## 二. 简单介绍

cmake(http://www.cmake.org/)的介绍，可以 Google。

接触 C++项目的人都知道，现在存在很多组织源代码进行编译的工具，windows 平台下的 nmake，visual studio（不同版本的 sln project 文件）以及 Eclipse 的 CDT。cmake 所处的位置，更像是这些已有工具的管理工具，它可以根据 cmakelists 文件来生成相应的 makefile，project 文件等等。

使用 cmake 的好处有很多。编写一次配置文件，就可以在不同的开发平台，不同的编译环境下使用。强大的定制功能，弥补很多工具自身的不足。自动化管理依赖，节省开发人员的时间和编译时间。等等。

当然也有他的不足。每一个工具都是有自己的局限性。学习曲线比较陡峭。编译过程变的稍微复杂，需要首先生成中间格式的文件，然后才能编译生成最终的文件。学习资料很少很散。

有好处有坏处，需要的就是权衡。在我看来，cmake 提供的功能比较诱人，将代码的配置信息与 build 环境剥离。这样做为程序员提供了很大的灵活性。

下载安装都很简单，直接到 cmake 的主页上下载安装包，按提示操作。

## 三. 文件目录结构

在每一层目录下创建一个 cmakelists.txt 的文件，父目录的 cmakelists 文件可以指定包含的子目录，子目录的 cmakelists 则管理该目录下所有的 sourcecode。目录可以多层嵌套，组织比较灵活，但是不建议目录层数太深。

|  |  |  |
|---|---|---|
| Dir | Helloworld | The project |

|　　　　　File　cmakelists.txt　　　　　　　　　　　　　　　　　　　　　　　　　The project main cmakelists.txt, which should define the whole project’s properties and include sub dir

|　　　　　Dir　HelloworldTest　　　　　　　　　　　　　　　　　　　　　　　　　The app dir, which contains the main function

|　　　　　|　　　　　File cmakelists.txt　　　　　　　　　　　　　　　　　　　　The app cmakelists.txt, which defines all the source files, compiler flags and link flags.

|　　　　　|　　　　　File main.cpp　　　　　　　　　　　　　　　　　　　　　　　　　　The cpp file

|　　　　　Dir　Hellowould　　　　　　　　　　　　　　　　　　　　　　　　　　　　　The library dir

|　　　　　|　　　　　File cmakelists.txt　　　　　　　　　　　　　　　　　　　　The library cmakelists.txt, which should define all source files and build flags

|　　　　　|　　　　　File helloworld.h　　　　　　　　　　　　　　　　　　　　　　　header file

|　　　　　|　　　　　File helloworld.cpp　　　　　　　　　　　　　　　　　　　　cpp file

从上边的目录结构可以看出，每一层目录下边都有一个 cmakelists.txt。

文件内容：

Helloworld/cmakelists.txt

```
CMAKE_MINIMUM_REQUIRED( VERSION 2.8 )

project( Helloworld )

set( CMAKE_EXE_LINKER_FLAGS "${CMAKE_EXE_LINKER_FLAGS} /machine:x86" )

add_subdirectory( Helloworld )
add_subdirectory( Helloworldtest )
```

Helloworld/Helloworld/cmakelists.txt

```
CMAKE_MINIMUM_REQUIRED( VERSION 2.8 )

set( HelloWorldSrc helloworld.cpp helloworld.h )

add_library( Helloworld STATIC ${HelloWorldSrc} )
```

Helloworld/HelloworldTest/cmakelists.txt

```
CMAKE_MINIMUM_REQUIRED( VERSION 2.8 )
set( HelloWorldTestSrc main.cpp)

include_directories( "${Helloworld_SOURCE_DIR}/Helloworld" )

add_executable( HelloworldTest ${HelloWorldTestSrc} )
```

```
target_link_libraries(HelloworldTest Helloworld )
add_dependencies( HelloworldTest Helloworld )
```

