# GROMACS - Bug #1889

## mdrun -cpi file presence dilemma

01/19/2016 10:29 AM - Berk Hess

| | | | |
|---|---|---|---|
| **Status:** | Rejected | | |
| **Priority:** | Normal | | |
| **Assignee:** | | | |
| **Category:** | mdrun | | |
| **Target version:** | | | |
| **Affected version - extra info:** | | **Difficulty:** | uncategorized |
| **Affected version:** | 5.1 | | |

**Description**

We purposely allow mdrun -cpi to continue without error when the checkpoint file is not found to simplify resubmit scripts with continuation. But this also allows setups like mdrun -cpi ../whole/state_step320.cpt to continue when the file is not found (e.g. due to a typo). In the latter case the user clearly wants a particular file and a small typo will lead to starting a run from the start which can easily be mistaken for a successful continuation.
I don't know if there is a nice solution to this issue. The only thing I can come up with is to only allow a missing checkpoint file when it comes from the current directory or only with the default or -deffnm file name.

**Related issues:**

| | |
|---|---|
| Related to GROMACS - Bug #1777: Teach mdrun about explicit -append | **Closed** |
| Related to GROMACS - Task #2169: remove 'continuation' mdp option | **New** |
| Related to GROMACS - Bug #2233: replica exchange and -append bugged? | **Accepted** |
| Related to GROMACS - Task #2344: Agree on standards for different types of ou... | **Closed** |
| Related to GROMACS - Task #2829: mdrun appends when checkpoint specified by -... | **Rejected** |

### History

#### #1 - 01/19/2016 07:25 PM - Mark Abraham

I handle the sub-case of gmx mdrun -cpi typo -g correct refusing to run in https://gerrit.gromacs.org/#/c/4439/, but need to do some follow-up work. However, there are further cases that I did not consider there, namely 1-5 below

```
1 gmx mdrun -cpi
2 gmx mdrun -cpi -deffnm correct
3 gmx mdrun -cpi -g correct
4 gmx mdrun -cpi typo
5 gmx mdrun -cpi typo -deffnm correct
6 gmx mdrun -cpi typo -g correct
```

Of those, I agree with Berk that 1 and 2 should permit a missing checkpoint file. Likewise 3. I think 4-6 should not permit a missing checkpoint file - there does not seem to be a good reason to give the checkpoint file a name and use the default name for other files, so if the checkpoint is missing then something is amiss.

Alternatively, we have -cpo for clues. If -cpi has been set, and not been filled from -deffnm (including by default) then the user has named a file. If that file is missing and its name is different from -cpo (however that value is filled) then the user can't be doing the kind of repeated running that we want to support. Cases 1-3 differ from 4-6 in that -cpo differs from -cpi. If the -cpi file is explicitly named and actually does exist, then the user seems to know what they are doing, else not.

#### #2 - 01/19/2016 10:28 PM - Berk Hess

Yes, not allowing 4-6 coincides with one of my suggestions. That's a satisfactory solution for me.

#### #3 - 01/19/2016 10:42 PM - Szilárd Páll

As long as only in clearly identifiable cases, e.g. 1-3 mentioned above, will -cpi be treated as required argument instead of an optional one, the suggestion seems good. I suggest that mdrun should also very clearly and visibly state both when it ignores and when it expects a checkpoint file.

> If the -cpi file is explicitly named and actually does exist, then the user seems to know what they are doing, else not.

Not necessarily. Unintended behavior could be continuation iso start from the beginning too e.g. if one wants to restart a run in a new directory and the run script contains leftover options including -cpi.

However, at some point the burden of checking inputs needs to be shifted to the user at some point, we can't cover all cases.

Additionally, adding some heuristic to try to detect typos may be feasible (and would not be too ambiguous if it results in ~0 false positives) by using some clever partial string matching (or even better path-aware matching). This could detect typos with a high confidence, e.g. -cpi state_step320.cp would be a clear typo if ./state_step320.cpt exists; similarly -cpi path/to/foo.cpt could be identified as a typo if either -cpi path/to/fo.cpt or -cpi path/to_better/foo.cpt exist. However, unless such code is available, implementing this seems too much hassle.

