算 写给自己的困惑

参考知识库



Java EE知识库
1453 关注 | 618 收录



Java SE知识库
9664 关注 | 454 收录



Java Web知识库
9985 关注 | 1050 收录