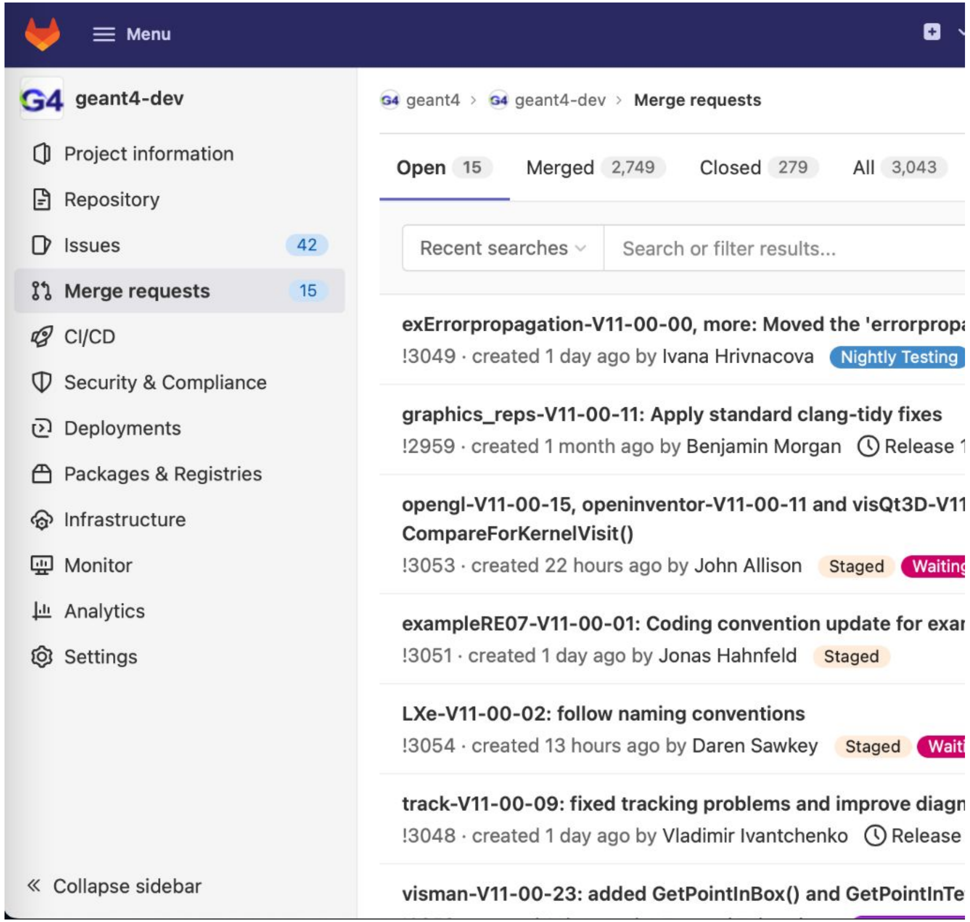
Geant4 Develop, Build and Test Tools Update

Ben Morgan

GitLab Status

- **Passed 3000 MRs, ~600 since 11.0!**
- **CERN plans to move from GitLab Enterprise to Community over next year due to license costs**
- Monitoring this, but expect all functionality used by Geant4 will be retained or locally implemented by CERN



exErrorpropagation-V11-00-00, more: Moved the 'errorpropa
!3049 · created 1 day ago by Ivana Hrivnacova Nightly Testing

graphics_reps-V11-00-11: Apply standard clang-tidy fixes
!2959 · created 1 month ago by Benjamin Morgan Release 1

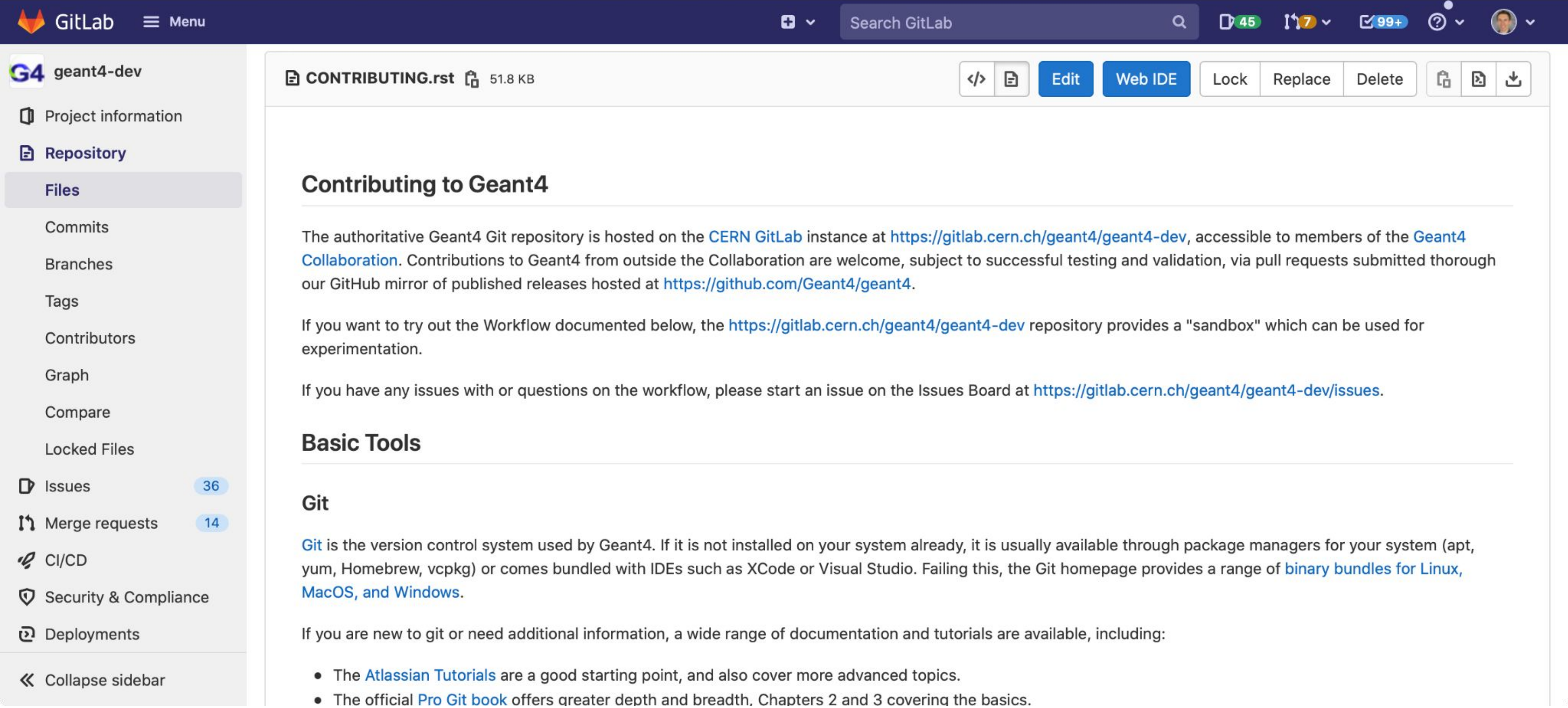
opengl-V11-00-15, openinventor-V11-00-11 and visQt3D-V11
CompareForKernelVisit()
!3053 · created 22 hours ago by John Allison Staged Waiting

exampleRE07-V11-00-01: Coding convention update for exar
!3051 · created 1 day ago by Jonas Hahnfeld Staged

LXe-V11-00-02: follow naming conventions
!3054 · created 13 hours ago by Daren Sawkey Staged Waiti

track-V11-00-09: fixed tracking problems and improve diagn
!3048 · created 1 day ago by Vladimir Ivantchenko Release

visman-V11-00-23: added GetPointInBox() and GetPointInTe



Contributing to Geant4

The authoritative Geant4 Git repository is hosted on the CERN GitLab instance at https://gitlab.cern.ch/geant4/geant4-dev, accessible to members of the Geant4 Collaboration. Contributions to Geant4 from outside the Collaboration are welcome, subject to successful testing and validation, via pull requests submitted thorough our GitHub mirror of published releases hosted at https://github.com/Geant4/geant4.

