DCMTK - Bug #389

Compilation on Windows 7 with MINGW64

2011-06-27 00:00 - Jörg Riesmeier

Status:	Closed	**Start date:**	
Priority:	High	**Due date:**	
Assignee:	Jan Schlamelcher	**% Done:**	100%
Category:	Library	**Estimated time:**	0:00 hour
Target version:	3.6.1+		
Module:	ofstd, dcmnet	**Compiler:**	
Operating System:			

Description

see http://forum.dcmtk.org/viewtopic.php?t=3023

History

#1 - 2012-06-20 15:55 - Jörg Riesmeier

- Category set to Library

- Target version set to 3.6.2

#2 - 2012-08-16 09:29 - Jörg Riesmeier

- Assignee set to Jan Schlamelcher

#3 - 2013-08-04 15:28 - Jan Schlamelcher

- File test_results.txt added

- File 0001-Minor-fixes-required-for-compiling-via-mingw.patch added

- Status changed from New to Feedback

- % Done changed from 0 to 50

I found two remaining problems to be solved for successfully compiling via mingw. One exists since commit 7419ad733d54c280923ee1aa0f295b0ccb1b7afb, where OFStandard::myrand_r was renamed to OFStandard::rand_r. What was the reason for this renaming? Mingw (though i believe this to be a bad pratice) offers the rand_r function as a macro instead of a real function. As the preprocessor does not expand the class scope, the macro name now collides. I can think of three possible fixes: 1. rename rand_r back to myrand_r. 2. Use #ifndef guard to check if rand_r is defined and only provide OFStandard::rand_r if not (i'm currently testing with this one. It results in using #ifndef guards not only in ofstd but also in every occurence of OFStandard::rand_r in all of DCMTK, therefore it's probably the worst alternative). 3. Make OFStandard::rand_r a free-standing function named OFrand_r (i would prefer this one).

The second problem is about using third-party libraries (e.g. zlib). It's hardcoded in the relevant CMake/3rdparty.cmake to look for xxx.lib when compiling for windows. Changing it to IF, i managed to use all third-party libraries i currently have on my system (libiconv, zlib, libpng, libtiff), using CMake's find_package() system as under unix instead of the hardcoded WIN32 stuff. I only tested a release build with static libraries so far, but there all tests succeede (see attachment). However, there are still lots of warnings when compiling with release C & CXX flags (http://nero.offis.de/projects/dicom/software/dcmtk/howto-release), i'll have a more detailed look at them in the near future.

I once created a small CMake-script-file to detect gcc-style compiler options and use for example mingw's include and libpath to search for packages working on a private project at home. I'm willing to donate a copy of this file to the DCMTK, which would allow us to for example automatically detect the libiconv included in mingw when configuring via CMake, as long as i keep the rights to do whatever i want with the original ;-). This script also allows us to disable the 'lib' prefix when creating shared objects. It's a problem about mingw authors believing shared objects should begin with 'lib' like it's the default for unix systems, while windows .dll files normally aren't named with this prefix. If you ask me, shared objects should either be named libxxx.so or xxx.dll, but not libxxx.dll like mingw authors propose. You don't want to throw "I compiled this with mingw" at the customers face who might expect "normal" windows binaries. We might add a CMake cache (checkbox) variable to allow enabling/disabling this prefix when compiling via mingw.
Last but not least, you might take a look at the attached patch to see my minimal changes in detail.

#4 - 2013-08-04 19:37 - Uli Schlachter

No idea why I renamed myrand_t. Apparently it was a good idea to have it called like this. So yeah, #ifdefs are the worst solution, everything else seems better. :-)

Also no idea if Mingw should be treated as Unix for finding packages. AFAIK it doesn't try to be quite unix-y, but if it works, I am fine with that. If Mingw wants a lib-prefix on DLLs, I have no problem with that. If I want "normal" Windows DLLs, then I would build with a Windows compiler and not with Mingw's GCC.

For oflog's tostring(): Since no callers need to be patched, it looks like these two functions are unused and could just be dropped? Alternatively, I would think that changing tostring_internal() to use OFString, too, could make that extra copy through c_str() unnecessary.

#5 - 2013-10-15 11:43 - Jörg Riesmeier

- Status changed from Feedback to Assigned

- Target version changed from 3.6.2 to 3.6.1+

Also see this forum posting on "mingw problems": http://forum.dcmtk.org/viewtopic.php?f=1&t=3838

#6 - 2014-03-04 18:24 - Jan Schlamelcher

- Status changed from Assigned to Resolved

- % Done changed from 50 to 100

I decided to fix the remaining problems (warnings) in a separate commit, since the warning fixes result in a very large patch that also affects other compilers' behaviours. Building via mingw is fixed by renaming OFStandard::rand_r to OFrand_r and changing oflog's affected string functions to work based on OFString. Some other issues have been identified that are also fixed in commit 7077471eed1911a21cddec8e00b3e866bfa743f3: An inline vs. dllimport problem (see http://stackoverflow.com/questions/11546403/importing-inline-functions-in-mingw) and a problem regarding file offsets and positions on 64 bit mingw.

The lib-prefix problem is not addressed in this commit, however I think it's not that easy as Uli states: Some companies for example use cross-compiling techniques to provide binaries for the windows platform, so they don't need to accuire windows licenses (read that in the boost mailing lists a couple of months ago, no link, sorry). I think we should allow to disable the lib-prefix if someone wants to use the DCMTK as a dependency for their projects in this way.

#7 - 2017-03-14 18:25 - Jan Schlamelcher

- Status changed from Resolved to Closed

Files

test_results.txt	18 KB	2013-08-04	Jan Schlamelcher
0001-Minor-fixes-required-for-compiling-via-mingw.patch	5.28 KB	2013-08-04	Jan Schlamelcher