GROMACS - Bug #1554
build of template.cpp is broken
07/04/2014 04:02 PM - Mark Abraham

Status: Closed
Priority: Low
Assignee: Teemu Murtola
Category: build system
Target version: 5.1
Affected version - extra info:
Affected version: 5.0

Description
Currently compiling the installed template.cpp depends on finding Boost for shared_ptr, but the machinery doesn't go looking for Boost, even the internal version (which we also install). Fix uploaded shortly.

Also, the structure around src/gromacs/utility/gmx_header_config_gen.h.cmakein leads to installation of include/gromacs/utility/gmx_header_config_gen.h with a value of GMX_CXX11 that reflects the compiler (and/or options) used to build GROMACS. For example, this breaks the template if the install used a C++11-capable compiler, and then the user tries to use a C++98 compiler. I'm unsure what solution might be available, because this header seems like it is also used during the original GROMACS build.

I'd like to get this working so that a Jenkins job can install GROMACS, and then use compilation of the template to verify that the install did at least some sane things.

Related issues:
Related to GROMACS - Task #1454: Remove gmx_header_config_gen.h Closed 03/05/2014

Associated revisions
Revision c6a675bf - 08/13/2014 11:38 PM - Teemu Murtola
Use native CMake mechanism for find_package(GROMACS)

Instead of various detection stuff that only worked well in the presence of pkg-config, use native CMake mechanisms: package configuration files and automatic export of library import definitions. Include directories and preprocessor macros influencing the installed headers are still propagated separately. The new mechanism also works for arbitrary suffixes, and is relocatable (as long as external libraries that GROMACS links against are not moved). A simple FindGROMACS.cmake is still there to hide some of the complexity to support multiple suffixes, but it is not strictly necessary if the using code wants to do those slightly more complex find_package() invocations directly.

Generalize the machinery that populated libgromacs.pc to use it also for this purpose.

TODO for later (requires changes that are better done outside this patch):
- Improve the versioning logic

The machinery and its current limitations are documented in the Doxygen page on using GROMACS as a library. Some of these could possibly be improved with additional effort, but the current approach is hopefully a reasonable compromise between usability, robustness, and complexity.

Closes #1430, #1554
Change-Id: I49c50375a5abebfe8614704c175e6ed22c9daa56

Revision 88dc658b - 09/01/2014 09:06 PM - Mark Abraham
Fix and document issues with template.cpp

Boost is a requirement for the template, so the FindGROMACS.cmake
needs machinery to deal with that. It will now find the Boost internal to GROMACS.

pkg-config generally helps (if available and GMXRC is sourced), and that use is now documented in the README.

Finding the GROMACS libraries and headers is also a requirement. This now works whether or not the user has sourced GMXRC or has pkg-config installed. The user can over-ride this with CMAKE_PREFIX_PATH if they want to.

The template needs to use the same compiler and compiler flags for post-C++98 support, and this is now documented in the README. In master branch, some of these issues are side-stepped, so take due care when merging.

Fixes #1554

Change-Id: ld30cf5149ead4a3f719499e37776a00f08309af

History

#1 - 07/04/2014 04:09 PM - Gerrit Code Review Bot
Gerrit received a related patchset '1' for Issue #1554.
Uploader: Mark Abraham (mark.j.abraham@gmail.com)
Change-Id: ld30cf5149ead4a3f719499e37776a00f08309af
c Gerrit URL: https://gerrit.gromacs.org/3756

#2 - 07/04/2014 04:26 PM - Teemu Murtola
The c++11 option affects the ABI of the library, so even if the headers would allow compilation with a c++98 compiler against such a library, either linking would fail or there would be mysterious runtime failures.

#3 - 07/04/2014 04:33 PM - Mark Abraham
Teemu Murtola wrote:

The c++11 option affects the ABI of the library, so even if the headers would allow compilation with a c++98 compiler against such a library, either linking would fail or there would be mysterious runtime failures.

... which suggests the template/CMakeLists.txt needs to acquire machinery that requires the same C++ compiler used with the GROMACS build. That might not even be possible, say installed via Linux distribution.

