GROMACS - Feature #3148

Task # 2045 (New): API design and language bindings

Roadmap for gmxapi filesystem interactions.

10/15/2019 05:16 PM - Eric Irrgang

Status:	New
Priority:	Normal
Assignee:	Eric Irrgang
Category:	gmxapi
Target version:	2021-infrastructure-stable
Difficulty:	uncategorized

Description

Many tasks need to be identified and resolved to allow abstract and optimized gmxapi data flow. To restrict the scope slightly, this issue considers the migration path for tools and library facilities from deep filesystem coupling to filesystem I/O as a use case detail.

Background

gmxapi is designed to facilitate large or complex research protocols. To allow flexibility in implementing scheduling and data placement optimizations, a principal abstraction is to represent tasks as data flow, with operations as nodes and inputs/outputs as graph edges. To reduce duplication of data or effort, and to support checkpointing, the API needs clear and simple access to the data inputs and outputs of library operations, inserting some abstractions into the library to separate "data" from "files".

As these API aspects develop, we need to consider the UI cases we want to support while finding reasonable workarounds for the current CLI-centric / filesystem-based limitations of the existing tools.

For instance, in order to determine the behavior of a command with filename options, one needs some combination of knowledge of the contents of the working directory, commands previously executed in that directory (and their options), details about the status of those commands, and the GROMACS versions associated with the commands and filesystem artifacts. There is not a clear API with which to do any of this inspection. In some cases, idempotence is poorly or inconsistently defined.

Topics / Tasks

- Consolidate the current filename handling. This probably amounts to completing the migration to the gmx::Options framework.
- Pull file handling out of module details. (It is hard to evolve the filesystem interactions when many modules handle their own I/O and filesystem manipulation.)
- Provide abstractions supporting data back ends other than filesystem objects.
- Refine the machinery by which UI filename options are expressed and with which filenames are managed internally to improve discoverability, introspection, and unique mappings of data to filesystem artifacts.
- Refine the API design and implementation for conveying filesystem-backed data between operations.
- Refine the Python UI / object-oriented interface for explicitly handling data as files.

Additional discussion

Some older relevant discussions should be migrated to the GROMACS issue tracking system, but can be found on GitHub:

- https://github.com/kassonlab/gmxapi/issues/190
- https://github.com/kassonlab/gmxapi/issues/204
- https://github.com/kassonlab/gmxapi/issues/203
- https://github.com/kassonlab/gmxapi/milestone/6

Subtasks:

Feature # 3149: Python user interface for obtaining simulation artifacts as files.	**New**

History

#1 - 10/15/2019 06:23 PM - Eric Irrgang

- Description updated