#### #4 - 05/08/2016 12:06 PM - Erik Lindahl

Hi,

Let's avoid adding a lot of complicated logic. I think we've seen this pattern quite a few times in the code:

1) Somebody wants a quick fix to avoid having to type something
2) That leads to a bug/side-effect in some other case, for which we then create a fix.
3) That fix causes a problem in a second location, so now we add even more logic.
4) That logic leads to yet another problem later

... not to mention that all these fixes keep complicating both the code and documentation.

I think it would be much better to simply require a checkpoint file to be present when the user uses the option to continue from a checkpoint. Yes, that might require the first job to be submitted slightly differently, but it will always be completely clear what the code does, and there will not be an complicated behaviour differences depending e.g. on what directory the checkpoint file is read from?

#### #5 - 05/11/2016 01:14 AM - Roland Schulz

Requiring a checkpoint but allowing it to be an empty file would simplify the logic, avoid the issue, but still allow to the convenience. One would only need to create the empty file (e.g. with touch) before starting the first job. But it would not be backward compatible.

#### #6 - 05/11/2016 09:58 AM - Szilárd Páll

Erik Lindahl wrote:

> I think it would be much better to simply require a checkpoint file to be present when the user uses the option to continue from a checkpoint. Yes, that might require the first job to be submitted slightly differently, but it will always be completely clear what the code does, and there will not be an complicated behaviour differences depending e.g. on what directory the checkpoint file is read from?

Sounds like a reasonable option, but it will break people's workflow which has been a high priority to avoid so far. Hence, I'd strongly consider whether a step in that direction is worth the (apparent) usability regression.

#### #7 - 05/23/2016 01:23 AM - Erik Lindahl

I only see two options:

1) Never allow runs with the -cpi argument if the file is missing.
2) Always allow it, but issue a warning.

I am strongly against the idea of adding more logic to "sometimes" allow it by trying to guess what the user wants to do. It just takes one look at the history in redmine to see how many historical bugs we have caused with that approach.

#### #8 - 05/23/2016 06:08 PM - Mark Abraham

*- Related to Bug #1777: Teach mdrun about explicit -append added*

#### #9 - 05/23/2016 06:16 PM - Mark Abraham

Erik Lindahl wrote:

> I only see two options:
>
> 1) Never allow runs with the -cpi argument if the file is missing.
>
> 2) Always allow it, but issue a warning.
>
> I am strongly against the idea of adding more logic to "sometimes" allow it by trying to guess what the user wants to do. It just takes one look at the history in redmine to see how many historical bugs we have caused with that approach.

I'm happy breaking workflows if it lets us produce simpler code that we can show works.

I'm happy with Erik's approach 1, but we need to be clear what we intend gmx mdrun and gmx mdrun -deffnm name to do, given that appending is the default and discussion at #1777 exists. Those implictly name an input checkpoint file. In combination I suggest we want only the following combinations to start a simulation, under the standing assumption that all relevant files and names are valid and unchanged from those in the checkpoint file (as applicable):

```
gmx mdrun
gmx mdrun -cpi
gmx mdrun -append
gmx mdrun -cpi -append
```

and all the above with -deffnm something, with or without specialized names for the various files. (I could also live with removing the ability to specialize the names of the -x, -g, -e, -c, -cpi, -cpo, -s etc. files, but there's no strong need for this now.)

Erik's approach 2 lets users append to a trajectory even if their intended checkpoint file is missing, which seems very dangerous.

#### #10 - 05/23/2016 06:23 PM - Roland Schulz

We could not break existing workflows (in the default case) and still simplify things (Erik's suggestion 1) if we would make grompp write out an empty checkpoint file. Of course if the users workflow contains copying only the tpr file to a different location (host/folder) he would still need to make a change (also copy the empty cpt). This would require a bit of extra code (create/read empty checkpoint) but not extra logic which is easier to break and harder to test.

