Visual Studio 2015+cmake 编译安装 MD 版的 DCMTK

目录

一、 编译环境

本文的编译机器是 Windows10 64 位系统，使用 VS2015 和 cmake3.7.2 编译 DCMTK 源码为 64 位包（运行时库为 MD 或 MDd）。

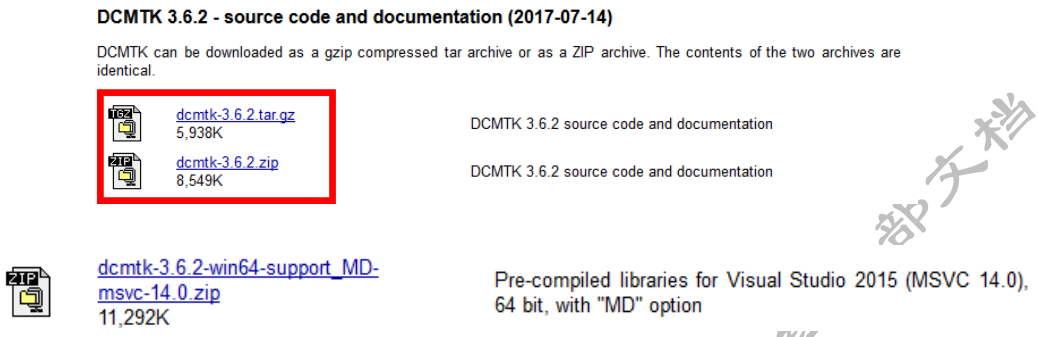
注（引用）： /MT 和/MTd 表示采用多线程 CRT 库的静态 lib 版本。该选项会在编译时将运行时库以静态 lib 的形式完全嵌入。该选项生成的可执行文件运行时不需要运行时库 dll 的参加，会获得轻微的性能提升，但最终生成的二进制代码因链入庞大的运行时库实现而变得非常臃肿。当某项目以静态链接库的形式嵌入到多个项目，则可能造成运行时库的内存管理有多份，最终将导致致命的“Invalid Address specified to RtlValidateHeap”问题。另外托管 C++ 和 CLI 中不再支持/MT 和/MTd 选项。

/MD 和/MDd 表示采用多线程 CRT 库的动态 dll 版本，会使应用程序使用运行时库特定版本的多线程 DLL。链接时将按照传统 VC 链接 dll 的方式将运行时库 MSVCRxx.DLL 的导入库 MSVCRT.lib 链接，在运行时要求安装了相应版本的 VC 运行时库可再发行组件包（当然把这些运行时库 dll 放在应用程序目录下也是可以的）。 因/MD 和/MDd 方式不会将运行时库链接到可执行文件内部，可有效减少可执行文件尺寸。当多项目以 MD 方式运作时，其内部会采用同一个堆，内存管理将被简化，跨模块内存管理问题也能得到缓解。

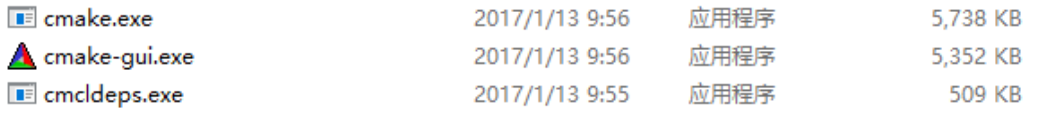
结论：/MD 和/MDd 将是潮流所趋，/ML 和/MLd 方式请及时放弃，/MT 和/MTd 在非必要时最好也不要采用了。

二、 下载文件

a) DCMTK 包 dcmtk-3.6.2.tar.gz：
直接下载链接或下载最新版 http://www.dcmtk.org/dcmtk.php.en
b) 支持库包 dcmtk-3.6.2-win64-support_MD-msvc-14.0.zip：
直接下载链接或下载最新版 http://www.dcmtk.org/dcmtk.php.en

