GROMACS - Task #2857

Clarify recommended function specifies (constexpr, noexcept, pure)

02/04/2019 04:48 AM - Roland Schulz

Status:	New
Priority:	Normal
Assignee:	
Category:	
Target version:	
Difficulty:	uncategorized

Description

Our dev docs don't have any guidelines. This would mean that the core guidelines apply but we don't follow them (at least regarding noexept):

Regarding noexcept: Core guideline F.6 says that every function which can't throw should be declared noexcept. It says whether this includes function which can throw bad_alloc depends on the program. Given that we mostly run on Linux where this is useless and we don't test for it I suggest we don't try to handle bad_alloc and declare also those function noexcept.

Regarding constexpr: F.4 states that only functions which might be compile time evaluated should be constexpr, because it is part of the API. On the other hand F.8 says to prefer pure functions and use constexpr to communicate that. The guidelines fails to mention that constexpr only requires the (already weak form of purity) for at least some inputs (rather than all) of inputs. So it isn't a powerful way to declare a function pure. Outside of the gmx API (where we should clearly use constexpr according to F.4) this leaves us without clear guidance or pro/cons. Jason (from C++ weekly) advocates to use constexpr everywhere (where it is possible). Otherwise I don't see much pro/cons discussions (other than the API issue).

Regarding nodiscard: It isn't available in C++14 but it is supported by all our compilers and is easy to avoid an issue with other compilers by using gmx_nodiscard. Core guidelines don't have a rule for it. It can help avoid bugs and doesn't have a significant downsides. Potential downsides longer function declaration and require declaring gmx_nodiscard/compat_nodiscard.

Regarding pure: gcc/clang/icc support the gcc attribute for pure. But it is only useful for free functions (and static member functions) and not member functions (which most of the new code is) because its definition of pure doesn't allow changing the object through the member function (allowed e.g. for constexpr). The additional opportunities for optimizations come mainly for functions which can't be inline. And we try to make all performance critical functions inline-able. Given that those are GCC attributes we would have to hide them behind gmx_pure/gmx_const. I'm not sure how much benefit we would get and whether it is worth adding those.

Related issues:

Related to GROMACS - Feature #2248: Label all SIMD functions as pure/nodiscard	**New**

History

#1 - 02/04/2019 04:48 AM - Roland Schulz

- Related to Feature #2248: Label all SIMD functions as pure/nodiscard added