**Gromacs - Task #838**

**Improve generic error reporting routines**

11/13/2011 08:48 PM - Teemu Murtola

<table>
<thead>
<tr>
<th>Status:</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td></td>
</tr>
<tr>
<td>Category:</td>
<td>core library</td>
</tr>
<tr>
<td>Target version:</td>
<td>5.x</td>
</tr>
</tbody>
</table>

**Description**

The following issues for the code under `source/src/gromacs/fatalerror/` would be good to consider in order to make the best out of the code:

1. Output formatting for fatal errors could be improved by adding line wrapping and by stripping the root of the `CMAKE_SOURCE_DIR` directory from the reported file names.
2. Elaborate the difference between APIError exception and `GMX_RELEASE_ASSERT` (or is there any?). Which should be used in which situation? Or should `GMX_RELEASE_ASSERT` throw an APIError on failure? Or should there perhaps be a separate `GMX_API_ASSERT` for this purpose? If it is necessary to test that an assert works as intended, it is convenient to have the assert throw an exception.
3. Consider better error messages for exceptions thrown by the standard library.
4. Consider the possibility of printing stack trace for fatal errors and/or failed assertions (at least on some most common development platforms).
5. Consider the possibility of installing a signal handler for segmentation faults to produce a reasonable error message and possibly a stack trace.

All of these, except the second one, should be possible to implement without changes to the existing public API. The last one of course needs an additional API, but that shouldn't affect existing functionality.

**Subtasks:**

- Task # 985: Context information for exceptions  
  Closed
- Task # 986: Handling C++ out-of-memory errors  
  New

**Associated revisions**

**Revision 14fd0b7b - 07/04/2012 01:25 AM - Teemu Murtola**

Error/exception handling improvements.

- Change `formatErrorMessage()` to `printFatalErrorMessage()` that directly prints the error message for an exception into a FILE (typically `stderr`) instead of creating a `std::string`. This makes it possible to handle out-of-memory errors internally within the function. The function is much easier to use if we can guarantee that it never throws.
- Add basic handling of `std::bad_alloc` into the above function.
- Resolve clang warnings in the function.
- Add a macro for conveniently catching all exceptions at C++ - C boundary and reporting them as fatal errors.
- Initialize `ProgramInfo` in all C++ binaries.

Related to #838.

Change-Id: I77c355480485f563c57b6c6ea9b07a8ce7f433a8

1/3
Add more control over formatting the wrapped lines:
- Add possibility to indent the wrapped lines.
- Add possibility to indent the first line after an explicit line break
differently from other lines.
- Add possibility to mark line continuation using an explicit character.
- Add possibility to either remove or keep leading whitespace after an
explicit line break.
All except the last are already useful for the current console output.
Depending on the chosen solution, some or all of these are also needed
for #969.

Also expose the lower-level interface used internally that makes it
possible to write code that iterates over the wrapped lines and does
something else than just format them into a buffer. May be useful for
some cases in #969, but is necessary to use the wrapper for #838 (line
wrapping in error messages).

Change-Id: i905ba29856773656bf000c4b2e14d1ed2ba4de31

---

Restructure fatal error formatting.

Split printFatalError() in errorformat.h to separate functions that
print the error header, (part of) the actual message, and the footer.
This makes it possible to print the message in calling code
incrementally instead of formatting it into a string buffer.
It is easier to print the message in parts this way, and there is also
less complications in handling possible std::bad_alloc errors from
manipulating a std::string.

Also added line wrapping functionality to the message writing function,
and improved the appearance of the header.

Part of #838 and #985.

Change-Id: le17399d8c7c2c0529e0bc8492726bff827e18c4c
Revision a8626ffe - 09/07/2012 07:19 PM - Teemu Murtola

Print more information on exceptions.

- Identify a few more exception types instead of printing "Unknown exception". Print the dynamic type of the exception for the unknown exception case as well as some other cases.
- Print errno information if it is included in the exception.

Part of #838.

Change-Id: If64bd37e32e2585f78f95a29a4a2eef95b79142

---

Revision 6aeb717c - 08/09/2013 07:57 PM - Teemu Murtola

Treat exceptions better in interactive selections.

If a user input error occurs in an interactive selection parser, the message is now printed and parsing continues, like it used to work before exceptions were introduced here. Although most code still does not use exceptions for user error reporting, this makes the code more future-proof (and a subsequent commit will add some exceptions that take advantage of this).

Refactored exception message formatting to support this, removing a TODO related to duplicated code in the process.

Related to #655 and #838.

Change-Id: I4ba23c6dd1005f61d50515e6857a5dc23fc1768a

---

History

#1 - 02/15/2014 07:56 PM - Teemu Murtola
- Project changed from Source code reorganization to Gromacs
- Category set to core library

#2 - 06/20/2014 10:41 AM - Erik Lindahl
- Target version changed from 5.0 to 5.x