If you want to try out the Workflow documented below, the https://gitlab.cern.ch/geant4/geant4-dev repository provides a "sandbox" which can be used for experimentation.

If you have any issues with or questions on the workflow, please start an issue on the Issues Board at https://gitlab.cern.ch/geant4/geant4-dev/issues.

Basic Tools

Git

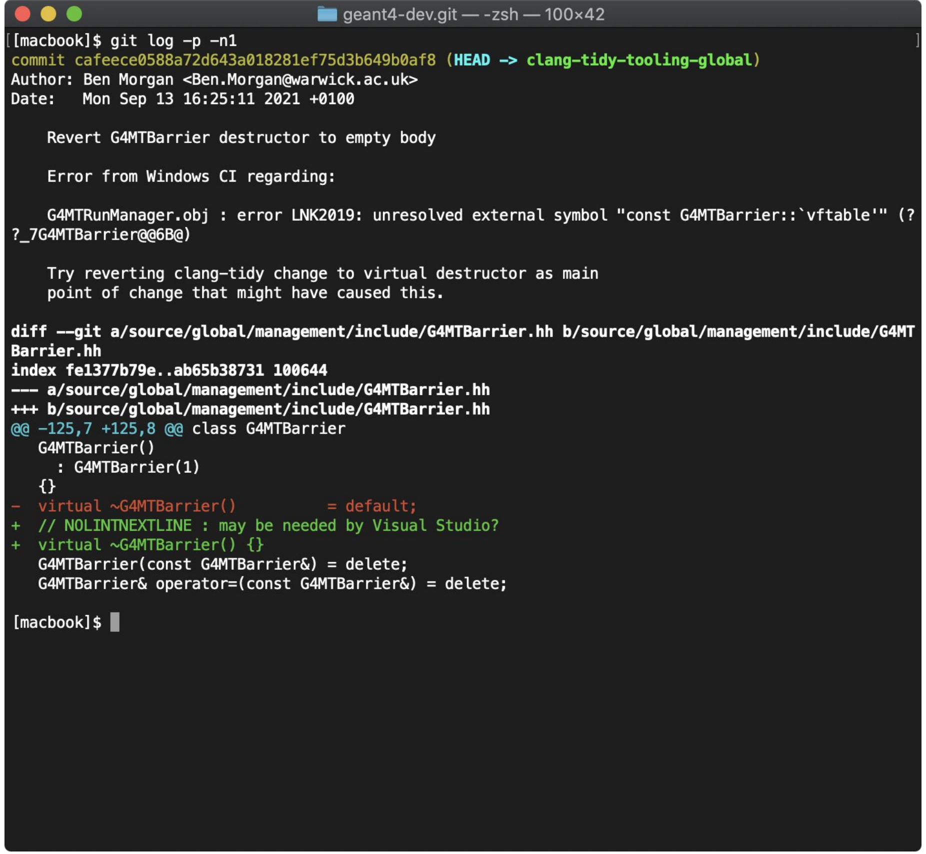
Git is the version control system used by Geant4. If it is not installed on your system already, it is usually available through package managers for your system (apt, yum, Homebrew, vcpkg) or comes bundled with IDEs such as XCode or Visual Studio. Failing this, the Git homepage provides a range of binary bundles for Linux, MacOS, and Windows.

If you are new to git or need additional information, a wide range of documentation and tutorials are available, including:

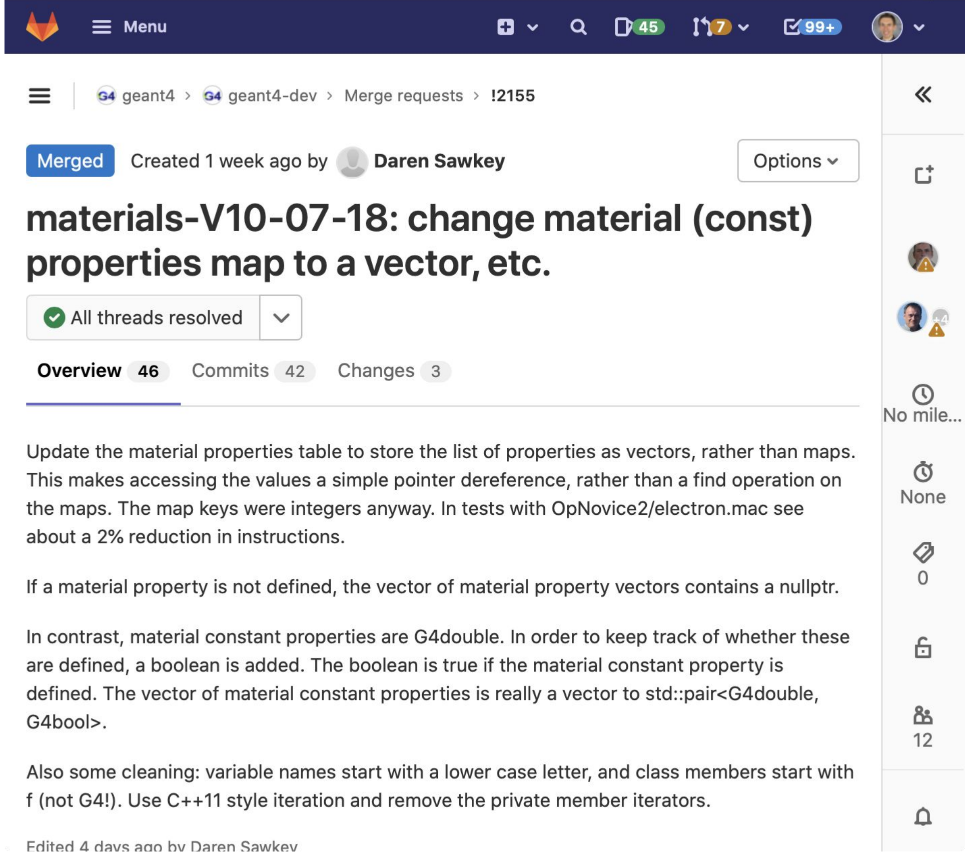
- The Atlassian Tutorials are a good starting point, and also cover more advanced topics.
- The official Pro Git book offers greater depth and breadth. Chapters 2 and 3 covering the basics.

