GROMACS - Bug #944

Double precision is broken on big-endian floating point systems with cmake

05/20/2012 06:17 PM - Teemu Murtola

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>Mark Abraham</td>
</tr>
<tr>
<td>Category:</td>
<td>build system</td>
</tr>
<tr>
<td>Target version:</td>
<td>4.5.6</td>
</tr>
<tr>
<td>Affected version:</td>
<td>4.5-4.5.5</td>
</tr>
<tr>
<td>Difficulty:</td>
<td>uncategorized</td>
</tr>
</tbody>
</table>

Description

src/gmxlib/maths.c has a few #ifdef IEEE754_BIG_ENDIAN_WORD_ORDER lines, but CMake defines the floating point format macros with a GMX_prefix. This means that double-precision gmx_erf() and gmx_erfc() will give nonsense results if Gromacs is compiled with CMake on a system where this define should be set. With autoconf, the macro is defined as the name that exists in the code.

If this is relevant for some architecture where Gromacs is used, then I suggest this is fixed for 4.5.6 and/or 4.6, since the fix is very simple, but set the target version however you want. If it is not relevant, I would suggest considering whether the floating-point format check is necessary at all. Note that it checks for a number of things, but these few #ifdefs are the only ones that actually use the results. Considering the comments in gmxTestFloatFormat.cmake, it's also questionable whether the code in question works with all possible combinations of integer and float endianness...

Associated revisions

Revision f361fa9c - 06/06/2012 02:40 PM - Mark Abraham

Fix endian test functionality

Code for big-endian systems was silently failing with CMake. This patch brings CMake and autotools into agreement.

Partial fix of #944
Change-Id: I1f14938fd23e926e05397977963fbf968f0406e0

Revision f361fa9c - 06/06/2012 02:40 PM - Mark Abraham

Fix endian test functionality

Code for big-endian systems was silently failing with CMake. This patch brings CMake and autotools into agreement.

Partial fix of #944
Change-Id: I1f14938fd23e926e05397977963fbf968f0406e0

Revision 5b959b65 - 06/06/2012 02:53 PM - Mark Abraham

Work around failure of endianness testing on BlueGene

Refs #944
Change-Id: l4bbfe9865800bbe4e407a88cdec6c6678d56f3b4

Revision 5b959b65 - 06/06/2012 02:53 PM - Mark Abraham

Work around failure of endianness testing on BlueGene

Refs #944
Change-Id: l4bbfe9865800bbe4e407a88cdec6c6678d56f3b4

History

#1 - 05/20/2012 06:21 PM - Erik Lindahl

We should fix this; right now the big-endian side of the world is a bit deserted (apart from Power), but with ARM64 appearing in a year or two we need...
to make sure we don't assume things are intel-only. (ARM has some horribly strange behavior with different endian-ness of bytes inside words and the multiple words in a double precision floating-point number).

#2 - 06/06/2012 02:42 PM - Mark Abraham

I attempted a fix for this without realising it was a known problem. [https://gerrit.gromacs.org/#/c/1076/](https://gerrit.gromacs.org/#/c/1076/)

BlueGene indeed requires those #defines to work correctly, so this definitely requires fixing. It affects all GROMACS builds that use CMake.

The automated testing for endianness does not work with the XLC cross-compiler for BlueGene/P. I have no idea why. I have hacked a hard fix for BlueGene, which I'll add in a separate commit.

#3 - 08/03/2012 01:11 PM - Roland Schulz

What remains to be done after Mark's (partial) fix?

#4 - 08/05/2012 07:14 AM - Teemu Murtola

- Target version set to 4.5.6
- Affected version - extra info changed from 4.5.* to 4.5-4.5.5

Roland Schulz wrote:

> What remains to be done after Mark's (partial) fix?

Mark's fix should make the cmake build work as well as the autoconf build did. Set the target version and known affected versions based on this, as the remaining problems (if any) are not common.

Still, I think it's quite confusing that:

- We explicitly test for the byte endian order of floats, but the result of the byte endian order is never used.
- The test code has comments that one should not assume the endianness to be the same as integer endian.
- The code in maths.c most likely will not work if the integer and double byte-endian order is different.

Don't know how relevant this is, but perhaps we could at least issue an error in CMake if we detect a mismatch, or clarify the comments.

#5 - 12/30/2012 07:51 AM - Teemu Murtola

- Status changed from New to Closed
- Assignee changed from Erik Lindahl to Mark Abraham

It seems that those confusing comments are not high on the priority list. Since the major issue is fixed, marking this as closed. If there ever is an architecture where the endianness differences between integers and floating-point numbers cause issues, I guess we will find out from bug reports...