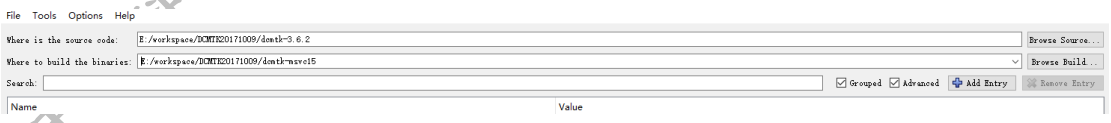


Cmake3.10 ： 直接下载，并解压，进入 bin 文件夹，点击 cmake-gui.exe.
官网：https://cmake.org/download/

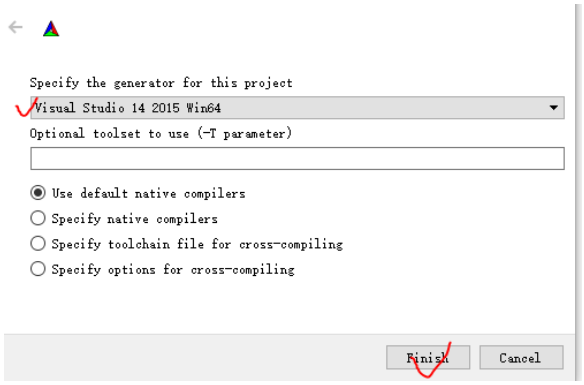


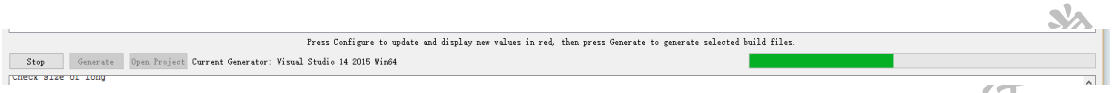
三、 用 cmake 构建 VS2015 工程

c) 点击 cmake-gui.exe 打开 cmake 界面，关闭所有跟 VS 有关的程序。
d) 在 where is the source code:,选择 dcmtk 源代码的文件夹,即 DCMTK3.6.2 包解压路径：E:/workspace/DCMTK20171009/dcmtk-3.6.2
e) 在 where to build the binaries:，选择你想存放 build 结果的文件夹（新建的空文件夹),E:/workspace/DCMTK20171009/dcmtk-msvc15,勾选 Grouped 和 Advanced



f) 点击 Configure,选择编译环境 Visual Studio 14 2015 Win64,点击 Finish,等待进度完成。





g） 进度条完成后，修改参数，本文的如下图：

MD/MDd 设置(重要)（防止产生错误三）

set "DCMTK_OVERWRITE _WIN32_COMPILER_FLAGS" to "OFF"

Xml support:

"DCMTK_WITH_XML" "ON"

"WITH_LIBXMLINC" 解压的支持库中 xml 的位置/dcmtk-3.6.2-win64-support_MD-msvc-14.0/libxml2-2.9.4

libpng support:

set "DCMTK_WITH_PNG" to "ON" and set "WITH_LIBPNGINC"

我的 E:/workspace/DCMTK20171009/dcmtk-3.6.2-win64-support_MD-msvc-14.0/libpng-1.6.30

libtiff support:

set "DCMTK_WITH_TIFF" to "ON" and set "WITH_LIBTIFFINC"

我的 E:/workspace/DCMTK20171009/dcmtk-3.6.2-win64-support_MD-msvc-14.0/libtiff-4.0.8

OpenSSL support:

set "DCMTK_WITH_OPENSSL" to "ON" and set "WITH_OPENSSLINC"

E:/workspace/DCMTK20171009/dcmtk-3.6.2-win64-support_MD-msvc-14.0/openssl-1.1.0f

zlib support:

set "DCMTK_WITH_ZLIB" to "ON" and set "WITH_ZLIBINC"

我的 E:/workspace/DCMTK20171009/dcmtk-3.6.2-win64-support_MD-msvc-14.0/zlib-1.2.11

libiconv support:

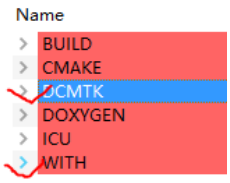
set "DCMTK_WITH_ICONV" to "ON" and set "WITH_LIBICONVINC"

