Gromacs - Task #1177

Moving manual repo into source repo

03/06/2013 02:50 PM - Mark Abraham

Status: Closed
Priority: Normal
Assignee: Mark Abraham
Category: documentation
Target version: 5.0

Description

As mentioned in https://gerrit.gromacs.org/#/c/2119/, life is much better if we have the manual in the same repo as the source.

1. Reviewers can see when there is (not) matching code and manual updates (so it's easier to block a patch until the documentation is acceptable)
2. The manual can find out about the executables built by the parent repo, rather than searching for them in an installation
3. The manual can find the in-source .mdp options documentation easily
4. Less duplication of version variable names
5. Easier to build matching code and manual (e.g. for automating the release process)

There are various ways to do it
A. git submodule (adds work every time the manual is updated)
B. grab the manual files and dump them into the repo as a squashed commit (loses history)
C. pretend the manual was in the repo all along https://www.kernel.org/pub/software/scm/git/docs/howto/using-merge-subtree.html

I think C is best, and so far it was almost as easy as B. One downside of C is that tools like gitk see the history as if it was a long independent branch that got merged in, and so seeing both code and manual histories is awkward. git log just shows commits in date order, so that works fine.

Currently, manual release-4-5-patches is merged into release-4-6, and https://gerrit.gromacs.org/#/c/2213/ will merge release-4-6 into master. In due course, merging the code release-4-5-patches into release-4-6 and then that into master will see the manual content move up. During the code-repo merge, any missing history from the manual repo's branches will have to be added then. Fiddly, but possible.

You could argue that this operation is only worth doing for master, but I want to streamline the process of making releases (point 5 above). There will be at least one more 4.5.x release, and monthly 4.6 releases. Also, if all the active repos now have the manual, nobody can be confused about what's going on.

I have already done C (but it is no big deal if we want A or B instead). Currently working on making things work after that.

Associated revisions

Revision 6876e9d9 - 10/16/2013 08:39 PM - Mark Abraham

Move manual repo into code repo, and make it build

Supersedes I12066943a1. Contains all the manual content from Ic1f74befee88. Manual history is lost - nobody was super keen on keeping it, and the git repos still exist if we care later on.

If the user opts in with cmake ...-DGMX_BUILD_MANUAL=on, this patch detects whether a manual build is possible. If so, creates a make target called "manual" which will build the PDF manual in the manual sub-directory. The manual is never automatically made, and the supporting detection is never automatically run. Thus, a normal GROMACS user or developer never has any problems.

Appendix D (compendium of man pages) is removed. The header of chapter 7 always contained advice on where to get that content. That advice is updated.

Also

- streamlined manual CMakeLists.txt for new context
- removed outdated 'gmxmaxmajor[]' LaTeX macro

11/25/2015
• noted some TODOs for future work

Refs #1177

Change-Id: i7e56a01182fbd980322c8d47cb8a9c88d7b932a

History

#1 - 03/06/2013 02:54 PM - Mark Abraham
Some matching changes to Jenkins config will be needed. Currently the automated manual build takes 30s, so I'd expect we can make the fastest build slave do the test of the manual build for us.

#2 - 03/06/2013 05:14 PM - Roland Schulz
It is possible to configure Jenkins to build a job only if changes to certain files have occurred. So we could keep building the manual only if manual files are changed. But given it is so fast it might not be worth it.

If we anyhow use submodules for regression tests it might be fine to use it also here (people don't have to learn it just for one). But given that the manual isn’t so big either A or C is fine.

#3 - 03/06/2013 06:37 PM - Mark Abraham
Roland Schulz wrote:

It is possible to configure Jenkins to build a job only if changes to certain files have occurred. So we could keep building the manual only if manual files are changed. But given it is so fast it might not be worth it.

Cool! That might also allow us to avoid Jenkins triggering on other kinds of patches, too.

If we anyhow use submodules for regression tests it might be fine to use it also here (people don't have to learn it just for one). But given that the manual isn’t so big either A or C is fine.

Yeah, but a submodule seems right for the regression tests because of the size of the (history of the) binary files. The manual does have some binary image files, but at least they don’t change much.

A hidden catch with C is that pushing to refs/for/release-4.5-patches requires

- uploading all those manual patches,
- having Gerrit permit me to forge identity (normally disallowed for obvious reasons), and then
- gerrit wanting to offer them all for review.

Apparently the solution for the third is to push straight to refs/heads to bypass review (https://groups.google.com/forum/?fromgroups=#!topic/repo-discuss/nOow3SBH90w).

