CTK编译教程(64位环境 Windows + Qt + MinGW或MSVC + CMake)

后端 (/categories/back/) ctk (/tags/ctk/) qt (/tags/qt/) 2021/02/24

CTK 简介

当前，CTK 工作的主要范围包括：

- DICOM (http://www.commontk.org/index.php/Documentation/Dicom_Overview)：提供了从 PACS 和本地数据库中查询和检索的高级类。包含 Qt 部件，可以轻松地设置服务器连接，并发送查询和查看结果。
- DICOM Application Hosting (http://www.commontk.org/index.php/Documentation/DicomApplicationHosting)：目标是创建 DICOM Part 19 Application Hosting specifications 的 C++ 参考实现。它提供了用于创建主机和托管应用程序的基础设。
- Widgets (http://www.commontk.org/index.php/Documentation/Widgets)：用于生物医学成像应用的 Qt Widgets 集合。
- Plugin Framework (http://www.commontk.org/index.php/Documentation/Plugin_Framework)：用于 C++ 的动态组件系统，以 OSGi 规范为模型。它支持一个开发模型，在这个模型中，应用程序（动态地）由许多不同（可重用的）组件组成，遵循面向服务的方法。
- Command Line Interfaces (http://www.commontk.org/index.php/Documentation/Command_Line_Interface)：一种允许将算法编写为自包含可执行程序的技术，可以在多个终端用户应用程序环境中使用，而无需修改。

Qt Creator 下 CMake 配置

CMake安装包自行到CMake官网 (https://cmake.org/download/)进行下载安装，我下载的是cmake-3.19.5-win64-x64.zip免安装版，下载成功后解压缩后放在适当的位置。

之后打开 Qt Creator，在菜单中选择：【工具】—>【选项】 —>【Kits】，在弹出的对话框中选择进入“CMake”标签页，由于选择的CMake版本不是安装版，因此这里CMake 不会被自动检测出来，需要手动配置，配置方法如下：

```
1.先点击ADD按钮；

2.然后在交互界面填写下面信息：
name: CMake #取自己喜欢的名字
path: C:\Qt\cmake\bin\cmake.exe #CMake的位置
其它缺省

3.最后点击“OK”按钮进行保存。
```

进入“Kits”标签页，进行CMake配置，具体配置情况如下所示：

- MSVC2019

```
1.先点击ADD按钮；

2.然后在交互界面填写下面信息：
name: CMake #取自己喜欢的名字
#下面信息根据自己的环境进行选取
compiler C: Microsoft Visual C++ Compiler 16.6.30320.27(amd64)
compiler C++: Microsoft Visual C++ Compiler 16.6.30320.27(amd64)
Qt version: Qt 5.15.1 MSVC2019 64bit
CMake generator: <none>-NMake Makefiles,Platform:<none>,Toolset:<none>
其它缺省

3.最后点击“OK”按钮进行保存。
```

- MinGW

  不建议用MinGW编译。***笔者使用MingW编译出的CTK，无法使用***

```
1.先点击ADD按钮；

2.然后在交互界面填写下面信息：
name: CMake #取自己喜欢的名字
#下面信息根据自己的环境进行选取
compiler C: MinGW 8.1.0 64-bit for C
compiler C++: MinGW 8.1.0 64-bit for C++
Debugger: GNU gdb 8.1 for MinGW 8.1.0 64-bit
Qt version: Qt 5.15.1 MinGW 64-bit
CMake generator: <none>-MinGW Makefiles,Platform:<none>,Toolset:<none>
其它缺省

3.最后点击“OK”按钮进行保存。
```

CTK源码配置

1.1 源码下载

从Github-CTK (https://github.com/commontk/CTK)上获取源码，然后解压缩。

1.2 Qt兼容性配置

为了避免Qt版本兼容性问题，需要将CTKmaster/CMake/ctkMacroSetupQt.cmake 文件中CTKQTVERSION 由 4 改为 5。修改后可以避免如下编译错误。

```
set(CTK_QT_VERSION "5" CACHE STRING "Expected Qt version")
```