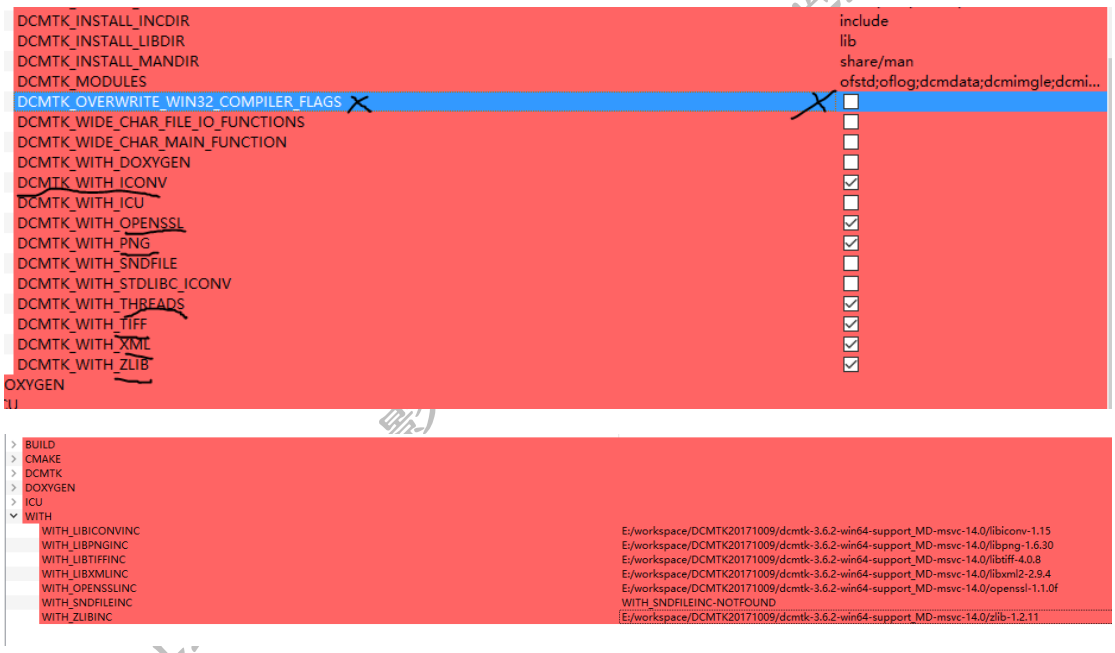
我的 E:/workspace/DCMTK20171009/dcmtk-3.6.2-win64-support_MD-msvc-14.0/libiconv-1.15

else

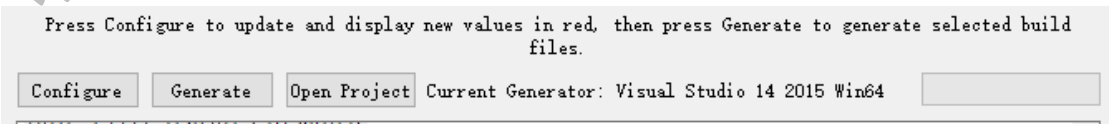
DCMTK_ENBABLE_CHARSET_CONVERSION <disable>