Merge Request Process: Documentation

A living document, so Issues/MRs to improve it are very welcome



```
[macbook]$ git log -p -n1
commit cafeece0588a72d643a018281ef75d3b649b0af8 (HEAD -> clang-tidy-tooling-global)
Author: Ben Morgan <Ben.Morgan@warwick.ac.uk>
Date:   Mon Sep 13 16:25:11 2021 +0100

    Revert G4MTBarrier destructor to empty body

    Error from Windows CI regarding:

    G4MTRunManager.obj : error LNK2019: unresolved external symbol "const G4MTBarrier::`vftable'" (?
?_7G4MTBarrier@@6B@)

    Try reverting clang-tidy change to virtual destructor as main
    point of change that might have caused this.

diff --git a/source/global/management/include/G4MTBarrier.hh b/source/global/management/include/G4MT
Barrier.hh
index fe1377b79e..ab65b38731 100644
--- a/source/global/management/include/G4MTBarrier.hh
+++ b/source/global/management/include/G4MTBarrier.hh
@@ -125,7 +125,8 @@ class G4MTBarrier
   G4MTBarrier()
     : G4MTBarrier(1)
   {}
-  virtual ~G4MTBarrier()          = default;
+  // NOLINTNEXTLINE : may be needed by Visual Studio?
+  virtual ~G4MTBarrier() {}
   G4MTBarrier(const G4MTBarrier&) = delete;
   G4MTBarrier& operator=(const G4MTBarrier&) = delete;

[macbook]$
```



Created 1 week ago by Daren Sawkey

materials-V10-07-18: change material (const) properties map to a vector, etc.

All threads resolved

Overview 46 Commits 42 Changes 3

Update the material properties table to store the list of properties as vectors, rather than maps. This makes accessing the values a simple pointer dereference, rather than a find operation on the maps. The map keys were integers anyway. In tests with OpNovice2/electron.mac see about a 2% reduction in instructions.

If a material property is not defined, the vector of material property vectors contains a nullptr.

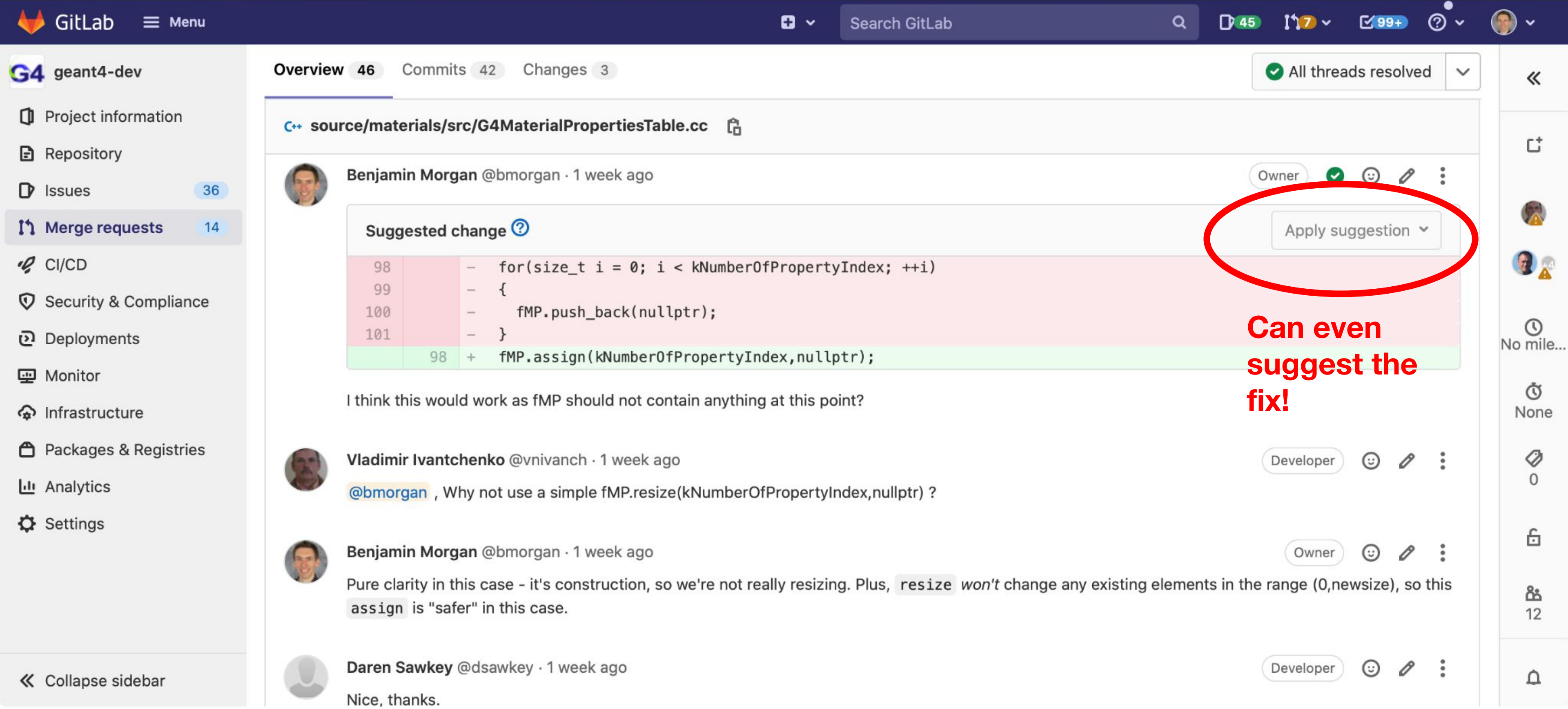
In contrast, material constant properties are G4double. In order to keep track of whether these are defined, a boolean is added. The boolean is true if the material constant property is defined. The vector of material constant properties is really a vector to std::pair<G4double, G4bool>.

Also some cleaning: variable names start with a lower case letter, and class members start with f (not G4!). Use C++11 style iteration and remove the private member iterators.

Edited 4 days ago by Daren Sawkey

Writing good Commit Messages and MR Descriptions

Really important for MR process and later maintenance



source/materials/src/G4MaterialPropertiesTable.cc

Benjamin Morgan @bmorgan · 1 week ago — Owner

Suggested change — Apply suggestion

```
 98      - for(size_t i = 0; i < kNumberOfPropertyIndex; ++i)
 99      - {
100      -   fMP.push_back(nullptr);
101      - }
     98  + fMP.assign(kNumberOfPropertyIndex,nullptr);
```

I think this would work as fMP should not contain anything at this point?

Can even suggest the fix!

Vladimir Ivantchenko @vnivanch · 1 week ago — Developer

@bmorgan , Why not use a simple fMP.resize(kNumberOfPropertyIndex,nullptr) ?