以上是最简单的 cmake 应用，使用了很基础的 cmake 命令，这里简单描述一下：

cmake_minimum_required( VERSION 2.8 ) ：表示当前 cmakelists 文件要求 cmake 版本最低为 2.8

project( Helloworld ) ：整个项目的名字，在我们的项目中，Helloworld。这个在 visual studio 中对应 solution 的名字。

set( NAME VAL1 VAL2 ) ：设置变量 NAME，值为 VAL1;VAL2 的列表。CMAKE 中列表使用;分隔。如果需要空格分隔的结果，可以用双引号转换。

include_directories()：包含一个目录

add_executable(): 增加一个编译单元，目标文件为可执行文件，在 windows 上则为 exe 文件。

add_library()：添加一个编译单元，目标文件为库。

以上命令的详细解释，可以参考 cmake 的帮助文档。

## 四。外部库的引入

所有源文件都存放在一个目录下的项目，现实生活中这样的情况并不多见。最经常见到的情形是，我们自己的项目 A，依赖于第三方库 B, C, D。这里的第三方库包含类似于 boost 这样的公共库，也包含我们自己编写的独立库。

未完

## 附录：Visual Studio 属性与对应 cmakelists 实现方法：

注意：此附录仅为 beta 版本，有些条目是推导出来的。

| **Visual Studio** | | | **Cmake** |
|---|---|---|---|
| Project Reference | | | add_dependencies() |
| General/Output Directory | | | |
| | | | |
| | | | |
| Debugging/Working Directory | | | CMAKE can’t set this value, because the info not stored in project file but some intermedia file generated by visual studio |
| | | | |

| | | | |
|---|---|---|---|
| c/c++ | General | Additional Include Directories | include_directories() |
| | | Resolve #using References | Don’ t know |
| | | Debug Information Format | CMAKE FAQ<br>set(CMAKE_C_FLAGS_DEBUG_INIT ″/D_DEBUG /MTd /Zi /Ob0 /Od /RTC1″)<br>set(CMAKE_C_FLAGS_MINSIZEREL_INIT ″/MT /O1 /Ob1 /D NDEBUG″)<br>set(CMAKE_C_FLAGS_RELEASE_INIT ″/MT /O2 /Ob2 /D NDEBUG″)<br>set(CMAKE_C_FLAGS_RELWITHDEBINFO_INIT ″/MT /Zi /O2 /Ob1 /D NDEBUG″)<br>Change the default flags for specific config. |
| | | Common language runtime support | set_target_properties( target PROPERTIES COMPILE_FLAGS “/clr” )<br>set_target_properties( target PROPERTIES COMPILE_FLAGS “/clr:pure” )<br>set_target_properties( target PROPERTIES COMPILE_FLAGS “/clr:safe” )<br>set_target_properties( target PROPERTIES COMPILE_FLAGS “/clr:oldSynax” ) |

| | | | |
|---|---|---|---|
| | | Suppress Startup Banner | set_target_properties( target PROPERTIES COMPILE_FLAGS  “/nologo”  ) |
| | | Warning Level | set_target_properties( target PROPERTIES COMPILE_FLAGS  “/W0”  )<br>set_target_properties( target PROPERTIES COMPILE_FLAGS  “/W1”  )<br>set_target_properties( target PROPERTIES COMPILE_FLAGS  “/W2”  )<br>set_target_properties( target PROPERTIES COMPILE_FLAGS  “/W3”  )<br>set_target_properties( target PROPERTIES COMPILE_FLAGS ″/W4″ )<br>set_target_properties( target PROPERTIES COMPILE_FLAGS  “/Wall”  ) |
| | | Treat Warnings As Errors | set_target_properties( target PROPERTIES COMPILE_FLAGS  “/WX-″ ) #No<br>set_target_properties( target PROPERTIES COMPILE_FLAGS  “/WX″ )   #Yes |
| | | Multi-processor Compilation | set_target_properties( target PROPERTIES COMPILE_FLAGS  “/MP″ )   #Yes<br>#Don’ t set means No |
| | | Use Unicode For Assembler Listing | set_target_properties( target PROPERTIES COMPILE_FLAGS  “/FAu″ ) #yes<br>#Don’ t set means no |