其他的支持库都关闭比如 ICU。





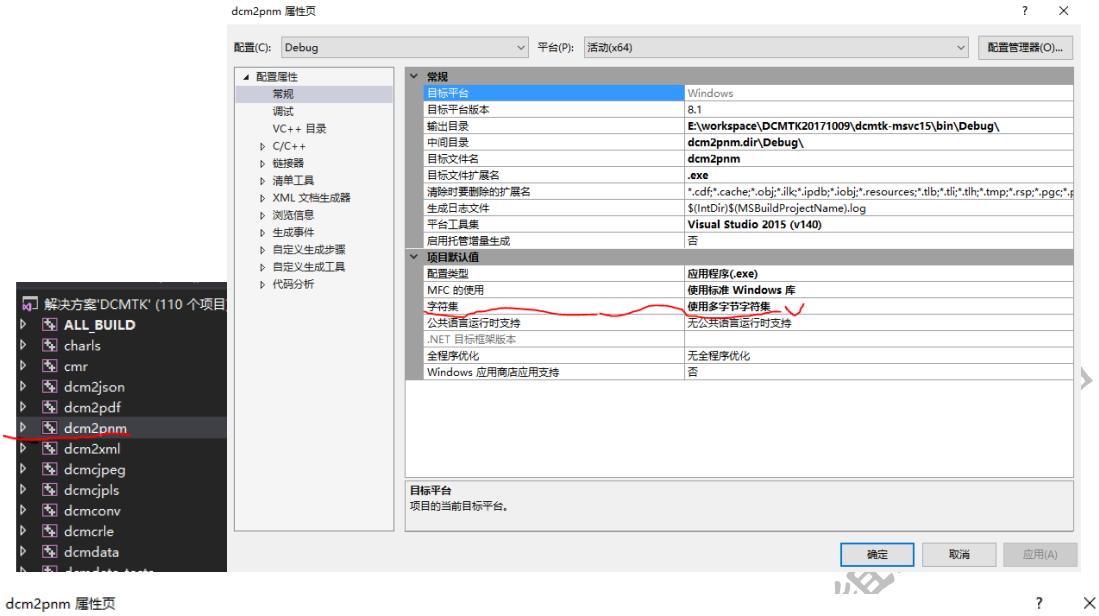
h) 修改参数完成后，点击“configure”，进度完成，点击 Generate,完成后就在 dcmtk-msvc15 文件夹下生产了 VS 工程文件，可以点击 Open Project 打开。



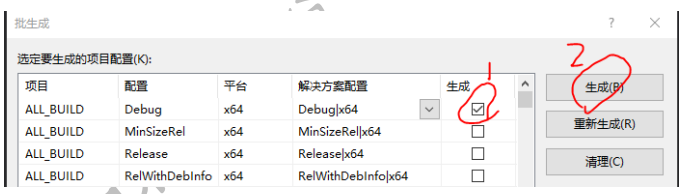
四、 编译和安装

i) 打开某个（如 dcm2pnm）项目的属性页（右击属性），查看字符集是否为使用多字节字符集，和运行库是否为 MDd(Debug 下)或 MD（release 下），如果

是证明生成项目成功。



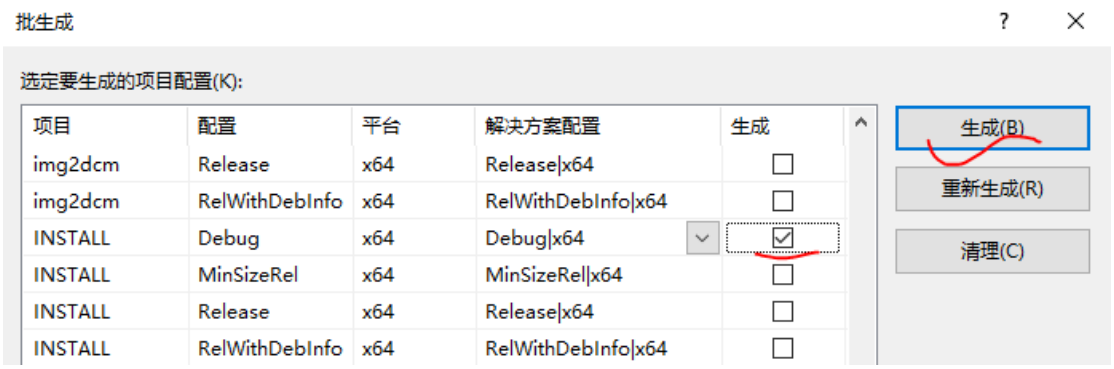
j) 编译：生成->批生成->选中您需要配置方式的 ALL_BUILD 项目，点击右边的生成，下图是生成 Debug 版的，最后生成：成功 95 个。