Benjamin Morgan @bmorgan · 1 week ago — Owner

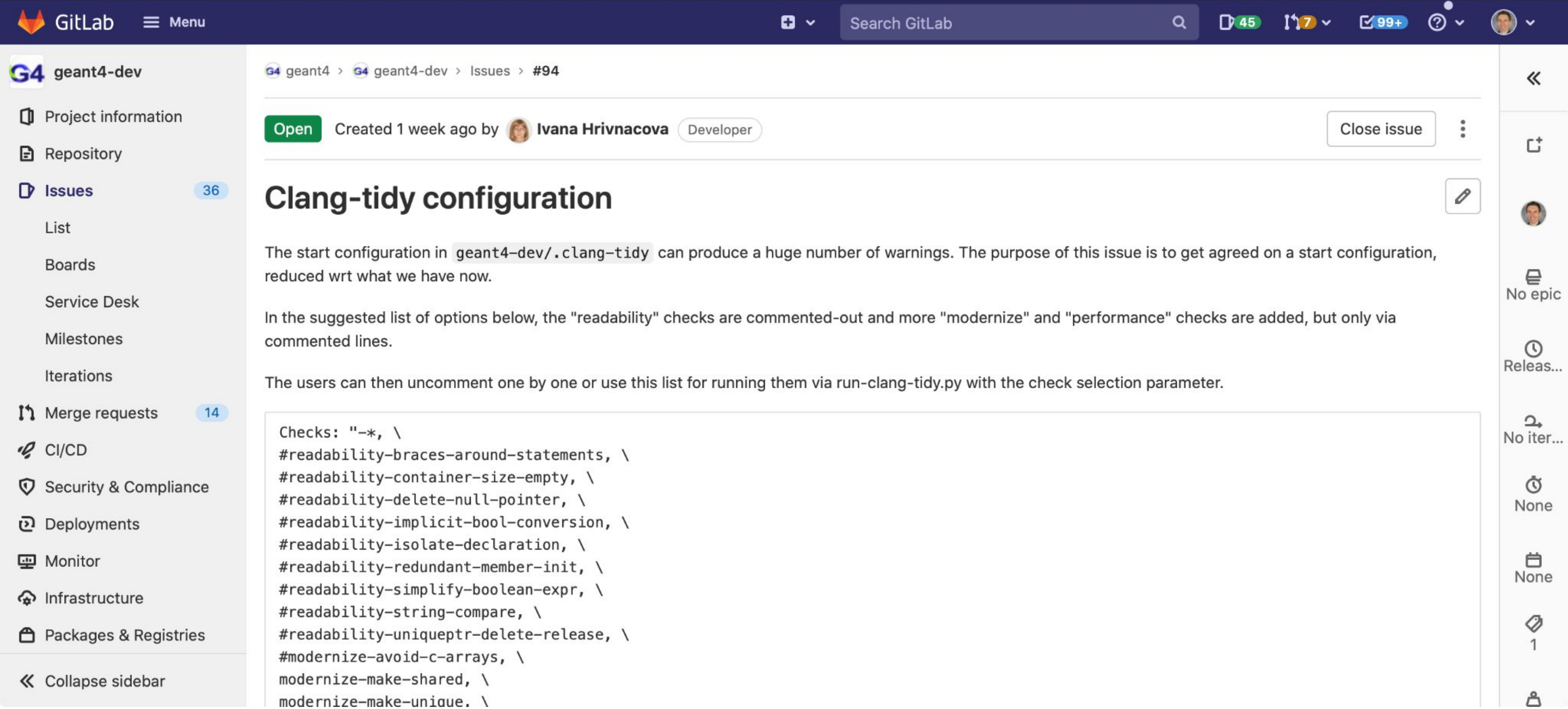
Pure clarity in this case - it's construction, so we're not really resizing. Plus, `resize` *won't* change any existing elements in the range (0,newsize), so this `assign` is "safer" in this case.

Daren Sawkey @dsawkey · 1 week ago — Developer

Nice, thanks.

Use the Review Process!

Fast and effective way to work **together** to clarify changes, and improve Geant4 **during development**



Clang-tidy configuration

The start configuration in `geant4-dev/.clang-tidy` can produce a huge number of warnings. The purpose of this issue is to get agreed on a start configuration, reduced wrt what we have now.

In the suggested list of options below, the "readability" checks are commented-out and more "modernize" and "performance" checks are added, but only via commented lines.

The users can then uncomment one by one or use this list for running them via run-clang-tidy.py with the check selection parameter.

```
Checks: "-*, \
#readability-braces-around-statements, \
#readability-container-size-empty, \
#readability-delete-null-pointer, \
#readability-implicit-bool-conversion, \
#readability-isolate-declaration, \
#readability-redundant-member-init, \
#readability-simplify-boolean-expr, \
#readability-string-compare, \
#readability-uniqueptr-delete-release, \
#modernize-avoid-c-arrays, \
modernize-make-shared, \
modernize-make-unique, \
```

Use Issues for Problems *and* Discussion *and* Work Items

You can link MRs to Issues! Start discussion in Issue, code in MR, Issue closes when MR(s) merged

Issues vs Merge Requests vs Reviews

- Has been some discussion about when/if to submit MR vs an Issue, especially for cross working group topics.
- There is no absolute rule here, only guidance:
  - *Never hesitate to submit a MR for any category, as it will always be seen/reviewed by coordinators…*
  - *… but they are free to outright reject, or heavily review it, or ask for further discussion through an Issue*
  - *It's on you, the developer, to balance whether to start discussion on a change in an Issue, or just submit in a MR, depending on the scale/amount of work involved*
- Clear **collaboration-wide communication** is key, and whilst discussions might start over coffee/in the office/email, progression to **actual development work should be recorded, somehow, in GitLab**
  - *Also valuable for "we discussed this before and it was a no-go, see here"*
  - *GitLab Boards/Milestones can also be valuable to track what's in progress/done.*

Developer Tools: CODING_GUIDELINES.rst

- Updated over past year to bring together existing material/knowledge on a range of day-to-day development topics for Geant4
- **C++ Use and Guidelines**
  - *Recommended use of C++ language/features*
  - *Code formatting with clang-format*
- **Geant4/CMake Build System**
  - *Code organization into "source code modules" and libraries*
  - *Managing "source code module" dependencies*
- **Static Analysis/Debugging**
  - *Use of Coverity*
  - *Use of clang-tidy*
  - *Sanitizers and backtracking*
- ***As always it's a living document, so MRs/Issues on it very welcome as is discussion this week! This includes potential final home(s) for the material, like file, web, wiki etc.***