| | | | |
|---|---|---|---|
| | Optimization | Optimization | set(CMAKE_C_FLAGS_DEBUG_INIT "/D_DEBUG /MTd /Zi /Ob0 **/Od** /RTC1") set(CMAKE_C_FLAGS_MINSIZEREL_INIT "/MT **/O1** /Ob1 /D NDEBUG") set(CMAKE_C_FLAGS_RELEASE_INIT "/MT **/O2** /Ob2 /D NDEBUG") set(CMAKE_C_FLAGS_RELWITHDEBINFO_INIT "/MT /Zi **/O2** /Ob1 /D NDEBUG") Change the default flags for specific config. |
| | | Inline Function Expansion | set(CMAKE_C_FLAGS_DEBUG_INIT "/D_DEBUG /MTd /Zi /**Ob0** /Od /RTC1") set(CMAKE_C_FLAGS_MINSIZEREL_INIT "/MT /O1 /**Ob1** /D NDEBUG") set(CMAKE_C_FLAGS_RELEASE_INIT "/MT /O2 /**Ob2** /D NDEBUG") set(CMAKE_C_FLAGS_RELWITHDEBINFO_INIT "/MT /Zi /O2 /**Ob1** /D NDEBUG") Change the default flags for specific config. |
| | | Enable Intrinsic Functions | set_target_properties( target PROPERTIES COMPILE_FLAGS "/Oi" ) #yes #Don't set means no |
| | | Favor Size or Speed | set_target_properties( target PROPERTIES COMPILE_FLAGS "/Os" ) #size |

| | | | |
|---|---|---|---|
| | | | set_target_properties( target PROPERTIES COMPILE_FLAGS “/Ot” ) #speed<br>#Don’ t set means neither |
| | | Omit Frame Pointers | set_target_properties( target PROPERTIES COMPILE_FLAGS “/Oy-” ) #no<br>set_target_properties( target PROPERTIES COMPILE_FLAGS “/Oy” ) #yes |
| | | Enable Fiber-safe Optimizations | set_target_properties( target PROPERTIES COMPILE_FLAGS “/GT” ) #yes<br>#not setting means no |
| | | Whole Program Optimization | set_target_properties( target PROPERTIES COMPILE_FLAGS “/GL” ) #yes<br>#not setting means no |
| | Preprocessor | Preprocessor Definitions | set_target_properties( target PROPERTIES COMPILE_DEFINITIONS DEFNAME=DEFVAL )<br>set_source_files_properties( filename.cpp PROPERTIES COMPILE_DEFINITIONS DEFNAME=DEFVAL ) |
| | | Undefine Preprocessor Definitions | set_target_properties( target PROPERTIES COMPILE_FLAGS “/UDEFNAME” ) |
| | | Undefine All Preprocessor Definitions | set_target_properties( target PROPERTIES COMPILE_FLAGS “/u” ) |

| | | | |
|---|---|---|---|
| | | Ignore Standard Include Path | set_target_properties( target PROPERTIES COMPILE_FLAGS “/X″ ) |
| | | Preprocess to a File | set_target_properties( target PROPERTIES COMPILE_FLAGS “/P″ ) |
| | | Preprocess Suppress Line Numbers | set_target_properties( target PROPERTIES COMPILE_FLAGS “/EP″ ) |
| | | Keep Comments | set_target_properties( target PROPERTIES COMPILE_FLAGS “/C″ ) |
| | Code Generation | Enable String Pooling | set_target_properties( target PROPERTIES COMPILE_FLAGS “/GF″ ) #yes<br>set_target_properties( target PROPERTIES COMPILE_FLAGS “/GF-″ ) #no |
| | | Enable Minimum Rebuild | set_target_properties( target PROPERTIES COMPILE_FLAGS “/Gm″ ) #yes<br>set_target_properties( target PROPERTIES COMPILE_FLAGS “/Gm-″ )#no |
| | | Enable C++ Exceptions | set_target_properties( target PROPERTIES COMPILE_FLAGS “/EHsc″ ) #yes<br>set_target_properties( target PROPERTIES COMPILE_FLAGS “/EHa″ ) #yes, with SEH exceptions<br>set_target_properties( target PROPERTIES |

|  |  |  |  |
|---|---|---|---|
|  |  |  | COMPILE_FLAGS “/EHs” ) #yes, with extern C functions<br>#not setting means no |
|  |  | Smaller Type Check | set_target_properties( target PROPERTIES COMPILE_FLAGS “/RTCc” ) #yes<br>#not setting means no |
|  |  | Basic Runtime Checks | set_target_properties( target PROPERTIES COMPILE_FLAGS “/RTCs” ) #Stack frame check<br>set_target_properties( target PROPERTIES COMPILE_FLAGS “/RTCu” ) #Uninitialized Variable<br>set_target_properties( target PROPERTIES COMPILE_FLAGS “/TRC1” ) #Both<br>#not setting means no |
|  |  | Runtime Library | CMAKE FAQ<br>set(CMAKE_C_FLAGS_DEBUG_INIT ″/D_DEBUG **/MTd** /Zi /Ob0 /Od /RTC1″)<br>set(CMAKE_C_FLAGS_MINSIZEREL_INIT ″**/MT** /O1 /Ob1 /D NDEBUG″)<br>set(CMAKE_C_FLAGS_RELEASE_INIT ″**/MT** /O2 /Ob2 /D NDEBUG″)<br>set(CMAKE_C_FLAGS_RELWITHDEBINFO_INIT ″**/MT** /Zi /O2 /Ob1 /D NDEBUG″) |