1.3 CTKData配置

手动从https://github.com/commontk/CTKData (https://github.com/commontk/CTKData) 上下载 CTKData，然后将 CTKData 解压后放到一个固定的位置（我是放在CTKmaster根目录下），并在CTKmaster/CMakeExternals/CTKData.cmake 中对CTKData的路径进行配置。

```
#set CTKData_path
set(CTKData_DIR ${CMAKE_CURRENT_SOURCE_DIR}/CTKData)
#上面的是新增加，下面的不管它，只起定位参考用

# Sanity checks
if(DEFINED CTKData_DIR AND NOT EXISTS ${CTKData_DIR})
  message(FATAL_ERROR "CTKData_DIR variable is defined but corresponds to non-existing
endif()
```

1.4 生成库相关的开关配置

由于默认配置下，有些库/插件（CTKPluginFramework.dll、CTKWidgets.dll以及其他库）是不会自动生成的，因为我们需要在 CTKmaster/CMakeLists.txt文件中修改的相应的开关设置，将OFF改为ON。

```
ctk_lib_option(Core
               "Build the Core library" ON)

ctk_lib_option(PluginFramework
               "Build the Plugin Framework" ON
               CTK_ENABLE_PluginFramework)

ctk_lib_option(Widgets
               "Build the Widgets library" ON
               CTK_ENABLE_Widgets OR (CTK_ENABLE_DICOMApplicationHosting AND CTK_BUI
```

```
foreach(_plugin ${plugin_list})
  ctk_plugin_option(${_plugin} "Build the ${_plugin} plugin." ON)
endforeach()
```

不对CTK进行调试，为了加快编译速度，要关闭调试BUILD_TESTING，同时不关闭编译还可能编译出错。

```
option(BUILD_TESTING "" OFF)
#上面的是新增加，下面的不管它，只起定位参考用
include(CTest)
```

1.5 安装配置

默认配置下CTK 在编译时是不会自动安装的， 因此需要手动在 CTKmaster/SuperBuild.cmake 文件中进行安装步骤及安装路径的配置。

```
#add install command
set(_INSTALL_CMD nmake install)
#上一行，MinGW用mingw32-make，MSVC2019 64位用nmake,MSVC2019 32位用make
set(_INSTALL_DIR ${CTK_BINARY_DIR}/CTKInstall)
#上面的是新增加

ExternalProject_Add(${proj}
  ${${proj}_EP_ARGS}
  DOWNLOAD_COMMAND ""
  CMAKE_CACHE_ARGS
    -DCTK_SUPERBUILD:BOOL=OFF
    -DCTK_SUPERBUILD_BINARY_DIR:PATH=${CTK_BINARY_DIR}
    -DCMAKE_C_COMPILER:FILEPATH=${CMAKE_C_COMPILER}
    -DCMAKE_CXX_COMPILER:FILEPATH=${CMAKE_CXX_COMPILER}
    -DCMAKE_CXX_FLAGS_INIT:STRING=${CMAKE_CXX_FLAGS_INIT}
    -DCMAKE_C_FLAGS_INIT:STRING=${CMAKE_C_FLAGS_INIT}
    -DCMAKE_INSTALL_PREFIX:PATH=${_INSTALL_DIR} #此行有修改
    -DCMAKE_MACOSX_RPATH:BOOL=${CMAKE_MACOSX_RPATH}
    ${ep_cxx_standard_arg}
  SOURCE_DIR ${CTK_SOURCE_DIR}
  BINARY_DIR ${CTK_BINARY_DIR}/CTK-build
  INSTALL_COMMAND ${_INSTALL_CMD} #此行有修改
```

CTK编译

进行完上述所有配置后就可以开始进行编译，***以“管理员权限”运行QtCreator***，使用Qt Creator 打开CTKmaster/CMakeLists.txt， Kit 选择刚刚配置好的CMake。

进行编译，右键项目：“执行 CMake >构建”后发现左侧的项目结构已经被展开了，经过漫长的编译过程，CTK 编译成功！***必须以“管理员权限”运行QtCreator进行编译，可能会因为QtCreator权限不够导致编译失败。**

在CTKbuild/bin目录下查看已经编译好的库，可以看到所有的库都已经生成。

进入安装目录 CTKInstall/include和 CTKInstall/lib下查看发现成功安装。

‹ 上一篇 (/archives/2021/03/100644.html)

下一篇 › (/archives/2021/02/100642.html)