require gcc > 5

01/14/2020 12:29 PM - Eric Irrgang

Description

Required gcc version was updated to 5.1 for C++14 support. Is it too harsh to update the requirement to 6+ only a year later?

Note that supported distributions that ship with gcc < 6 will likely have gcc >= 6 packages available already. See, for instance, https://launchpad.net/~ubuntu-toolchain-r/+archive/ubuntu/test

Subtasks:

Task #3041: Remove workaround for gcc bug 58265

Associated revisions

Revision 175d3bb6 - 04/30/2020 09:31 AM - Eric Irrgang

Require C++17 standard and STL features.

Closes #3297

- Update project CMAKE_CXX_STANDARD.
- Migrate to std::optional and std::string_view.
- Update requirements in client packages.
- Provide some tool chain hints.
- Replace deprecated use of std::uncaught_exception() with std::uncaught_exceptions().
- Update install-guide.

Revision 44208ac3 - 04/30/2020 10:51 AM - Eric Irrgang

Update header sorting.

- Add missing C++ standard library headers to includesorter.py
- Work around some logic errors in the way includesorter.py enforces the stated scoping of sorted include blocks. Document.
- Apply updated includesorter.py

Refs #3297

History

#1 - 01/14/2020 04:00 PM - Mark Abraham

IIRC we were a little pushy with the bumps to requirements for GROMACS 2020. That doesn't mean we can't make minor increases, but I think we should have a clear advantage demonstrated

#2 - 01/14/2020 06:15 PM - Roland Schulz

Would it make sense to skip GCC 6 and directly require GCC 7 in a year or two? That then would allow us to switch to C++17. AFAIK, all other (CPU) compiler would allow us to switch to C++17 already now.

#3 - 01/15/2020 08:42 AM - Eric Irrgang

- Parent task deleted (#3047)

Roland Schulz wrote:

Would it make sense to skip GCC 6 and directly require GCC 7 in a year or two? That then would allow us to switch to C++17. AFAIK, all other (CPU) compiler would allow us to switch to C++17 already now.
That sounds like reasonable timing and good justification.

I will update #3047 with this proposal and remove it as a parent issue.

How do we create new "Target version" milestones? It wouldn't hurt to have 2022-infrastructure-stable to tract things that are definitely planned but definitely deferred. Otherwise, should this issue be targeted to null or to "future"?

#4 - 01/15/2020 04:15 PM - Berk Hess
- Status changed from Feedback wanted to Resolved
  
Applied in changeset 36a6581675754f80c0843922c44dba892f3a8e64.

#5 - 01/15/2020 05:58 PM - Eric Irrgang
- Status changed from Resolved to Accepted

#6 - 02/27/2020 11:33 PM - Erik Lindahl
I too agree with Roland, and perhaps we should even think of Going to GCC-7 now?

The C++11 transition in particular was difficult since many vendors (not Intel, though ;-) were way too lazy and didn't update their compilers, but after that they seem to have realised the C++ development pace is now much faster. Other big packages - e.g. Ceph - already require C++17.

Given that, I think we too can be more aggressive and start requiring C++17 now, and target C++20 from the Gromacs-2023 dev cycle in two years.

#7 - 03/04/2020 10:35 PM - Erik Lindahl
FYI: Based on discussions on Twitter, the Fujitsu compilers for A64fx/Fugaku are currently based on clang 7.1.0, which includes virtually all of C++17.

#8 - 03/04/2020 11:04 PM - Roland Schulz
I think this makes sense for C++17. C++20 is a bigger release than either 14 or 17 and arguable as big as 11. Therefore the support for it might be a bit slower than it was for 17. And we might need to watch how support develops especially for the big features like concepts, modules, and coroutines. Depending how it progresses we might need to either delay C++20 adoption or exclude some of those bigger features for another 1-2 years after adapting most of C++20.

#9 - 03/05/2020 01:16 PM - Mark Abraham
Eric Irrgang wrote:

> How do we create new "Target version" milestones? It wouldn't hurt to have 2022-infrastructure-stable to tract things that are definitely planned but definitely deferred. Otherwise, should this issue be targeted to null or to "future"?

See https://redmine.gromacs.org/projects/gromacs/settings/versions to make new versions. If it doesn't work, we can upgrade you

#10 - 03/05/2020 01:34 PM - Mark Abraham
Mark Abraham wrote:

IIRC we were a little pushy with the bumps to requirements for GROMACS 2020. That doesn't mean we can't make minor increases, but I think we should have a clear advantage demonstrated

Possible worthwhile advantages of C++17:
- structured bindings auto [a, b] = returnsATupleOrStruct()
- scoped variables initialized within if/switch if (auto var = getVar(); evalCondition(var))
- constexpr if if constexpr (a) means the branch of the predicate not taken is never compiled, so in TMP you don't have to make it compile, and for performance you're not dependent on compilers doing dead-code-elimination optimizations
- std::variant, std::optional, std::any (so we can get rid of the compat-namespace versions we have)

That seems reasonably useful, and there's a lot of other minor stuff.

It requires gcc 7, msvc 2017 and clang 5, which in practice means cuda >= 10.0 (for gcc 7, per https://gist.github.com/ax3l/9489132).

CUDA device-side c++17 isn't supported at all yet by nvcc, so no no constexpr there, sadly.

#11 - 03/05/2020 02:01 PM - Eric Irrgang

See https://redmine.gromacs.org/projects/gromacs/settings/versions to make new versions. If it doesn't work, we can upgrade you
I see "information" and "issue categories" tabs, but no "versions," and no options to edit/add versions in the "information" tab.

#12 - 03/05/2020 02:05 PM - Paul Bauer
Here is the link to make new versions: https://redmine.gromacs.org/projects/gromacs/versions/new

#13 - 03/05/2020 05:44 PM - Eric Irrgang
Paul Bauer wrote:

Here is the link to make new versions: https://redmine.gromacs.org/projects/gromacs/versions/new

I am not authorized.

#14 - 03/05/2020 07:16 PM - Roland Schulz
One question: Should we consider requiring C++17 language and library features separately? I think the pro/cons are quite different. For library features the only advantage of requiring C++17 is the reduced maintenance burden of keeping the compat:: types in our code base. For library features it isn't sufficient to use a new enough compiler (which easily can be installed in parallel to the compiler provided by the distribution) but also link against a new enough library. And linking against a new enough library in turn means that it isn't possible to link against other libraries linked against the system version of the STL. This might affect programs trying to link against the GROMACS library and other C++ libraries. I have no preference either way. Just wanted to check that we considered that.

#15 - 03/05/2020 10:13 PM - Mark Abraham
Eric Irrgang wrote:

Paul Bauer wrote:

Here is the link to make new versions: https://redmine.gromacs.org/projects/gromacs/versions/new

I am not authorized.

Buffed you

#16 - 04/30/2020 11:00 AM - Eric Irrgang
- Status changed from Accepted to Resolved

Applied in changeset 175d3bb67f4a1c0940cb9e18ff370549baa7996c.