| | | | |
|---|---|---|---|
| | | | Change the default flags for specific config. |
| | | Struct Member Alignment | set_target_properties( target PROPERTIES COMPILE_FLAGS “/Zp1″ ) set_target_properties( target PROPERTIES COMPILE_FLAGS “/Zp2″ ) set_target_properties( target PROPERTIES COMPILE_FLAGS “/Zp4″ ) set_target_properties( target PROPERTIES COMPILE_FLAGS “/Zp8″ )set_target_properties( target PROPERTIES COMPILE_FLAGS “/Zp16″ ) |
| | | Buffer Security Check | set_target_properties( target PROPERTIES COMPILE_FLAGS “/GS″ ) #yes set_target_properties( target PROPERTIES COMPILE_FLAGS “/GS-″ ) #no |
| | | Enable Function-Level Linking | set_target_properties( target PROPERTIES COMPILE_FLAGS “/Gy″ ) #yes set_target_properties( target PROPERTIES COMPILE_FLAGS “/Gy-″ ) #no |
| | | Enable Enhaunced Instruction Set | set_target_properties( target PROPERTIES COMPILE_FLAGS “/arch:SSE″ ) set_target_properties( target PROPERTIES COMPILE_FLAGS “/arch:SSE2″ ) |

| | | | |
|---|---|---|---|
| | | Floating Point Model | set_target_properties( target PROPERTIES COMPILE_FLAGS “/fp:precise″ ) set_target_properties( target PROPERTIES COMPILE_FLAGS “/fp:strict″ ) set_target_properties( target PROPERTIES COMPILE_FLAGS “/fp:fast″ ) |
| | | Enable Floating Point Exceptions | set_target_properties( target PROPERTIES COMPILE_FLAGS “/fp:except″ ) |
| | | Create Hotpatchable Image | set_target_properties( target PROPERTIES COMPILE_FLAGS “/hotpatch″ ) |
| | Language | Disable Language Extensions | set_target_properties( target PROPERTIES COMPILE_FLAGS “/Za″ ) |
| | | Treat Wchar_t As Built in Type | set_target_properties( target PROPERTIES COMPILE_FLAGS “/Zc:wchar_t″ )#yes set_target_properties( target PROPERTIES COMPILE_FLAGS “/Zc:wchar_t-″ ) #no |
| | | Force Conformance in For Loop Scope | |

| | | | |
|---|---|---|---|
| | | Enable Run-Time Type Information | set_target_properties( target PROPERTIES COMPILE_FLAGS “/GR″ ) #yes<br>set_target_properties( target PROPERTIES COMPILE_FLAGS “/GR-″ ) #no |
| | | Open MP Support | set_target_properties( target PROPERTIES COMPILE_FLAGS “/openmp″ )#yes<br>set_target_properties( target PROPERTIES COMPILE_FLAGS “/openmp-″ )#no |
| | PreCompiled Header | PreCompiled Header | set_target_properties( target PROPERTIES COMPILE_FLAGS ″/Yc″ ) #create<br>set_target_properties( target PROPERTIES COMPILE_FLAGS ″/Yu″ ) #use<br>#not setting means no |
| | | PreCompiled Header File | set_target_properties( target PROPERTIES COMPILE_FLAGS ″/Ycstdafx.h″ )<br>set_target_properties( target PROPERTIES COMPILE_FLAGS ″/Yustdafx.h″ ) |
| | | Precompiled Header output File | set_target_properties( target PROPERTIES COMPILE_FLAGS ″/FpPathAndName.pch″ ) |
| | Output Files | Expand Attributed Source | set_target_properties( target PROPERTIES COMPILE_FLAGS ″/Fx″ ) |

