



Updates on CMake, Modularization, and C++17

WARWICK
THE UNIVERSITY OF WARWICK

Ben Morgan, Gabriele Cosmo, Gunter Folger, Jonathan Madsen

CMake: Minor Updates

- Mostly bug fixes:
 - *Addition of missing `GEANT4_USE_SMARTSTACK` option*
 - *Removed `-DG4FPE_DEBUG` from main `DEBUG` build mode due to incompatibilities with Qt*
 - *New “`DEBUG_FPE`” mode for debug with FPE testing*
 - *Support use of legacy OpenGL over GLVND*
 - *Fixes in `Geant4Config` for compiler flag, path, and configuration errors reported via Bugzilla*

MR !108: Replacement of Core -D flags

- Ensures required global level definitions are consistent when building and using Geant4
- Eases configuration and portability
- No impact on Geant4 code, User code must include G4Types.hh to pick up definitions

The screenshot shows a GitLab Merge Request (MR !108) titled "cmake-V10-05-02, global-V10-05-05: Use generated header in place of -D flags". The MR is merged and was opened 7 months ago by Benjamin Morgan. The description explains that Geant4 uses -D flags for preprocessor definitions, and this MR promotes them to fixed #defines in a new header. A list of promoted flags is provided: G4USE_STD11, G4MULTITHREADED, G4_STORE_TRAJECTORY, G4VERBOSE, and GEANT4_USE_TIMEMORY. The MR also mentions a workaround for G4gl2ps and lists categories and fixes. The MR was edited 4 months ago by Gabriele Cosmo. The bottom section shows a "Request to merge" button, a successful pipeline status, and a "Merged by Geant4 GitLab Bot" notification.

gitlab.cern.ch

Projects Groups More Search or jump to... 15 3 51

G4 geant4 > G4 geant4-dev > Merge Requests > !108

Merged Opened 7 months ago by Benjamin Morgan Edit

cmake-V10-05-02, global-V10-05-05: Use generated header in place of -D flags

Geant4 uses -D flags to set preprocessor definitions that affect how the toolkit is built. In most cases, they affect both the API and ABI of the libraries and thus the exact same set as used to build the toolkit must be used by users when compiling their applications. This makes supporting users via CMake, geant4-config, or otherwise, fragile as all flags must be known, exported, and correctly applied.

As flags are determined by options set at Geant4 build time, promote them to fixed #defines in a new header generated by CMake using the chosen build options to #define/#undef as appropriate. The set of definitions promoted are confined to those affecting all categories:

- G4USE_STD11
- G4MULTITHREADED
- G4_STORE_TRAJECTORY
- G4VERBOSE
- GEANT4_USE_TIMEMORY

Ensure the header is included by primary Global category headers so that Global and client categories will pick up the changes transparently.

A workaround is needed in G4gl2ps as this does not link to G4global.

Categories: cmake, global, visexternalsgl2ps, externals/clhep, DICOM2

Fixes: JIRA Dev-250

Edited 4 months ago by Gabriele Cosmo

Request to merge bmorgan: cmake-V10-05-02 into master

Pipeline #827194 passed for 458f628e on bmorgan: cmake-V10-05-02

No approval required

View eligible approvers

Merged by Geant4 GitLab Bot 4 months ago Revert Cherry-pick

MR !264: Replacement of Geometry -D Flags

- Same purpose as for Global -D flags
- No changes required in Geant4 or User code
- Some residual -D flags are still propagated from VecGeom
- Ticket raised on VecGeom JIRA to address this: <https://sft.its.cern.ch/jira/projects/VECGEOM/issues/VECGEOM-535>

The screenshot shows a GitLab Merge Request (MR !264) titled "cmake-V10-05-03, geommmng-V10-05-05, geom-csg-V10-05-00, geom-specific-V10-05-07". The MR is merged and was opened 4 months ago by Benjamin Morgan. The description explains that the replacement of G4 with VecGeom solids is done via setting of a number of -D symbols when compiling toolkit/client code. It also mentions that the selection of VecGeom solid replacement fixes the API/ABI, and that the -D symbols are promoted to #define/undef statements in a C++ header G4GeomConfig.hh. The MR includes categories: cmake, geommmng, geom-csg, geom-specific. It was edited 4 months ago by Gabriele Cosmo. The MR is merged by Geant4 GitLab Bot 4 months ago, with the changes merged into master with commit 05e4bad2. The source branch has been deleted. The MR has 0 thumbs up and 0 thumbs down.