Perhaps we should drop the "installed template" idea and instead offer something in the source tree. I can't see managing this kind of dependency getting easier in the future.

#4 - 07/04/2014 04:47 PM - Teemu Murtola
That is certainly possible, e.g., with a CMak e package configuration file that hardcodes the most important build settings used to build Gromacs and gets installed by make install, so that it's there also for distributions. We don't need to cater for every possible use case (in particular, since it seems very low priority for most people to support this). But if we can't make even a simple case like this to work, we could just as well drop the installed headers and all other installed developer support completely. But that would be a somewhat major decision, given that this has been there for ages.

#5 - 07/04/2014 05:26 PM - Roland Schulz
I think it isn't so bad if the user has to manually make sure that c++03/c++11 is the same for the template/library. But I guess we need something automatic if we want to auto-test the template. I think the solution Teemu suggested would work well. Another option would be to use the flags specified in libgromacs.pc (we should fix that -std=c++11 is added there if needed). In theory one could also query the library whether c++11 is needed. But I'm not sure we want to do that.

BTW: I uploaded https://gerrit.gromacs.org/#/c/3750/ which is related.

#6 - 07/06/2014 05:58 AM - Gerrit Code Review Bot
Gerrit received a related patchset '1' for Issue #1554.
Uploader: Teemu Murtola (teemu.murtola@gmail.com)
Change-Id: f49c50375a5aeb6fe8614704c175e6ed22c9d9a56
c Gerrit URL: https://gerrit.gromacs.org/3762

#7 - 07/13/2014 07:12 PM - Teemu Murtola
- Related to Task #1454: Remove gmx_header_config_gen.h added
Copying some stuff here from Gerrit:

The last time I tested (and fixed the most obvious issues), I think this used to work, provided that you

1. source GMXRC before you run cmake for configuring the template build, and
2. have pkg-config, and
3. don't have any exotic stuff like MPI (it currently does not support any suffixing except for _d for double).

Also, it's not sufficient to find some boost; it really should be that version of boost that was used to compile Gromacs, in case the ABI has changed. This is similar to the general requirement of using the same C++ compiler than used to compile Gromacs to ensure ABI compatibility and usage of a single standard library instance.

The FindGROMACS.cmake only propagates include paths and other compilation options if pkg-config is present. My change linked here replaces this with a mechanism that only depends on CMake that also allows arbitrary build settings to be propagated from the original Gromacs build. But it may be too big of a chance for release-5.0. Additionally, GMX_CXX11 no longer affects the API in master (see #1454), so it does not need to be propagated.

So I propose that for release-5.0, we at most fix the propagation of the C++11 compiler flag through libgromacs.pc, but leave the mechanism otherwise the same, with whatever limitations it may have.

Copying some stuff here from Gerrit:

The last time I tested (and fixed the most obvious issues), I think this used to work, provided that you

1. source GMXRC before you run cmake for configuring the template build, and
2. have pkg-config, and
3. don't have any exotic stuff like MPI (it currently does not support any suffixing except for _d for double).

Also, it's not sufficient to find some boost; it really should be that version of boost that was used to compile Gromacs, in case the ABI has changed. This is similar to the general requirement of using the same C++ compiler than used to compile Gromacs to ensure ABI compatibility and usage of a single standard library instance.

The FindGROMACS.cmake only propagates include paths and other compilation options if pkg-config is present. My change linked here replaces this with a mechanism that only depends on CMake that also allows arbitrary build settings to be propagated from the original Gromacs build. But it may be too big of a chance for release-5.0. Additionally, GMX_CXX11 no longer affects the API in master (see #1454), so it does not need to be propagated.

So I propose that for release-5.0, we at most fix the propagation of the C++11 compiler flag through libgromacs.pc, but leave the mechanism otherwise the same, with whatever limitations it may have.

Minimal fix for release-5.0 sounds great to me. I will take a further look at my commit https://gerrit.gromacs.org/#/c/3756 tomorrow in view of comments and subsequent developments and see what can be done.

I suspect this has been fixed in several changes in master branch, but if there's things still broken, then a new issue is appropriate.