However, we don’t want to do all that if people will hate the in-repo manual build, so I will upload a temporary version of B, so that people can review the form of what comes next, before we commit to one of A, B and C (or other alternatives).
#4 - 03/06/2013 06:54 PM - Mark Abraham


#5 - 03/06/2013 07:08 PM - Mark Abraham

We could now package the manual source in our source tarballs, or not. No strong feelings here.

#6 - 03/06/2013 07:19 PM - Roland Schulz

Mark Abraham wrote:

- gerrit wanting to offer them all for review.
  Apparently the solution for the third is to push straight to refs/heads to bypass review (https://groups.google.com/forum/?fromgroups=#!topic/repo-discuss/nOow3SBH90w).

If commits are already reviewed in a different branch than they don't show up again but only the merge commit. So one could first push them into a branch of the gromacs repo (can be done using git command line tool on the gerrit server to bypass all gerrit permissions). Then only the merge commit would show up which could be reviewed. Only the view for such a topic merge is not very useful because it doesn't show the change relative to before but instead the (useless) conflict (fixed by: https://gerrit-review.googlesource.com/#/c/33960/). Otherwise you also need to temporary disable the requirement for change-ids.

#7 - 03/06/2013 07:40 PM - Teemu Murtola

- For configuring Jenkins to run the manual build only for a subset of changes may not be worth the trouble, as identifying all files that affect the manual can be tricky (e.g., there is quite a lot of source code files that can actually affect the manual). As it isn't that slow, and does not need to be built on multiple platforms, I think it is best to simply build it every time.
- I don't see much value in using submodules for the manual, as the manual isn't really that big. They add some hassle, and remove some benefits that a single repository has: e.g., with submodules, it is still possible to merge manual changes into the manual main branch before the code changes are in. Right now, the master branch of the manual has merged documentation for IMD (has had that for something like a year or more), while that code is still waiting for code review...
- The only disadvantage with a single repository I see is that it will be a harder to do independent releases of the manual (i.e., releasing the manual later than the corresponding source release). But I think that will be outweighed by the benefits of better integration and possibility to more easily automate creating the manual corresponding to a source release.
- I don't have any strong opinion on whether to keep the history or not.
- I don't have any strong opinion on whether to include the manual in the source tarballs or not (don't see any big benefit in including it, so it may be better to leave it out to reduce the size).

PS. Is Michael really the appropriate assignee?

#8 - 03/06/2013 08:04 PM - Mark Abraham
Teemu Murtola wrote:

- For configuring Jenkins to run the manual build only for a subset of changes may not be worth the trouble, as identifying all files that affect
  the manual can be tricky (e.g., there is quite a lot of source code files that can actually affect the manual). As it isn't that slow, and does not
  need to be built on multiple platforms, I think it is best to simply build it every time.

The Windows Jenkins slaves are very slow already, so they are definitely not going to be building the manual (even if they had a suitable set of
software). So some cunning is already required.

- I don't see much value in using submodules for the manual, as the manual isn't really that big. They add some hassle, and remove some
  benefits that a single repository has: e.g., with submodules, it is still possible to merge manual changes into the manual main branch before
  the code changes are in. Right now, the master branch of the manual has merged documentation for IMD (has had that for something like a
  year or more), while that code is still waiting for code review...
- The only disadvantage with a single repository I see is that it will be a harder to do independent releases of the manual (i.e., releasing the
  manual later than the corresponding source release). But I think that will be outweighed by the benefits of better integration and possibility
  to more easily automate creating the manual corresponding to a source release.
- I don't have any strong opinion on whether to keep the history or not.
- I don't have any strong opinion on whether to include the manual in the source tar balls or not (don't see any big benefit in including it, so it
  may be better to leave it out to reduce the size).

Thanks for feedback.

PS. Is Michael really the appropriate assignee?

Er no, but he's next to me in the list... Sorry Michael!

#9 - 03/06/2013 08:10 PM - Mark Abraham

Roland Schulz wrote:
Mark Abraham wrote:

- gerrit wanting to offer them all for review.
  Apparently the solution for the third is to push straight to refs/heads to bypass review (https://groups.google.com/forum/?fromgroups=#topic/repo-discuss/nOow3SBH90w).

If commits are already reviewed in a different branch than they don't show up again but only the merge commit. So one could first push them into a branch of the gromacs repo (can be done using git command line tool on the gerrit server to bypass all gerrit permissions). Then only the merge commit would show up which could be reviewed. Only the view for such a topic merge is not very useful because it doesn't show the change relative to before but instead the (useless) conflict (fixed by: https://gerrit-review.googlesource.com/#/c/33960/). Otherwise you also need to temporary disable the requirement for change-ids.