#### #11 - 05/23/2016 09:50 PM - Mark Abraham

Roland Schulz wrote:

> We could not break existing workflows (in the default case) and still simplify things (Erik's suggestion 1) if we would make grompp write out an empty checkpoint file. Of course if the users workflow contains copying only the tpr file to a different location (host/folder) he would still need to make a change (also copy the empty cpt). This would require a bit of extra code (create/read empty checkpoint) but not extra logic which is easier to break and harder to test.

I don't like the current way you can have either .tpr, or .tpr+.cpt to start a simulation. Roland's suggestion is one way to break it, but means people have things to manage. Why don't we write just write a full .tpr as the checkpoint? (Later, with a switch to JSON or something, it becomes ~trivial.)

#### #12 - 05/23/2016 09:59 PM - Roland Schulz

Mark Abraham wrote:

> I don't like the current way you can have either .tpr, or .tpr+.cpt to start a simulation. Roland's suggestion is one way to break it, but means people have things to manage. Why don't we write just write a full .tpr as the checkpoint? (Later, with a switch to JSON or something, it becomes ~trivial.)

I think it is nice if inputs don't get modified. And I prefer that over the convenience of having only 1 file. If tpr was a container which can be easily analyzed/split/changed by cmd-line tools (e.g. tgz) one could have both advantages.

#### #13 - 05/23/2016 10:46 PM - Mark Abraham

Yeah, preserving the immutability of an initial input is a good point. Similarly, we probably don't want to be in the game of gmx mdrun -s topol_part0002.tpr or gmx mdrun -s topol_next.tpr.

A containerized .tpr that accumulated checkpoints is an interesting idea - it solves the "missing .cpt" problem in a direct way.

#### #14 - 05/24/2016 11:07 PM - Erik Lindahl

I think the discussion might boil down to not changing anything at all.

In short: If the user tells GROMACS to do the wrong thing by having typos, we might do the wrong thing just as instructed - too bad, but that's far better than trying to be smart.

In particular if we are thinking of changing formats a year from now I would prefer not do introduce changes that will only be temporary, and I don't really fancy the idea of forcing all users who don't care about checkpointing to start worrying about those files just so we can help checkpointing users who can't type correctly.

#### #15 - 05/25/2016 11:14 PM - Erik Lindahl

*- Status changed from New to Rejected*

Based on the discussion in https://gerrit.gromacs.org/#/c/5890/ and #1777, I'm rejecting this for now.

As we've said both there and above, there simply isn't any foolproof way to automatically decide what to do, even when the users explicitly tells us to do something else - and on the downside it can cause some other pretty bad bugs.

The one solution that would resolve it is to be strict about the options and force the user not to add -cpi when starting the first simulation, so we can require files to be present. If we don't want that, we'll have to live with the risk of not catching a typo.

#### #16 - 05/05/2017 04:11 PM - Mark Abraham

*- Related to Task #2169: remove 'continuation' mdp option added*

#### #17 - 08/21/2017 02:55 PM - Mark Abraham

Erik Lindahl wrote:

> Based on the discussion in https://gerrit.gromacs.org/#/c/5890/ and #1777, I'm rejecting this for now.
>
> As we've said both there and above, there simply isn't any foolproof way to automatically decide what to do, even when the users explicitly tells us to do something else - and on the downside it can cause some other pretty bad bugs.
>
> The one solution that would resolve it is to be strict about the options and force the user not to add -cpi when starting the first simulation, so we can require files to be present. If we don't want that, we'll have to live with the risk of not catching a typo.

Note that permitting -cpi to name a file that is not found is potentially problematic with multi-simulation runs, where a file might be missing for one crashed simulation, leading to who-knows-what result. A problem was reported at #2233

#### #18 - 08/21/2017 02:55 PM - Mark Abraham

*- Related to Bug #2233: replica exchange and -append bugged? added*

#### #19 - 03/01/2018 10:30 PM - Mark Abraham

User ran into problems reported at https://redmine.gromacs.org/issues/2344#note-13

#### #20 - 03/01/2018 10:30 PM - Mark Abraham

*- Related to Task #2344: Agree on standards for different types of output and log files added*

#### #21 - 02/15/2019 01:35 PM - Mark Abraham

*- Related to Task #2829: mdrun appends when checkpoint specified by -cpi is not found added*