MR !138: Support use of clang-format

- Allows automatic code formatting when clang-format is available via, e.g.
 - make G4global-format
- Totally optional, but could be automated
- Style matches current Geant4 formatting

gitlab.cern.ch

Projects ▾ Groups ▾ More ▾ Search or jump to... 🔍 15 3 51

G4 geant4 > G4 geant4-dev > Merge Requests > !138

Merged Opened 6 months ago by Jonathan Madsen Edit Report abuse

Added tooling for clang-format

- {Category}-format targets are provided
- clang-format-helper.py is provided to do maintenance operations
- geant-clang-format is the Geant4 accepted format
- modified some source.cmake to exclude formatting certain files in format targets
- updated pre-commit to fix issues with on macOS
- updated pre-commit to check if formatting has been applied

Edited 6 months ago by Jonathan Madsen

Request to merge jmadson:clang-format-tooling into master

Pipeline #748400 passed for 8072ca18 on jmadson:clang-format-tooling

No approval required

View eligible approvers

Merged by Geant4 GitLab Bot 2 months ago Revert Cherry-pick

The changes were merged into master with e69a28e3

The source branch has been deleted

0 0

Show all activity 1/4 threads resolved

Discussion 30 Commits 3 Pipelines 3 Changes 11

Geant4 GitLab Bot @geant4bot · 6 months ago Maintainer

Warnings:

- the merge request is marked as a work-in-progress.

MR !138:Excluding Code

- May be parts of your code you don't want formatted
 - e.g numeric tables, certain chains of statements
- You can disable formatting of any block by bracketing it between `// clang-format on/off` comments:

```
#include <optional>

// This will be formatted
std::optional<int> GetAnswer(int r) {
    if ( some_condition_on(r) )
        return 42;

    return {};
}

// This will be left untouched

// clang-format off
auto result = GetAnswer(someInteger);

if (result)
    PrintResult(*result);
else
    Report("no result obtained");
// clang-format on
```




Avoiding Parkinson's Law of Triviality

Style is a classic bikeshed. Use for what it's good for: consistency, maintainability

WIP: CMake Minimum Version Update

- 10.6 will require 3.12 for building, at least 3.8 if building applications with CMake
 - *Fully support Windows DLL builds without workarounds*
 - *Fully support C++17 builds*
 - *Improved support for finding, using, and exporting third party packages (e.g. zlib, Qt)*
 - *Better support for newer platforms*

WIP: MR !318 on OpenInventor

- Several fixes identified during Vis mini workshop
- Main complexity is tracking interdependent options and build/use time.
- See later item on use of third party packages in general.

The screenshot shows a GitLab Merge Request (MR) page. At the top, the browser address bar shows 'gitlab.cern.ch'. The page header includes navigation links for 'Projects', 'Groups', and 'More', along with a search bar and notification icons. The main content area displays the MR title 'WIP: Updates for build of OpenInventor driver' and its status 'Open', opened 3 months ago by Benjamin Morgan. Below the title, a description explains the updates needed for the OpenInventor driver. The MR is categorized into 'General fixes' and 'SoQt support', each with a list of bullet points. A 'Request to merge' section shows the source branch 'bmorgan:oiqt-driver' and the target branch 'master', with a note that the source is 586 commits behind. Below this, there are buttons for 'Open in Web IDE', 'Check out branch', and a dropdown menu. A 'Merge requests are a place to propose changes...' section includes an illustration of a computer screen with code and a chat bubble. At the bottom, there are reaction buttons (thumbs up, thumbs down, smiley) and a 'Show all activity' dropdown. The 'Discussion' tab is active, showing 1 discussion, 0 commits, and 0 changes.

