### Description

```
[-Wundef]
#if HAVE_CLOCK_GETTIME && _POSIX_THREAD_CPUTIME >= 0

as this is sometimes defined in pthread.h. However, why is that test there?

Is something like the attached what is intended?

Or up the stakes to -std=c++11 and assume std::chrono implemented everywhere? (Not sure it does per-thread timing)
```

### Associated revisions

**Revision 6d850e72 - 07/01/2016 08:57 PM - Mark Abraham**

Fix use of _POSIX_THREAD_

```
This fixes a couple of aspects of behaviour. Formerly, if
_POSIX_THREADS was defined and equal to zero, we might have used
clock_gettime and got some kind of error (compiling/linking/runtime
behaviour). Similarly, if _POSIX_THREADS was undefined, C99 defines
such preprocessor symbols as zero, so we again used clock_gettime
inappropriately.

Now we avoid compiler warnings if the symbol is undefined, and when it
is defined we use clock_gettime only when POSIX_THREADS has a value
such that it is supposed to work.

Adapted this an the BG/Q fix also for
gmx_gettime_per_thread(). Expanded the documentation of why the code
is the way it is. Noted future TODO to consider std::chrono.

Fixes #1980
```

**Change-Id: Ib3e40903e2344354074c5328d40e8467f264b51f**

### History

**#1 - 06/01/2016 10:45 AM - Mark Abraham**

The test is there because a value of _POSIX_TIMERS > 0 is supposed to promise support for clock_gettime. We should indeed test whether it is
defined before we test its value, but merely being defined is not sufficient.

We could indeed replace some of this functionality with std::chrono, but we won't do that for the 2016 release. We also need to reconsider whether we
need the functionality of gmx_gettime_per_thread using CLOCK_THREAD_CPUTIME_ID, given our expectation that the affinity of threads is set if
performance is of interest, and we only care about accuracy of timing measurements when performance is of interest.

**#2 - 06/01/2016 10:51 AM - Gerrit Code Review Bot**

Gerrit received a related patchset '2' for Issue #1980
Uploader: Mark Abraham (mark.j.abraham@gmail.com)
Change-Id: lb3e40903e2344354074c5328d40e8467f264b51f
Gerrit URL: https://gerrit.gromacs.org/5919
But from the report I conclude it's still supporting clock_gettime, even though the macro is not defined. I guess we should try to use it anyhow, since the documentation says you need the macro.

PS Mark wrote _POSIX_TIMERS > 0, but the code uses >= 0.

I didn't check your HAVE_CLOCK_GETTIME test - are you saying that HAVE_CLOCK_GETTIME can be defined even when clock_gettime() isn't supported?

Now I'm even more confused.
So we have two macro when can/need to check: HAVE_CLOCK_GETTIME and _POSIX_TIMERS
And Mark says _POSIX_TIMERS > 0, where the code says >= 0.
Do we need both macro either, or xor ...?

The use of the >= was an error I introduced in 690c59fd0d0b8c4f4e207b7b61c2c84ab402566. There, I also introduced HAVE_CLOCK_GETTIME so that we also test that the C run-time library (libcrt) can link in the symbol for clock_gettime (which is problematic e.g. with static linking on Crays). We need to use clock_gettime only when it is known to work, and the updated fix at https://gerrit.gromacs.org/#/c/5919/3 does this.

The use of the >= was an error I introduced in 690c59fd0d0b8c4f4e207b7b61c2c84ab402566. There, I also introduced HAVE_CLOCK_GETTIME so that we also test that the C run-time library (libcrt) can link in the symbol for clock_gettime (which is problematic e.g. with static linking on Crays). We need to use clock_gettime only when it is known to work, and the updated fix at https://gerrit.gromacs.org/#/c/5919/3 does this.

Files

- clock.diff 944 Bytes 05/31/2016 Patrick Welche