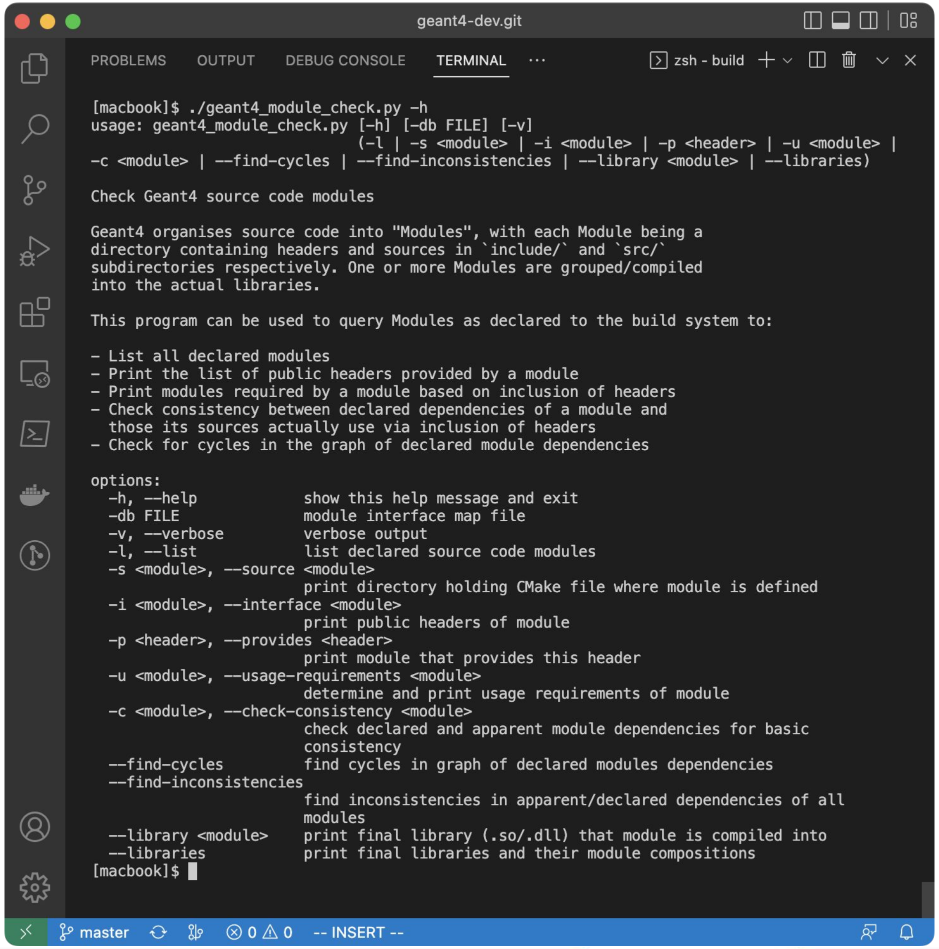
Developer Tools 1:
Source Code Modules

- New section in guidelines doc on how to write `sources.cmake` for your module(s)
  - *geant4_add_module*, *geant4_module_link_libraries* ...

```
# - sources.cmake

geant4_add_module(G4foo
  PUBLIC_HEADERS
    G4Foo.hh
  SOURCES
    G4Foo.cc
)

geant4_module_link_libraries(G4foo
  PUBLIC G4globman
  PRIVATE G4intercoms
)
```

Developer Tools 1: Source Code Modules

- New section in guidelines doc on how to write `sources.cmake` for your module(s)
  - *`geant4_add_module`, `geant4_module_link_libraries` …*
- New `geant4_module_check.py` script to query/check module interfaces, e.g.
  - *What module provides header "X"?*
  - *Are there cycles in module dependencies?*
  - *Are module dependencies correct?*
- Checks added as tests in Continuous and Nightly CI, full instructions on running in local developer builds in `CODING_GUIDELINES.rst`



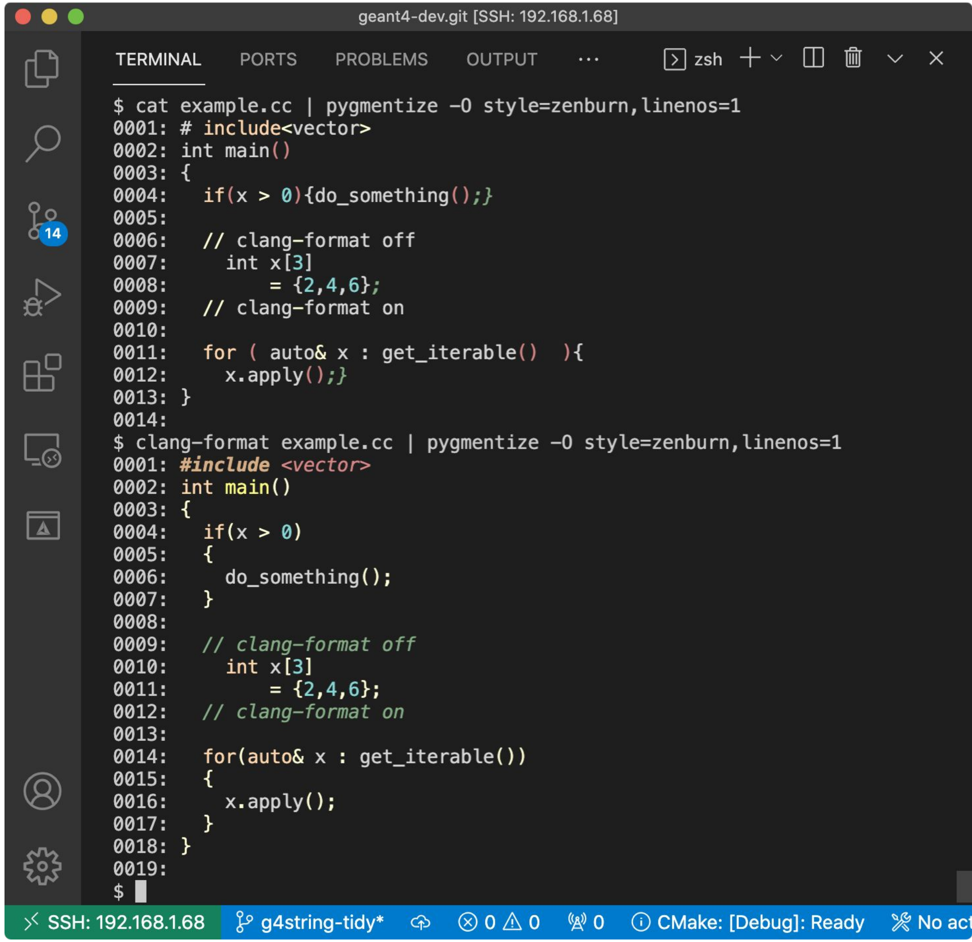
```
[macbook]$ ./geant4_module_check.py -h
usage: geant4_module_check.py [-h] [-db FILE] [-v]
                              (-l | -s <module> | -i <module> | -p <header> | -u <module> |
-c <module> | --find-cycles | --find-inconsistencies | --library <module> | --libraries)

Check Geant4 source code modules

Geant4 organises source code into "Modules", with each Module being a
directory containing headers and sources in `include/` and `src/`
subdirectories respectively. One or more Modules are grouped/compiled
into the actual libraries.

This program can be used to query Modules as declared to the build system to:

- List all declared modules
- Print the list of public headers provided by a module
- Print modules required by a module based on inclusion of headers
- Check consistency between declared dependencies of a module and
  those its sources actually use via inclusion of headers
- Check for cycles in the graph of declared module dependencies

options:
  -h, --help            show this help message and exit
  -db FILE              module interface map file
  -v, --verbose         verbose output
  -l, --list            list declared source code modules
  -s <module>, --source <module>
                        print directory holding CMake file where module is defined
  -i <module>, --interface <module>
                        print public headers of module
  -p <header>, --provides <header>
                        print module that provides this header
  -u <module>, --usage-requirements <module>
                        determine and print usage requirements of module
  -c <module>, --check-consistency <module>
                        check declared and apparent module dependencies for basic
                        consistency
  --find-cycles         find cycles in graph of declared modules dependencies
  --find-inconsistencies
                        find inconsistencies in apparent/declared dependencies of all
                        modules
  --library <module>    print final library (.so/.dll) that module is compiled into
  --libraries           print final libraries and their module compositions
[macbook]$ 
```