Projects ▾ Groups ▾ More ▾ Search or jump to... 15 3 51

geant4 > geant4-dev > Merge Requests > !318

Open Opened 3 months ago by Benjamin Morgan Edit Close merge request

WIP: Updates for build of OpenInventor driver

As discussed at the Geant4 Vis Workshop, some updates are needed to the configuration of the OpenInventor driver to support the SoQt component, and static builds on Unix.

General fixes:

- SoQt/Qt libraries need to be listed before Coin to ensure correct static linking order
- Coin requires `pthread` and `d1` linked (esp. for static)
- Coin requires `libXi` for SoQt

SoQt support:

- Unix only
 - Warn when using SoQt on macOS (should work, but install of SoQt can be troublesome)
- Can choose only ONE of SoQt or SoXt
- When choosing SoQt, `GEANT4_USE_QT`` should also be enabled

Request to merge `bmorgan:oiqt-driver` into `master`
The source branch is 586 commits behind the target branch

Open in Web IDE Check out branch

Merge requests are a place to propose changes you have made to a project and discuss those changes with others. Interested parties can even contribute by pushing commits if they want to. Currently there are no changes in this merge request's source branch. Please push new commits or use a different branch.

Create file

👍 0 👎 0 😊

Show all activity ▾

Discussion 1 Commits 0 Changes

WIP: MR !484 on Usage Requirements

- Backporting three items of functionality from new CMake system (MR !474)
- Remove requirement for `include_directories` in your `sources.cmake`
- Full deployment of [usage requirements](#)
- Improved link safety with target "namespaces"

The screenshot shows a GitLab Merge Request (MR !484) titled "WIP: cmake-V10-05-13: Migrate to use of CMake target usage requirements". The MR is in a "WIP" (Work in Progress) state, indicated by a yellow icon and the text "This is a Work in Progress". It was opened 1 week ago by Benjamin Morgan. The description explains that while MR !474 provides a long-term fix for CMake build issues, several items were discovered that can be implemented now without the full system. The changes include:

- Use `target_include_directories` on each Geant4 library to set the `INTERFACE_INCLUDE_DIRECTORIES` usage requirement to the list of directories containing headers for that library.
 - This means that developers no longer have to call `include_directories` in `sources.cmake` for any directory under `sources`
 - Likely still needed for externals (hence the WIP)
- Use `install(... INCLUDES_DESTINATION ...)` to set `INTERFACE_INCLUDE_DIRECTORIES` for the installed versions of the libraries.
 - CMake-based clients should then no longer need to use `include_directories(${Geant4_INCLUDE_DIRS})`, though as above, may be needed for externals as currently implemented.
- Exported Targets now use the `NAMESPACE` feature so that targets are named for example `Geant4::G4global`. This increases link safety by guaranteeing that `libG4global.<ext>` is present and will not appear as `-lG4global` in the link line (possibly resulting in another version being picked up).

These changes are fully backward compatible. Any errors in testing likely highlight an issue with external libraries.

The MR is 49 commits behind the target branch (master). A pipeline #1092510 passed for commit 61c15aa8. No approval is required. The MR is marked as a Work in Progress, and there is a button to "Resolve WIP status".

WIP: MR !474 on New Library Build Model

- Implementation of [system presented at Ferrara Meeting](#)

- *Delayed by Git work*

- Developers declare:

- *Code in this (sub)category (include/src)*

- *(Sub)categories used by this one*

- "Integrator" declares

- *Composition of (sub)categories into libraries*

- Single way to create libraries, but can be modified easily without impacting developers

Projects ▾ Groups ▾ More ▾ Search or jump to... 15 3 51

geant4 > geant4-dev > Merge Requests > !474

Open Opened 1 week ago by Benjamin Morgan 2 of 4 tasks completed Edit Close merge request