| | | | |
|---|---|---|---|
| | | Assembler Output | set_target_properties( target PROPERTIES COMPILE_FLAGS ″/FA″ )<br>set_target_properties( target PROPERTIES COMPILE_FLAGS ″/FAc″ )<br>set_target_properties( target PROPERTIES COMPILE_FLAGS ″/FAs″ )<br>set_target_properties( target PROPERTIES COMPILE_FLAGS ″/FAcs″ )<br>#not setting means no list |
| | | ASM List Location | s~~et_target_properties( target PROPERTIES COMPILE_FLAGS ″/FaDebug″ )~~ |
| | | Object File name | set_target_properties( target PROPERTIES COMPILE_FLAGS ″/FoName.obj″ ) |
| | | Program DataBase File Name | s~~et_target_properties( target PROPERTIES COMPILE_FLAGS ″/FdC:/Debug/good.pdb″ )~~ |
| | | Generate XML Documentation Files | set_target_properties( target PROPERTIES COMPILE_FLAGS ″/doc″ ) |
| | | XML Documentation Filename | set_target_properties( target PROPERTIES COMPILE_FLAGS ″/docDocument.xml″ ) |
| | Browse Information | Enable Browse Information | set_target_properties( target PROPERTIES COMPILE_FLAGS ″/FR″ ) |

| | | | |
|---|---|---|---|
| | | Browse Information File | set_target_properties( target PROPERTIES COMPILE_FLAGS "/FRfilename" ) |
| | Advanced | Calling Convention | set_target_properties( target PROPERTIES COMPILE_FLAGS "/Gd" ) #_cdecl<br>set_target_properties( target PROPERTIES COMPILE_FLAGS "/Gr" )   #_fastcall<br>set_target_properties( target PROPERTIES COMPILE_FLAGS "/Gz" ) #_stdcall |
| | | Compile As | set_target_properties( target PROPERTIES LINKER_LANGUAGE "CXX" ) #C++<br>set_target_properties( target PROPERTIES LINKER_LANGUAGE "C" ) #C<br>or<br>set_target_properties( target PROPERTIES COMPILE_FLAGS "/TP" ) #CXX<br>set_target_properties( target PROPERTIES COMPILE_FLAGS "/TC" ) #C |
| | | Disable Specific Warnings | set_target_properties( target PROPERTIES COMPILE_FLAGS "/wd4710" ) |
| | | Forced Include File | set_target_properties( target PROPERTIES COMPILE_FLAGS "/FIinclude.h" ) |

|  |  |  |  |
|---|---|---|---|
|  |  | Forced #using File | set_target_properties( target PROPERTIES COMPILE_FLAGS ″/FUname″ ) |
|  |  | Show Includes | set_target_properties( target PROPERTIES COMPILE_FLAGS ″/showIncludes″ ) |
|  |  | Use full Paths | set_target_properties( target PROPERTIES COMPILE_FLAGS ″/FC″ ) |
|  |  | Omit Default Library name | set_target_properties( target PROPERTIES COMPILE_FLAGS ″/ZI″ ) |
|  |  | Internal Compiler Error Reporting | set_target_properties( target PROPERTIES COMPILE_FLAGS ″/errorReport:queue″ )<br>set_target_properties( target PROPERTIES COMPILE_FLAGS ″/errorReport:none″ )<br>set_target_properties( target PROPERTIES COMPILE_FLAGS ″/errorReport:prompt″ )<br>set_target_properties( target PROPERTIES COMPILE_FLAGS ″/errorReport:send″ ) |
|  |  | Treat Specific Warnings as Errors | Don’t know |
| Linker | General | Output File | #normal case<br>set_target_properties( target PROPERTIES OUTPUT_NAME ″Helloworld″ )<br>set_target_properties( target PROPERTIES |

| | | | |
|---|---|---|---|
| | | | PREFIX ″lib″ )<br>set_target_properties( target PROPERTIES SUFFIX ″lib″ )<br><br>#for debug version<br>set_target_properties( target PROPERTIES DEBUG_OUTPUT_NAME ″Helloworld″ )<br>set_target_properties( target PROPERTIES DEBUG_PREFIX ″lib″ )<br>set_target_properties( target PROPERTIES DEBUG_SUFFIX ″lib″ )<br><br>#For dlls<br>set_target_properties( target PROPERTIES OUTPUT_NAME ″Helloworld″ )<br>set_target_properties( target PROPERTIES IMPORT_PREFIX ″lib″ )<br>set_target_properties( target PROPERTIES IMPORT_SUFFIX ″lib″ )<br>set_target_properties( target PROPERTIES PREFIX ″bin″ )<br>set_target_properties( target PROPERTIES SUFFIX ″dll″ )<br><br>#output path |