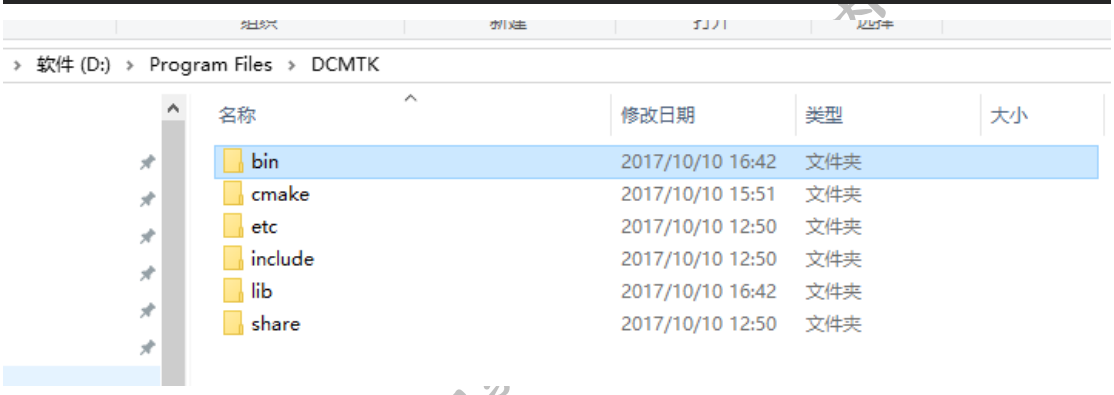



```
------ 已启动生成：项目：ALL_BUILD，配置：Debug x64 ------
  Building Custom Rule E:/workspace/DCMTK20171009/dcmtk-3.6.2/CMakeLists.txt
  CMake does not need to re-run because E:\workspace\DCMTK20171009\dcmtk-msvc15\CMakeFiles\generate.stamp is up-to-date.
========== 生成：成功 95 个，失败 0 个，最新 0 个，跳过 0 个 ==========
```

k) 为了防止产生错误一和错误二，请修改不同目录下的 cmake_install.cmake 文件，参考错误一和错误二。

l) 安装：生成->批生成->选中您需要配置方式的 INSTALL 项目，点击右边的生成，下图是安装 Debug 版的，默认安装在 C:/Program Files/DCMTK/，修改到其他盘，请参考错误一的解决方案二

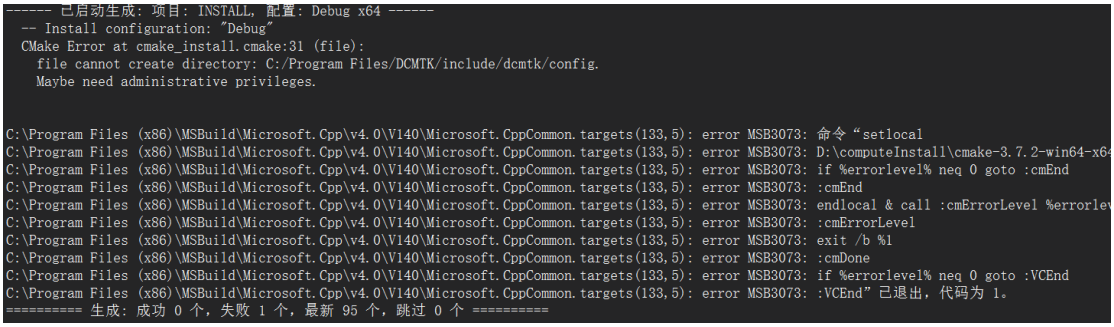




五、 错误集锦

1) 错误一：生成 INSTALL 项目时，出现以下错误

```
------ 已启动生成: 项目: INSTALL, 配置: Debug x64 ------
  -- Install configuration: "Debug"
  CMake Error at cmake_install.cmake:31 (file):
    file cannot create directory: C:/Program Files/DCMTK/include/dcmtk/config.
    Maybe need administrative privileges.
```