WIP: cmake-V10-05-14: Implement new system for building Geant4 libraries

Since Geant4 migrated to use of CMake for building its component libraries, only the "global" library layout has been supported over the "granular". Here "granular" libraries are the lowest level components in Geant4, and correspond to a collection of source code organized as:

```
+ granular_library/  
+ CMakeLists.txt  
+ sources.cmake  
+ include/  
| +- ... public headers ...  
+ src/  
+ ... implementation details ...
```

The code under the `include` and `src` directories are compiled into a single "granular library" which may link to others. "Global" libraries are then composed (by source) from 1-N granular libraries. As discussed (see e.g. [Presentation at Ferrara collaboration meeting](#)), neither system provides a good balance between modularity and coherence. In addition, the current CMake "API" for developers to define their code to the build is awkward, requiring knowledge of transitive dependencies.

The Merge Request addresses these issues through use of a new CMake API for defining how Geant4 code gets composed and compiled into libraries. A full technical outline of the proposed changes implemented here is [described here](#). The key features are:

- "granular libraries" are replaced by "geant4_modules": a set of source code and other "geant4_modules" used by that code
 - A CMake API is implemented to declare "geant4_modules" and their build/usage requirements much like CMake does for executables/libraries
- "global libraries" are replaced by "geant4_libraries": a set of "geant4_modules"
 - Actual libraries ('so/dylib/dll/lib') are compiled/linked from the from the "sum" of all code and requirements of the component modules.

This solves the transitive dependency issue via CMake's usage requirement system so that developers of a "geant4_module" only have to declare the "geant4_modules" directly used by their module. The system maps between "geant4_modules" and "geant4_libraries" to track usage, so developers never need to know which library their module eventually ends up in. This mapping also ensures that a module cannot be composed into multiple libraries, and that there are no "dangling" modules.

Due to the major nature of this change, the use of the new CMake system is only enabled when the CMake option `GEANT4_USE_NEW_CMAKE` is set to `ON`. It is `OFF` by default, and this *should* result in zero changes to the build/test/library structure. To minimise changes to existing CMake scripts at the category/module level (`sources.cmake` and similar), the new API provides the existing interfaces such as `geant4_define_module` but implemented in terms of the functions such as `geant4_add_module`. However, some changes are needed in

Changes you will need to make after MR !474

- To begin with, **NONE**
 - *System is backward compatible with existing code in your sources .cmake scripts*
- Initial library structure identical to “Global” to allow comparative testing
- sources .cmake migration over several weeks and Merge Requests (Git helps immensely here!)
 - *System will also be fully documented through Sphinx*
- **Input and Feedback on MR !474 are welcome!**

Build Topics after 10.6

- These are to motivate discussion amongst you this week and for next year's workplan
- *Use, support, and deprecation of Third Party Libraries in Geant4*
- *Choice of minimum C++ Standard and Platforms to support in 2020 release*

Third Party Library use in Geant4

- Primarily used in Analysis (HDF5, FontConfig) and UI/Vis (Qt, OpenGL, Inventor ...), Geometry (Xerces/VecGeom) categories
 - *This is a **good** thing - we shouldn't reinvent the wheel, but...*
- Try and avoid exposing Third Party headers in Geant4 headers. This makes building/deploying applications harder.
- **We should not be afraid of removing components of Geant4 that depend on EOL, obsolete, or otherwise unsupported packages**

Migrate to C++17 in 2020?

- **Question:** Should we make the 2020 release require the C++17 Standard as the minimum?
 - See [Presentation at Lund](#) on language features added by C++17
- **Benefit** would be use of these new features to help improve and simplify Geant4 for users
- **Cost** would be dropping support for older platforms/toolchains
 - https://en.cppreference.com/w/cpp/compiler_support
 - **Personal opinion:** I think we go for it through a Major Release, core/common set of platforms are fine directly or with easily available tooling

Discussion

- Work on modularisation system underway
 - *Comment/Review/Feedback under MR !474 are very welcome*
- **Two build related questions for you to discuss now and over the coming days:**
 - *Better management and deprecation timeline(s) for third party libraries used in Geant4*
 - *Should we move to requiring C++17 for the 2020 release?*