Developer Tools 2: Modularization and GNUMake

- CMake “modular” build support added as part of work towards restructuring composition of Geant4 libraries from source code modules
  - *“Modular” build run only in Continuous CI to confirm no missing transient dependencies between source code modules (with 1 library == 1 module)*
  - ***Full work on library restructure delayed due to lack of time (but see GitLab #122)***
- This CI build, plus the checks CMake and `geant4_module_check.py` implement, make the GNUmake system to build/test Geant4 obsolete
- **All `GNUmakefiles` under `source/` and `tests/` will therefore be removed after the 11.1 Release.**
- **The GNUmake system itself will be retained, albeit still deprecated, for building user applications**
  - *Full removal here depends on implementing `pkg-config` files for Geant4, which is easiest done as part of the library restructure*

Developer Tools 3: clang-format

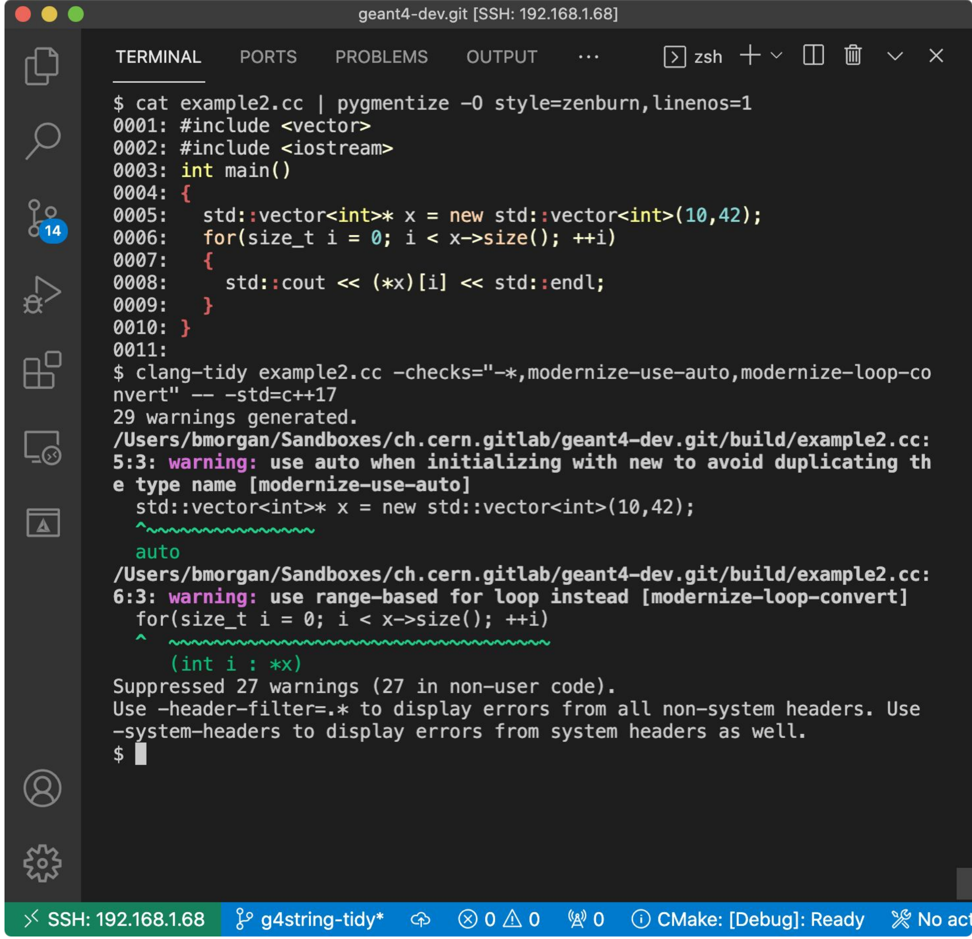
- Tool for formatting C++ code, style from .clang-format file in project root
- **Why? Consistency and clarity across the project so focus is what code does**
  - ***Geant4 is not consistent, in some places not even within single files!***
- Easy to **disable**, e.g. for array/matrix data, with special comment blocks
- `.clang-format` developed to **match de-facto Geant4 style, minimize changes**.
- Full rollout post-11.1 release?
  - *Single one-off reformat or gradual?*
  - *Tooling in Git/GitLab to assist, e.g. "Do: reformat" MR command*