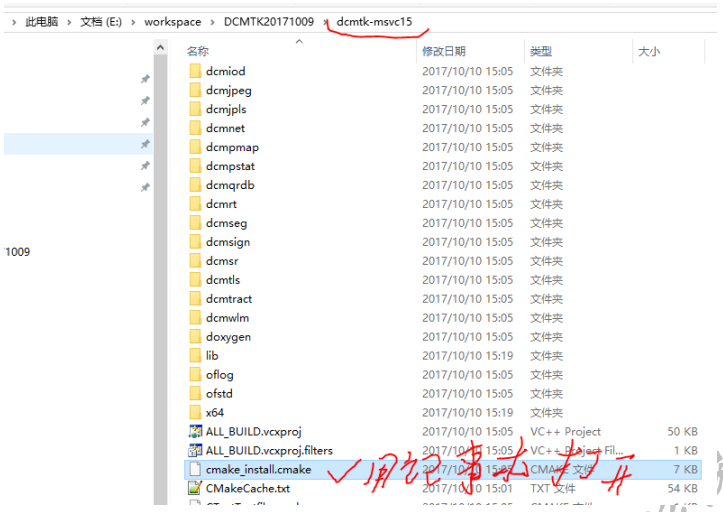


解决方案：

[1]. 以管理员方式启动 VS2015,重新生成。

[2]. 修改解决方案路径下的 cmake_install.cmake 文件（用文本方式打开，我用的是 NotePad++），把安装路径 C:/Program Files/DCMTK/修改到 D 盘如下图,清理解决方案重新生成

```
# Set the install prefix
if(NOT DEFINED CMAKE_INSTALL_PREFIX)
  set(CMAKE_INSTALL_PREFIX "D:/Program Files/DCMTK")
```



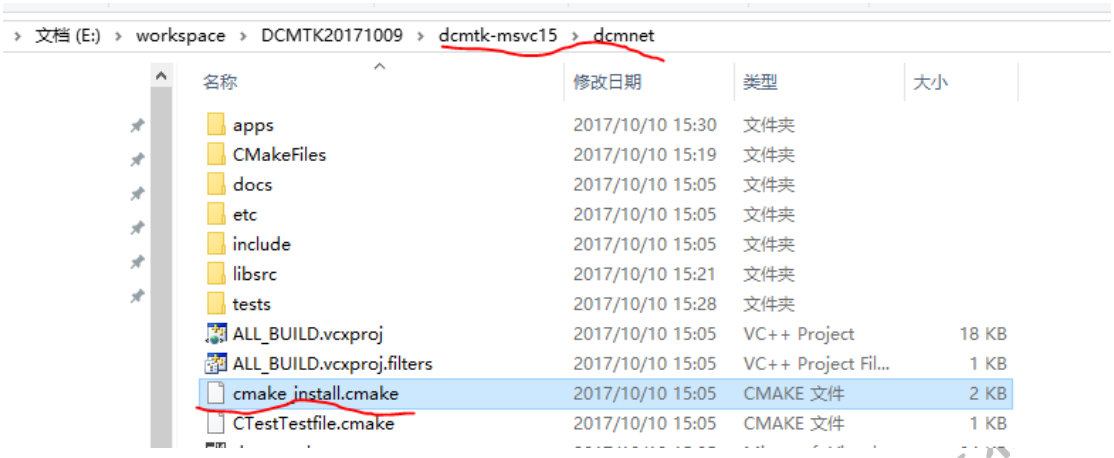
2) **错误二：生成 INSTALL 项目时，出现以下错误**

```
CMake Error at dcmnet/apps/cmake_install.cmake:139 (file):
  file INSTALL cannot find
  "E:/workspace/DCMTK20171009/dcmtk-3.6.2-win64-support_MD-msvc-14.0/openssl-1.1.0f/bin/dcmtkeay.dll".
Call Stack (most recent call first):
  dcmnet/cmake_install.cmake:33 (include)
  cmake_install.cmake:100 (include)
```

解决方案：

[1]. 把文件夹 dcmtk-3.6.2-win64-support_MD-msvc-14.0\openssl-1.1.0f\bin 下的 dll 复制到安装目录 D:/Program Files/DCMTK/bin 下。

[2]. 修改解决方案路径下的文件夹 dcmnet/apps 中的 cmake_install.cmake 文件,把 dcmtkeay.dll 修改为"dcmtkcrypto_d-1_1-x64.dll",dcmtkssl.dll 修改为" dcmtkssl_d-1_1-x64.dll"如下图




```
file(INSTALL DESTINATION "${CMAKE_INSTALL_PREFIX}/bin" TYPE FILE FILES
  "E:/workspace/DCMTK20171009/dcmtk-3.6.2-win64-support_MD-msvc-14.0/openssl-1.1.0f/bin/dcmtkeay.dll"
  "E:/workspace/DCMTK20171009/dcmtk-3.6.2-win64-support_MD-msvc-14.0/openssl-1.1.0f/bin/dcmtkssl.dll"
  )
```

```
("${CMAKE_INSTALL_COMPONENT}" STREQUAL "bin" OR NOT CMAKE_INSTALL_COMPONENT)
file(INSTALL DESTINATION "${CMAKE_INSTALL_PREFIX}/bin" TYPE FILE FILES
  "E:/workspace/DCMTK20171009/dcmtk-3.6.2-win64-support_MD-msvc-14.0/openssl-1.1.0f/bin/dcmtkcrypto_d-1_1-x64.dll"
  "E:/workspace/DCMTK20171009/dcmtk-3.6.2-win64-support_MD-msvc-14.0/openssl-1.1.0f/bin/dcmtkssl_d-1_1-x64.dll"
  )
```