|  |  |  |  |
|---|---|---|---|
|  |  | Show Progress | set_target_properties( target PROPERTIES LINK_FLAGS ″/VERBOSE″ )<br>set_target_properties( target PROPERTIES LINK_FLAGS ″/VERBOSE:Lib″ )<br>set_target_properties( target PROPERTIES LINK_FLAGS ″/VERBOSE:ICF″ )<br>set_target_properties( target PROPERTIES LINK_FLAGS ″/VERBOSE:REF″ )<br>set_target_properties( target PROPERTIES LINK_FLAGS ″/VERBOSE:SAFESEH″ )<br>set_target_properties( target PROPERTIES LINK_FLAGS ″/VERBOSE:CLR″ )<br>#not setting means no |
|  |  | Version | set_target_properties( target PROPERTIES VERSION 0.1.2.3) |
|  |  | Enable Incremental Linking | set_target_properties( target PROPERTIES LINK_FLAGS ″/INCREMENTAL″ )<br>set_target_properties( target PROPERTIES LINK_FLAGS ″/INCREMENTAL:NO″ )<br><br>set( CMAKE_EXE_LINKER_FLAGS_DEBUG ″/INCREMENTAL″ )<br>set( CMAKE_EXE_LINKER_FLAGS_DEBUG ″/INCREMENTAL:NO″ ) |

| | | | |
|---|---|---|---|
| | | Suppress Startup Banner | set_target_properties( target PROPERTIES LINK_FLAGS ″/NOLOGO″ ) |
| | | Ignore Import Library | Don’t know |
| | | Register Output | Don’t know |
| | | Per-user Redirection | Don’t know |
| | | Additional Library Directories | link_directories( dir1 dir2 )<br><br>set_target_properties( target PROPERTIES LINK_FLAGS ″/LIBPATH:dir1 /LIBPATH:dir2″ ) |
| | | Link Library Dependencies | Don’t know |
| | | Use Library Dependency Inputs | Don’t know |
| | | Link Status | set_target_properties( target PROPERTIES LINK_FLAGS ″/LTCG:STATUS″ )<br>set_target_properties( target PROPERTIES LINK_FLAGS ″/LTCG:NOSTATUS″ ) |

| | | | |
|---|---|---|---|
| | | Prevent DLL Binding | set_target_properties( target PROPERTIES LINK_FLAGS ″/ALLOWBIND:NO″ ) set_target_properties( target PROPERTIES LINK_FLAGS ″/ALLOWBIND:YES″ ) |
| | | Treat Linker Warnings As Errors | set_target_properties( target PROPERTIES LINK_FLAGS ″/WX″ ) |
| | | Force File Output | set_target_properties( target PROPERTIES LINK_FLAGS ″/FORCE″ ) |
| | | Create Hot Patchable Image | set_target_properties( target PROPERTIES LINK_FLAGS ″/FUNCTIONPADMIN″ ) set_target_properties( target PROPERTIES LINK_FLAGS ″/FUNCTIONPADMIN:16″ ) #Itanium only set_target_properties( target PROPERTIES LINK_FLAGS ″/FUNCTIONPADMIN:6″ ) #x64 only set_target_properties( target PROPERTIES LINK_FLAGS ″/FUNCTIONPADMIN:5″ ) #x86 only |
| | | Specify Section Attributes | Don’t know |

| | | | |
|---|---|---|---|
| | Input | Additional Dependancies | target_link_libraries( target item1 item2 ) |
| | | Ignore All Default Libraries | set_target_properties( target PROPERTIES LINK_FLAGS "/NODEFAULTLIB" ) |
| | | | |
| | | | |
| | | | |
| Put files into folders | | | source_group( header FILES includeme.h ) |
| | | | |
| | | | |
| | | | |

结论：CMAKE 默认会修改的选项有时候是没有办法再次修改的。CMAKE 默认不修改的选项大部分情况都可以自己定义。