GROMACS - Bug #3024

Bad logic for Sphinx detection CMake output

07/04/2019 07:53 PM - Eric Irrgang

Status:	Closed		
Priority:	Normal		
Assignee:			
Category:	build system		
Target version:	2020		
Affected version - extra info:		**Difficulty:**	uncategorized
Affected version:	git master		

Description

docs/CMakeLists.txt uses the presence of FIND_PACKAGE_MESSAGE_DETAILS_Sphinx to decide whether to set(Sphinx_FIND_QUIETLY ON) on repeated CMake runs, but this is not triggered in default configuration for which Sphinx was not found.

I believe the intended behavior is

1. find_package(Sphinx) to report success or failure on the first run of CMake
2. find_package(Sphinx) to be quiet on subsequent runs of the CMake configuration when Sphinx was found and cached.
3. find_package(Sphinx) to be quiet on subsequent runs of the CMake configuration either way if the user is not trying to build documentation.
4. find_package(Sphinx) to report failure on subsequent runs of the CMake configuration the user is trying to build documentation.

With the current structure of the CMake logic around the documentation builds, I'm not sure of 3 and 4 or how best to proceed. For instance, if Sphinx is not found the first time but can be found the second time, or the other way around, I would expect CMake output.

Note that after the first unsuccessful pass of find_package(Sphinx), SPHINX_EXECUTABLE is set to FILEPATH=SPHINX_EXECUTABLE-NOTFOUND, which evaluates to FALSE in a if(SPHINX_EXECUTABLE) condition. Also note that users may set -DSPHINX_EXECUTABLE on the CMake command line on the first or subsequent runs. (I, for example, have to set it because sphinx-build is the only means of execution checked and this entry point can be missing or wrong depending on Python installation or distribution. A separate issue could be to use $PYTHON_EXECUTABLE -m sphinx instead...)

Associated revisions

Revision 592b6c20 - 07/08/2019 09:33 AM - Mark Abraham

Fix hwloc and sphinx detection

These were too noisy and were not implemented efficiently (e.g. not caching the result of execute process). If an environment change is needed to detect something (like loading a module) then we expect developers to be able to unset a cache variable (or just regenerate a new cache).

Refs #3024, #3011, #2998

Change-Id: I84dfb03856b9f900fb21004bc676e4ff2647a4b4

History

#1 - 07/05/2019 03:03 PM - Mark Abraham

GMX_DEVELOPER_BUILD does interact with the tests build (the "all" target now also includes tests) and I think it triggers more docs detection also. Not sure if relevant to anybody's observations. But perhaps the docs behaviour could be improved if it's a problem here.

1, 2 and 4 are correct. With 3, the only way sphinx can be found is if the user changed the environment and the cache variable has previously been unset, so the detection now finds the executable. I think we earlier rejected a change that routinely unset the <feature>_FOUND variable that Roland thought was a decent idea. Forcing our cmake to keep trying to find sphinx every time doesn't sound great. We should prefer consistency rather than local optima. Expecting such a user in case 3 to use cmake -Uwhatever is reasonable, and consistent with the simplest cmake anyone might write.

#2 - 07/05/2019 05:19 PM - Eric Irrgang

The issue description may be too narrow. The big picture regarding case 3 could include whether to run find_package(Sphinx) at all after the first configuration.

Mark Abraham wrote:

> Expecting such a user in case 3 to use cmake -Uwhatever is reasonable, and consistent with the simplest cmake anyone might write.

This would imply always running find_package(Sphinx) but setting Sphinx_FIND_QUIETLY ON after the first run IFF the user is not trying to build documentation. How do we distinguish this use case from 4?

#3 - 07/05/2019 05:44 PM - Eric Irrgang

Eric Irrgang wrote:

> The issue description may be too narrow. The big picture regarding case 3 could include whether to run find_package(Sphinx) at all after the first configuration.
>
> Mark Abraham wrote:
>
> > Expecting such a user in case 3 to use cmake -Uwhatever is reasonable, and consistent with the simplest cmake anyone might write.
>
> This would imply always running find_package(Sphinx) but setting Sphinx_FIND_QUIETLY ON after the first run IFF the user is not trying to build documentation. How do we distinguish this use case from 4?

I'm unclear again. Your comment at https://redmine.gromacs.org/issues/2615#note-14 indicates that find_package(Sphinx) output should always be swallowed on subsequent runs unless some change clearly warrants explaining to the user that cached dependency checks have been invalidated by some non-CMake specific use case heuristic that I don't think has been clearly articulated yet.

Correct me if I'm wrong, but the intended usage of find_package is to abstract the CMake cache from the code that follows so that imported targets and variables look the same whether (a) provided by the user, (b) fetched from the cache, or (c) freshly discovered with a Find...cmake script. It's just not completely clear to me yet how the GROMACS configuration command-line experience is supposed to modify / replace / interact with the standard CMake experience.

#4 - 11/25/2019 01:21 PM - Paul Bauer

- Status changed from New to Resolved

I think this has been resolved?

#5 - 11/25/2019 02:46 PM - Eric Irrgang

- Status changed from Resolved to Closed

I think Change-Id: I84dfb03856b9f900fb21004bc676e4ff2647a4b4 resolves it to the extent of GROMACS conventions, as explained by Mark.

The Sphinx search will be quieted even if find_package is rerun with different results. The cache variable to override in order to see the new output is not particularly well-documented, but appears to be consistent with similar build system functionality.

I think the behavior is confusing to someone expecting standard CMake behavior for find_package, but that is a technical consequence of a project policy decision, so I don't think there is anything left to resolve here.