That's a good trick. So if we do C, we can make a temporary branch, push to it brutally so that gerrit thinks those ChangeIds in the gromacs project have been reviewed. (I gather having been reviewed in the manual project is no good. Not quite sure how the old pre-gerrit patches with no ChangeID will cope. Can rubber-stamp them with ChangeIds if we need to) Then do C. Since the merge commit that does the import has no content, there's nothing to review (even if the diff is ugly/useless). The point being reviewed is "do we want the manual in the repo with all the history."

#10 - 03/06/2013 08:20 PM - Mark Abraham
On https://gerrit.gromacs.org/#/c/2215/, Teemu said:

Which approach do we want to take with such non-essential targets in CMake? Currently, master has a few, e.g., for building the doxygen documentation, that are created unconditionally (as long as the required tools are available) as custom targets that are not included in ALL. Here, we instead have a CMake option controlling whether that target is created at all, and if that CMake option is set, the target is always built.

I don't have a strong opinion whether a CMake option or a make target controls whether the manual is built - but it should not build by default (and does not in the current proposed patch).

Either way, some more work is probably in order. At the moment you have to cd to build-cmake/manual and use make from there to actually get the PDF. It would be easy to create "make manual" to do that for you.

Having targets not included in ALL allows one much more easily to build those parts only when necessary without any hassle with turning CMake options on and off all the time. They can be a bit more difficult to implement (the UseLATEX.cmake file may not support that out of the box), but shouldn't be too hard. I don't think that anyone wants to continuously be building the manual when they develop code, but it can be useful to be able to easily build it from time to time.

One issue with using a make target is that the (e.g.) pdflatex detection now always has to take place at CMake time. (Unless we do some kind of hackery like we do for the internal FFTW build.) Same goes for Doxygen in master. We have to take care that this kind of thing does not confuse a user who just wants to build GROMACS and move on.

By non-essential here I mean that the users don't normally need to know of these targets, and they are only useful for developers in some contexts. So it doesn't really matter that much if it doesn't work on every imaginable platform combination, as long as it does not mess with the
Yes. Requiring the use of an explicit top-level make target to build the manual is a good thing either way. A CMake option allows us to insulate non-manual builders from worrying about the detection of manual build stuff. Using CMake option is somewhat consistent with how we handle automatic regression testing and building things that require optional external libraries (GSL, X11). Also unit testing in master (LibXML2)?

#11 - 03/06/2013 08:39 PM - Teemu Murtola
Mark Abraham wrote:

One issue with using a make target is that the (e.g.) pdflatex detection now always has to take place at CMake time. (Unless we do some kind of hackery like we do for the internal FFTW build.) Same goes for Doxygen in master. We have to take care that this kind of thing does not confuse a user who just wants to build GROMACS and move on.

I agree that some detection code needs to run, but as long as it gracefully simply disables those targets if everything is not found, I don't see any problem that could be caused for normal users. Also, even if the detection is broken and finds something that doesn't work, it shouldn't matter as long as invoking that target requires explicit user action, so that normal users will not see those failures as part of normal builds. If we want to isolate things further, we could have a "developer" option in CMake that disables all these detection parts and custom targets if not set.

Yes. Requiring the use of an explicit top-level make target to build the manual is a good thing either way. A CMake option allows us to insulate non-manual builders from worrying about the detection of manual build stuff. Using CMake option is somewhat consistent with how we handle automatic regression testing and building things that require optional external libraries (GSL, X11). Also unit testing in master (LibXML2)?

IMO, the main reason why those are controlled by a CMake variable is that they affect the build or other operations that normal users may want to do, and there needs to be a way to disable them to avoid breaking the normal build. The main reason why I added the option to explicitly disable unit testing in master is that Google Mock may not classify as "simple" C++, so to avoid potential issues with some compilers, it is now easy to disable it in case it breaks a simple "make". I don't really see why GMX_XML would be necessary in the current code, though (I just left it there, since it was there before).

#12 - 03/06/2013 08:46 PM - David van der Spoel
Just a side note: libxml2 will come back. So please leave the cmake switches in there as well.
David van der Spoel wrote:

Just a side note: libxml2 will come back. So please leave the cmake switches in there as well.

Continuing on this side note: I was more implying that in the current code, there is no point in giving the user an option to set GMX_XML=OFF, i.e., building without libxml2 if it is found. And given that http://www.gromacs.org/Developer_Zone/Programming_Guide/Compilation has also said "Coding changes that will be effectuated directly following release of 4.1: ... libxml2 will be obligatory" for something like at least four years, I would simply remove GMX_XML and always assume that the user wants to use it if it is available. Then it is simple to make it a fatal error if libxml2 is not available when it is actually required for something else than just unit tests.