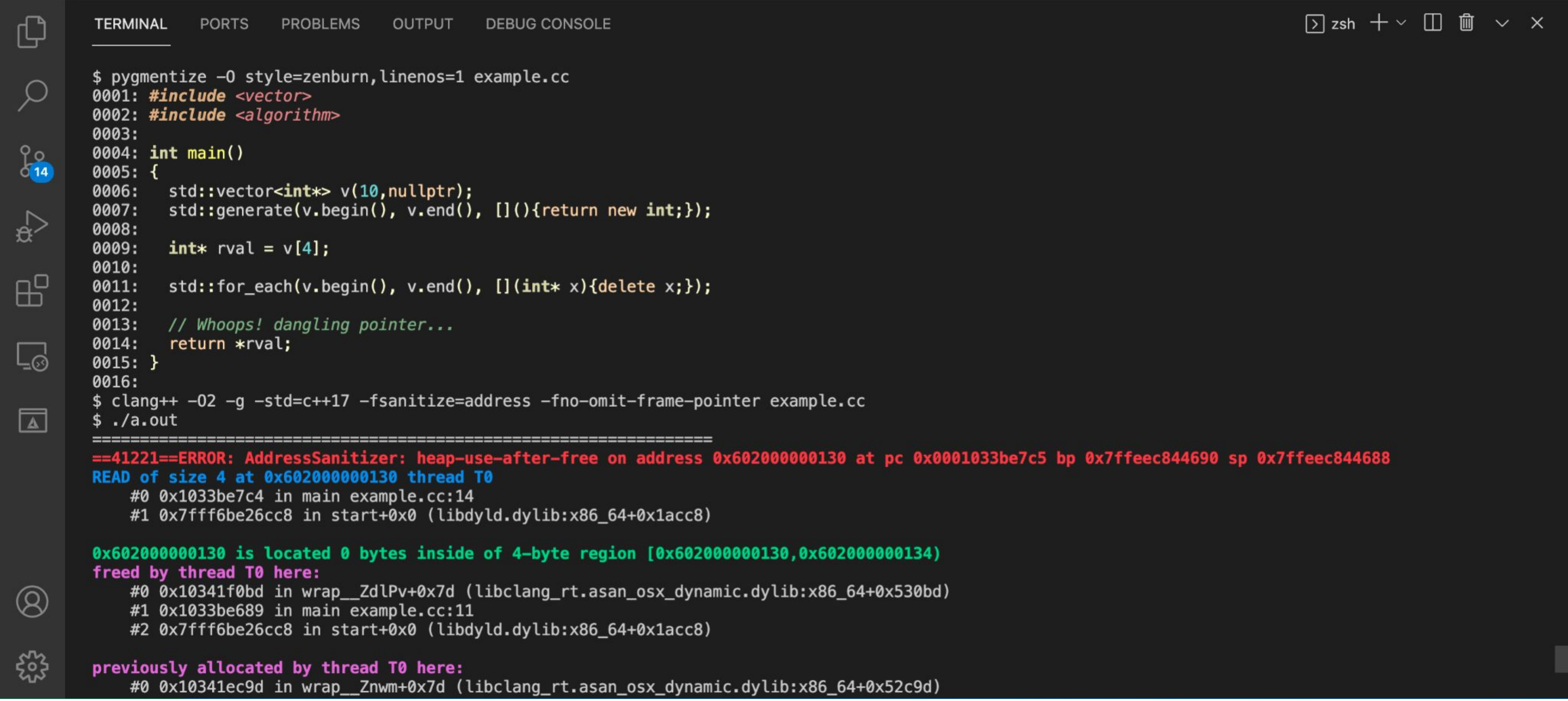
```
$ cat example.cc | pygmentize -O style=zenburn,linenos=1
0001: # include<vector>
0002: int main()
0003: {
0004:   if(x > 0){do_something();}
0005:
0006:   // clang-format off
0007:     int x[3]
0008:         = {2,4,6};
0009:   // clang-format on
0010:
0011:   for ( auto& x : get_iterable()  ){
0012:     x.apply();}
0013: }
0014:
$ clang-format example.cc | pygmentize -O style=zenburn,linenos=1
0001: #include <vector>
0002: int main()
0003: {
0004:   if(x > 0)
0005:   {
0006:     do_something();
0007:   }
0008:
0009:   // clang-format off
0010:     int x[3]
0011:         = {2,4,6};
0012:   // clang-format on
0013:
0014:   for(auto& x : get_iterable())
0015:   {
0016:     x.apply();
0017:   }
0018: }
0019:
$
```

Developer Tools 4: clang-tidy

- Tool for linting C++ code with a set of checks for "better practice" such as clarity, modernity, performance.
  - *Checks in `.clang-tidy` file or supplied on command line*
- **Why? Assist developer to identify code that could be improved**
  - ***Consistency as well - checks can match/complement coding guidelines***
  - ***Like -format, can disable if required***
- Gradual rollout through kernel categories, testing/iterating on checks and guidelines
  - *See CODING_GUIDELINES.rst for use guide and current recommended, suggested, optional checks*



```
$ cat example2.cc | pygmentize -O style=zenburn,linenos=1
0001: #include <vector>
0002: #include <iostream>
0003: int main()
0004: {
0005:   std::vector<int>* x = new std::vector<int>(10,42);
0006:   for(size_t i = 0; i < x->size(); ++i)
0007:   {
0008:     std::cout << (*x)[i] << std::endl;
0009:   }
0010: }
0011:
$ clang-tidy example2.cc -checks="-*,modernize-use-auto,modernize-loop-co
nvert" -- -std=c++17
29 warnings generated.
/Users/bmorgan/Sandboxes/ch.cern.gitlab/geant4-dev.git/build/example2.cc:
5:3: warning: use auto when initializing with new to avoid duplicating th
e type name [modernize-use-auto]
  std::vector<int>* x = new std::vector<int>(10,42);
  ^~~~~~~~~~~~~~~~~
  auto
/Users/bmorgan/Sandboxes/ch.cern.gitlab/geant4-dev.git/build/example2.cc:
6:3: warning: use range-based for loop instead [modernize-loop-convert]
  for(size_t i = 0; i < x->size(); ++i)
  ^   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     (int i : *x)
Suppressed 27 warnings (27 in non-user code).
Use -header-filter=.* to display errors from all non-system headers. Use
-system-headers to display errors from system headers as well.
$
```



```
$ pygmentize -O style=zenburn,linenos=1 example.cc
0001: #include <vector>
0002: #include <algorithm>
0003:
0004: int main()
0005: {
0006:   std::vector<int*> v(10,nullptr);
0007:   std::generate(v.begin(), v.end(), [](){return new int;});
0008:
0009:   int* rval = v[4];
0010:
0011:   std::for_each(v.begin(), v.end(), [](int* x){delete x;});
0012:
0013:   // Whoops! dangling pointer...
0014:   return *rval;
0015: }
0016:
$ clang++ -O2 -g -std=c++17 -fsanitize=address -fno-omit-frame-pointer example.cc
$ ./a.out
=================================================================
==41221==ERROR: AddressSanitizer: heap-use-after-free on address 0x602000000130 at pc 0x0001033be7c5 bp 0x7ffeec844690 sp 0x7ffeec844688
READ of size 4 at 0x602000000130 thread T0
    #0 0x1033be7c4 in main example.cc:14
    #1 0x7fff6be26cc8 in start+0x0 (libdyld.dylib:x86_64+0x1acc8)

0x602000000130 is located 0 bytes inside of 4-byte region [0x602000000130,0x602000000134)
freed by thread T0 here:
    #0 0x10341f0bd in wrap__ZdlPv+0x7d (libclang_rt.asan_osx_dynamic.dylib:x86_64+0x530bd)
    #1 0x1033be689 in main example.cc:11
    #2 0x7fff6be26cc8 in start+0x0 (libdyld.dylib:x86_64+0x1acc8)

previously allocated by thread T0 here:
    #0 0x10341ec9d in wrap__Znwm+0x7d (libclang_rt.asan_osx_dynamic.dylib:x86_64+0x52c9d)
```

Developer Tools 5: Sanitizers

CMake option `GEANT4_BUILD_SANITIZER` to select Address, Thread or UB. More info in `CODING_GUIDELINES.rst`

Packaging: Official packages, CPack, DEB, RPM

- Official packages (also discussed in Issue #80)
  - *Available in Arch Linux, Conda, Gentoo Linux, Mac Ports, NixOS, and Spack*
- CMake build system can build binaries using CPack
  - *Just run* `cpack -G TYPE`*, where* TYPE *is one of* TGZ, DEB, RPM*, etc*
  - *Needs appropriate config for dependencies to be added automatically*
- Used RPM Packaging Guide to create starting point for official distribution
  - *Created SPEC file with geant4, geant4-{data,devel,examples} packages*
  - *Published experimental RPM repo for CS8 built from the SPEC file at http://lcgpackages.web.cern.ch/lcgpackages/test/geant4*
  - *Post in the Geant4 Forum to let users try out the experimental packages*
  - *RPM packages can be used for building container images with Geant4*
- ***Needs volunteers and effort to maintain and evolve***
  - *Community effort as well, but good to have official packages for common platforms*

