# GROMACS - Task #672

## Improve command-line trajectory analysis application

01/19/2011 05:54 PM - Teemu Murtola

| | |
|---|---|
| **Status:** | Closed |
| **Priority:** | Normal |
| **Assignee:** | Teemu Murtola |
| **Category:** | core library |
| **Target version:** | 5.0 |
| **Difficulty:** | uncategorized |

**Description**

The application is currently built under source:src/programs/g_ana/, but it could perhaps be renamed to g_trajana because it will really only work for analysis of trajectories. Other issues that come to mind:

- The application could be improved to detect the name of the module from the name of the program being run. For example, when invoked as g_select, it would run the select module. This would make it possible to replace all existing analysis tools with symbolic links to this single binary, reducing the linking times significantly, as well as the installation footprint when shared libraries are not used.
- Proper help should be printed with g_ana -h, e.g., listing the available modules.

**Related issues:**

| | | |
|---|---|---|
| Related to GROMACS - Task #666: Improve help printing in the command-line runner | **Closed** | **01/14/2011** |
| Related to GROMACS - Feature #685: Wrapper binary | **Closed** | **01/26/2011** |

### Associated revisions

**Revision ab300758 - 05/07/2012 12:16 PM - Teemu Murtola**

Generic handling for multiple cmd-line modules.

Added generic methods for multiple independent modules within a single command-line tool, and modified g_ana to use this.

By itself, does not change the external behavior significantly, but makes it easier to implement parts of #666 and #672.
The wrapper binary in #685 should also be easy to implement using this approach.

Change-Id: I3058bf5db2cbb7c7a1f9e4c87dd2db8f9727fe82

**Revision 2ea25c76 - 05/08/2012 07:35 AM - Teemu Murtola**

Print list of modules with 'g_ana help'.

IssueIDs #666 and #672.

Change-Id: I001966ab3ba44c59b0186456a6467322b3b3d26d

**Revision 91f6572b - 05/09/2012 09:18 AM - Teemu Murtola**

Detect module name from symlinks.

This makes it possible to create, e.g., a symlink named g_select and point it to g_ana; executing the symlink will work exactly like 'g_ana select'.

Closes #672.

Change-Id: Ia27069466bea19b1203370c35b9ecdacfe0cdd98

### History

**#1 - 01/24/2011 01:11 PM - Mark Abraham**

Teemu Murtola wrote:

- The application could be improved to detect the name of the module from the name of the program being run. For example, when invoked

> as g_select, it would run the select module. This would make it possible to replace all existing analysis tools with symbolic links to this single binary, reducing the linking times significantly, as well as the installation footprint when shared libraries are not used.

I like the idea. However now the statically-linked Swiss-army-knife tool will have a serious memory footprint, and might be slow to load, too.

There must be some tools that don't belong in the mix, too.

#### #2 - 01/24/2011 07:12 PM - Teemu Murtola

Mark Abraham wrote:

> I like the idea. However now the statically-linked Swiss-army-knife tool will have a serious memory footprint, and might be slow to load, too.

I don't think that these will be major issues nowadays. There has to be enormous amounts of code for it to have a significant effect on modern computers on the memory footprint, and typical analysis tools now seem to have around 1000 lines on average. In comparison, the analysis framework (which will be common to all tools anyways) has something like 50k lines, and other Gromacs routines used in analysis can easily have a similar amount of code. For large systems, the bottleneck in memory use will anyways come from loading the frames and/or evaluating selections (unless the analysis itself scales badly with the system size), not from loading the code. For speed, the ability to have the whole program in disk cache (i.e., all analysis tools) might easily overweight any initial loading cost in workflows where one runs more than one analysis (and loading the trajectory is always slow if the system is not very small).

But the main argument would still be that in order to support more flexible usage of the tools (e.g., Python bindings, or running more than one tool on one pass over the trajectory), also the tool-specific code has to be in a library, so there aren't that many options.

> There must be some tools that don't belong in the mix, too.

That's true. Most tools that work on trajectory frames should benefit from the analysis framework, and could thus be run using the runner, but tools that don't work on trajectories (e.g., g_analyze or g_wham) would still be separate.

#### #3 - 04/13/2012 05:42 PM - Teemu Murtola

*- Target version set to 5.0*

#### #4 - 05/07/2012 12:38 PM - Teemu Murtola

*- Status changed from New to In Progress*

*- Assignee set to Teemu Murtola*

#### #5 - 05/10/2012 05:52 AM - Teemu Murtola

*- Status changed from In Progress to Closed*

The points mentioned, except for the rename, have been merged. Improving the help continues as part of #666. Left out the rename because with the current approach, it is just as easy to integrate all analysis tools into the same binary (or even all of Gromacs, see #685), even if not all of them use the trajectory analysis framework.

#### #6 - 02/16/2014 09:00 PM - Teemu Murtola

*- Project changed from Next-generation analysis tools to GROMACS*

*- Category set to core library*