Roland Schulz wrote:

For handling external data (e.g. for tests) the cmake module ExternalData exists: https://github.com/Kitware/ITK/blob/master/CMake/ExternalData.cmake. It will ship with cmake 2.8.11 but can also be used with earlier versions. Not sure we want to use it, just adding it here as an option.

Mark Abraham wrote:

https://gerrit.gromacs.org/#/c/2214/ is updated to do option C above. It looks exactly like the old version, except that the manual history is now present in the code repo. Browsing that history is not actually workable, though, and it can make the output of tools like gitk unwieldy, so I'm also happy for a solution like B (i.e. like patch 1 of https://gerrit.gromacs.org/#/c/2214/). (The two differ only in their parents, and one bugfix patch in the meantime.)

Mark Abraham wrote:

- Status changed from New to Accepted
- Target version changed from 4.5.7 to future
- Affected version set to N/A

Mark Abraham wrote:

- Status changed from Accepted to In Progress

Mark Abraham wrote:

- Target version changed from future to 5.0

Going to re-target work for this to master branch, and give up on preserving history. The will probably documentation reorganization at some point and the history will get to messy to be useful. Old repo history is still available when we want it.

Teemu Murtola wrote:

What's the current plan here? If we only put the manual in master without history, what's the plan of merging potential fixes from manual/release-4-6 branch? Or should we simply freeze the 4.6 manual after this is done and target all further fixes to 5.0 manual? In any case, another set of merges is required in the manual repo to bring all changes to master. Having this done would simplify work on the wrapper binary and on #959.

Mark Abraham wrote:

#13 - 03/06/2013 09:18 PM - Teemu Murtola

#14 - 03/18/2013 07:57 PM - Roland Schulz

#15 - 04/16/2013 05:08 PM - Mark Abraham

#16 - 04/29/2013 06:43 PM - Mark Abraham

#17 - 04/29/2013 06:43 PM - Mark Abraham

#18 - 05/22/2013 05:35 AM - Mark Abraham

#19 - 07/07/2013 03:23 PM - Teemu Murtola

#20 - 07/07/2013 05:18 PM - Mark Abraham
All I can envisage happening is

- a copy-paste dump of the updated 4.6 manual repo into the master branch source repo,
- plus CMake configuration to make it build (roughly as in the dormant gerrit patch above),
- plus tweak to Jenkins so it builds the manual from the binaries in the source tree, rather than the nightly binaries.

I'm away much of next week, but it'll be easy for me to do the first two tomorrow if it makes life easier for you.

We can effectively freeze the 4.6 manual now, as far as I know.

---

**#21 - 07/07/2013 06:07 PM - Teemu Murtola**

No hurries, I'm also away for several weeks with limited computer access, so I won't be able to do much more than give general comments. But I think there's currently unique content in each of the 4.5, 4.6 and master manual branches, so there should be a final merge between the branches before dumping the content in the source repo.

---

**#22 - 10/01/2013 06:31 AM - Teemu Murtola**

When doing the merge, the main issues to consider are:

- how to merge changes from release-4-6 before and after the merge (including those now pending in gerrit, if there are any), or should we prohibit them?, and
- what to do with the current IMD manual content that's there in master, but for which the code changes are still in gerrit? At latest before the release, the content of the manual should be synchronized with what is actually included.

---

**#23 - 10/11/2013 05:30 PM - Mark Abraham**

Teemu Murtola wrote:

When doing the merge, the main issues to consider are:

- how to merge changes from release-4-6 before and after the merge (including those now pending in gerrit, if there are any), or should we prohibit them?, and

I have committed the last release-4-6 manual change, and have merged that up to master ([https://gerrit.gromacs.org/#/c/2698/](https://gerrit.gromacs.org/#/c/2698)).

- what to do with the current IMD manual content that's there in master, but for which the code changes are still in gerrit? At latest before the release, the content of the manual should be synchronized with what is actually included.

Yes, I have pinged the IMD code patch ([https://gerrit.gromacs.org/#/c/1080/](https://gerrit.gromacs.org/#/c/1080)) and we will remove that manual chunk later as required.
#24 - 12/31/2013 06:45 PM - Teemu Murtola
- Status changed from In Progress to Resolved

#25 - 01/01/2014 06:51 AM - Teemu Murtola
- Tracker changed from Bug to Task
- Subject changed from moving manual repo into source repo to Moving manual repo into source repo
- Status changed from Resolved to Closed

Was done in the linked commit. Additional work, if required, can happen outside this task.