3) 错误三运行时库（MT MD）冲突

导致各种错误，如 warning LNK4098：默认库“MSVCRT”与其他库的使用冲突；
使用 /NODEFAULTLIB:library
如：LNK2001 无法解析的外部符号 __imp_*

解决方案：运行时与支持库运行时匹配

六、 测试例子

1) 测试代码

```
#include "dcmtk/config/osconfig.h"
#include "dcmtk/dcmdata/dctk.h"
#include<iostream>

using namespace std;
int main()
{
	DcmFileFormat fileformat;
	OFCondition oc = fileformat.loadFile("111.dcm");
	if (oc.good()) {
		OFString patientName;
		if (fileformat.getDataset()->findAndGetOFString(DCM_PatientName, patientName).good())
```

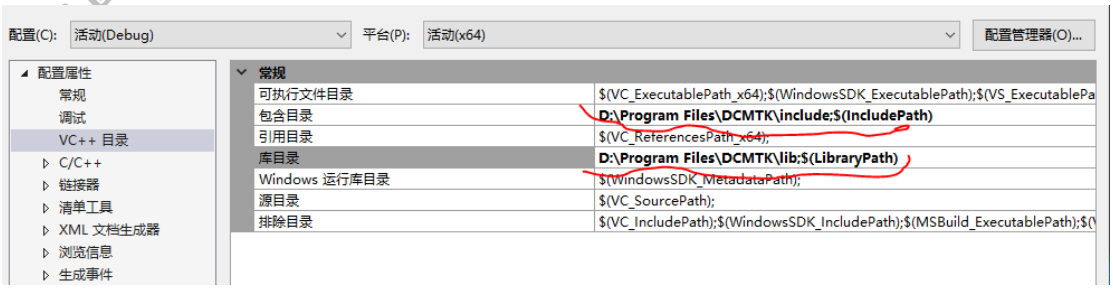
```
        {
            cout << "Patient Name:" << patientName.data()<<endl;
        }
    }

    system("pause");

    return 0;
}
```

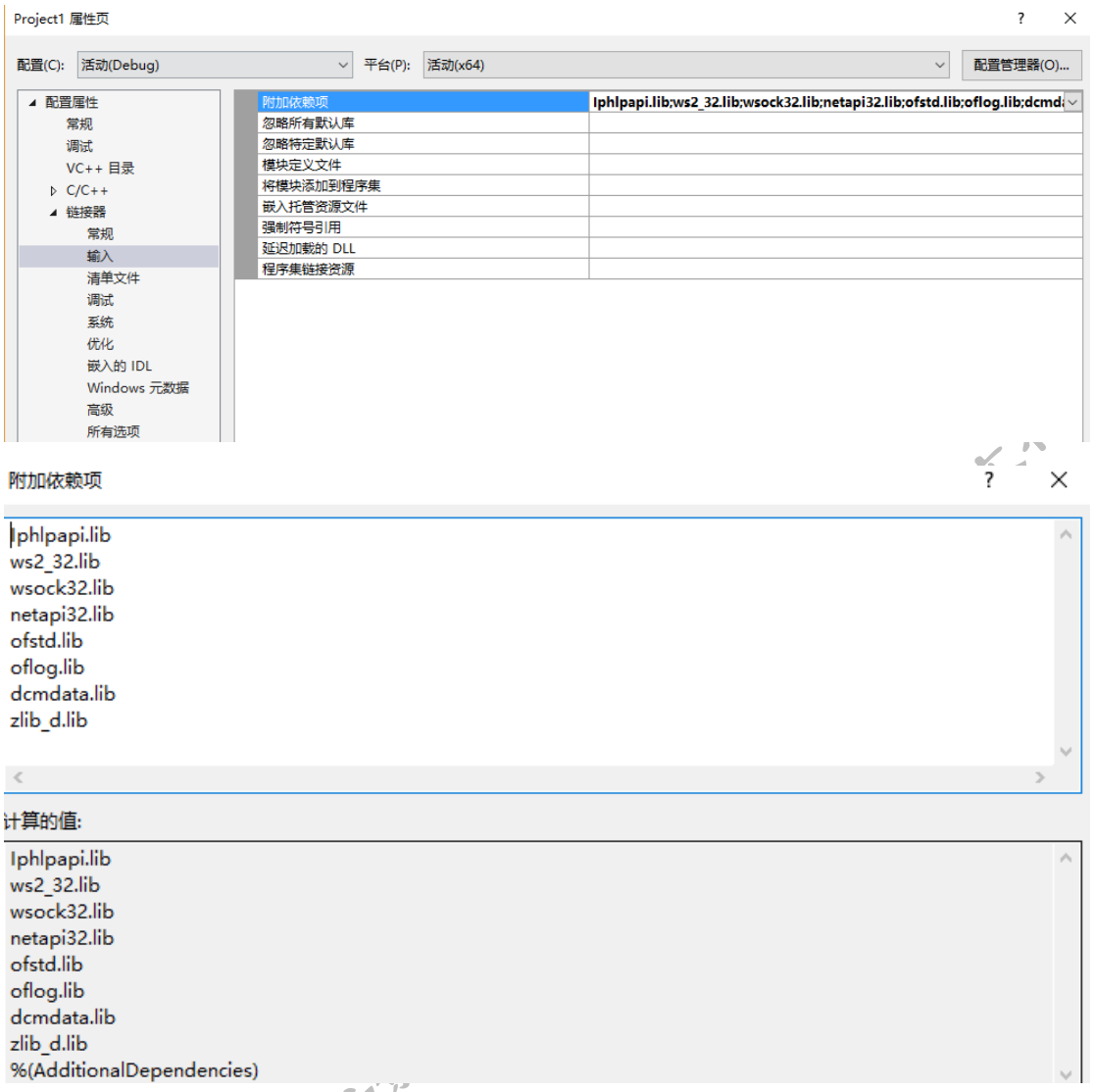
```
#include "dcmtk/config/osconfig.h"
#include "dcmtk/dcmdata/dctk.h"
#include<iostream>

using namespace std;
int main()
{
    DcmFileFormat fileformat;
    OFCondition oc = fileformat.loadFile("111.dcm");
    if (oc.good()) {
        OFString patientName;
        if (fileformat.getDataset()->findAndGetOFString(DCM_PatientName, patientName).good())
        {
            cout << "Patient Name:" << patientName.data()<<endl;
        }
    }

    system("pause");

    return 0;
}
```

结果：

```
E:\workspace\test\Project1\x64\Debug\Project1.exe
Patient Name:he ying hou
请按任意键继续. . .
```

2) **配置：**





按顺序添加依赖项 Iphlpapi.lib；ws2_32.lib；wsock32.lib；netapi32.lib；ofstd.lib；oflog.lib；dcmdata.lib；zlib_d.lib。
Iphlpapi.lib；ws2_32.lib；wsock32.lib；netapi32.lib；为系统库文件
ofstd.lib；oflog.lib；dcmdata.lib；为 DCMTK 生成的库文件
zlib_d.lib 支持包里的库文件

3) **测试数据**

测试数据用下面链接下载
http://www.casmi.science/index.php/s/mAn8XKYRd04FVIH

七、 说明

本文档仅用于交流学习，胡朝恩编写于 2017 年 10 月 10 日星期二 18：06

有任何问题发邮件到 huchaoen@fingerpass.net.cn 或 mitk@fingerpass.net.cn

欢迎访问中国科学院分子影像重点实验室 www.3dmed.net
www.radiomics.net.cn
www.mitk.net
mail.fingerpass.net.cn