Credit: Guilherme Amadio

Packaging: Geant4 Docker Images

- Original work by A. Dotti and W. Takase (similar efforts by J. Madsen)
- x86 and ARM images for Geant4 from C. Mancini:
  - *https://hub.docker.com/r/carlomt/geant4 (Images)*
  - *https://github.com/carlomt/docker-geant4 (Sources)*
- Data separate to keep image size small, mounted into container at runtime
- Example of building an image for a user application on top of this:
  - *https://github.com/carlomt/docker-dicom-g4example*
- How best to use/continue this line of packaging?
  - *Could be useful for training, testing/profiling*
  - *RPM/Deb/etc better as "official" binaries, allowing easier user customization of their Docker images*
  - *Issue #80 on GitLab to discuss further*

Credit: Carlo Mancini

Geant4 Dockerfile using experimental RPMs

Dockerfile

```
FROM cern/cs8-base
RUN dnf update -y dnf install -y epel-release
COPY geant4.repo /etc/yum.repos.d/geant4.repo
RUN dnf install -y geant4-devel
```

Contents of geant4.repo

```
[geant4]
name=Geant4 Collaboration
baseurl=http://lcgpackages.web.cern.ch/lcgpackages/test/geant4
enabled=1
gpgcheck=0
priority=10
```

Image pushed to Docker Hub, try it with ***docker run -it geant4/geant4*** *(note: very big! 6.2GB size total)*

Credit: Guilherme Amadio

Key Points

- Will monitor changes to CERN GitLab version, but **no cause for concern** at present
- CODING_GUIDELINES.rst document updated with C++, CMake, and Tooling guides
  - ***Feedback, input very welcome***
- Significant work on binary packaging, **needs volunteers and effort to take it forward**

- One more thing…
  - *… modernization topics and ideas to discuss*

Longer Term Topics: Code Evolution and Spring Clean

Maybe controversial, but intended to motivate discussion this week! Includes some that are likely for next major release - yes, should start thinking about this now!

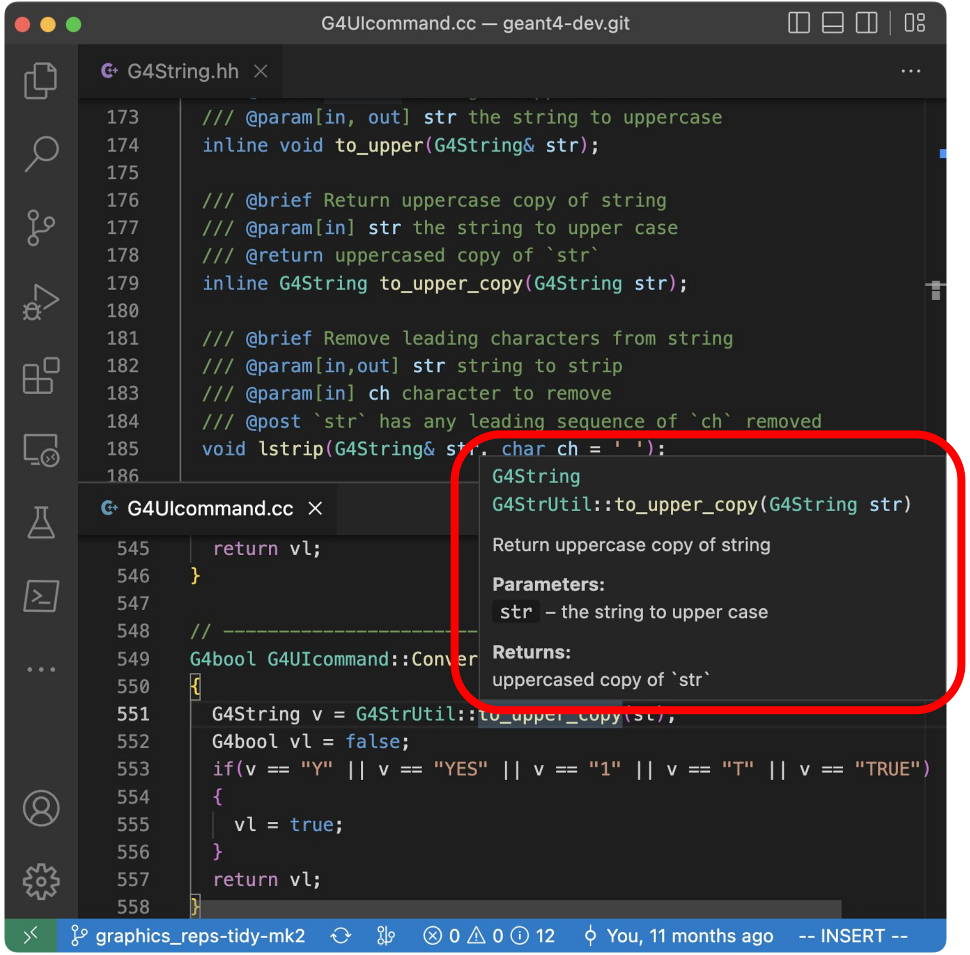
Organizing Test Code

- ***Move per-category test/ directories from source/ to tests/***
  - *Clearer separation of source and test code*
  - *Identify obsolete/redundant tests, or those suitable for use in CI*
  - *Identify potential for use of unit testing using, e.g. Google Test*
- Final arbiter of “correctness” the integration and validation tests, but unit tests provide an extra layer of defence or to add regression checks.
- [GitLab #137 for discussion]

```
- source/
  - .. move testing code from here to ..
- tests/
  - tools/
    - .. as is
  - integration/
    - test00/
    - .. all current `testXY`
  - examples/
    - CMakeLists.txt
    - .. current `tests/ctests/CMakeLists.txt`
  - source/
    - global/
      - management/
        - .. source/global/management/test files
        - .. or under "integration" if appropriate
```

Doxygen-style C++ interface docstrings

- ***Interface docstrings do exist, but are incompatible with Doxygen as strings appear after the declaration (tossed a coin and lost…)***
- Initial benefit for **developers**:
  - *Integration with IDE Intellisense*
  - *Clarify interface **contracts**, e.g. in/out params, ownership, pre/post conditions*
- Longer term, Doxygen docs would nicely complement other Docs and LXR for **users**
- Can be done little by little (and mostly copy paste!), **focus on core kernel/user interfaces**
- **GitLab #87 for Discussion**



```
/// @param[in, out] str the string to uppercase
inline void to_upper(G4String& str);

/// @brief Return uppercase copy of string
/// @param[in] str the string to upper case
/// @return uppercased copy of `str`
inline G4String to_upper_copy(G4String str);

/// @brief Remove leading characters from string
/// @param[in,out] str string to strip
/// @param[in] ch character to remove
/// @post `str` has any leading sequence of `ch` removed
void lstrip(G4String& str, char ch = ' ');
```

```
  return vl;
}

// -----------------------
G4bool G4UIcommand::Conver
{
  G4String v = G4StrUtil::to_upper_copy(st);
  G4bool vl = false;
  if(v == "Y" || v == "YES" || v == "1" || v == "T" || v == "TRUE")
  {
    vl = true;
  }
  return vl;
}
```

```
G4String
G4StrUtil::to_upper_copy(G4String str)
```

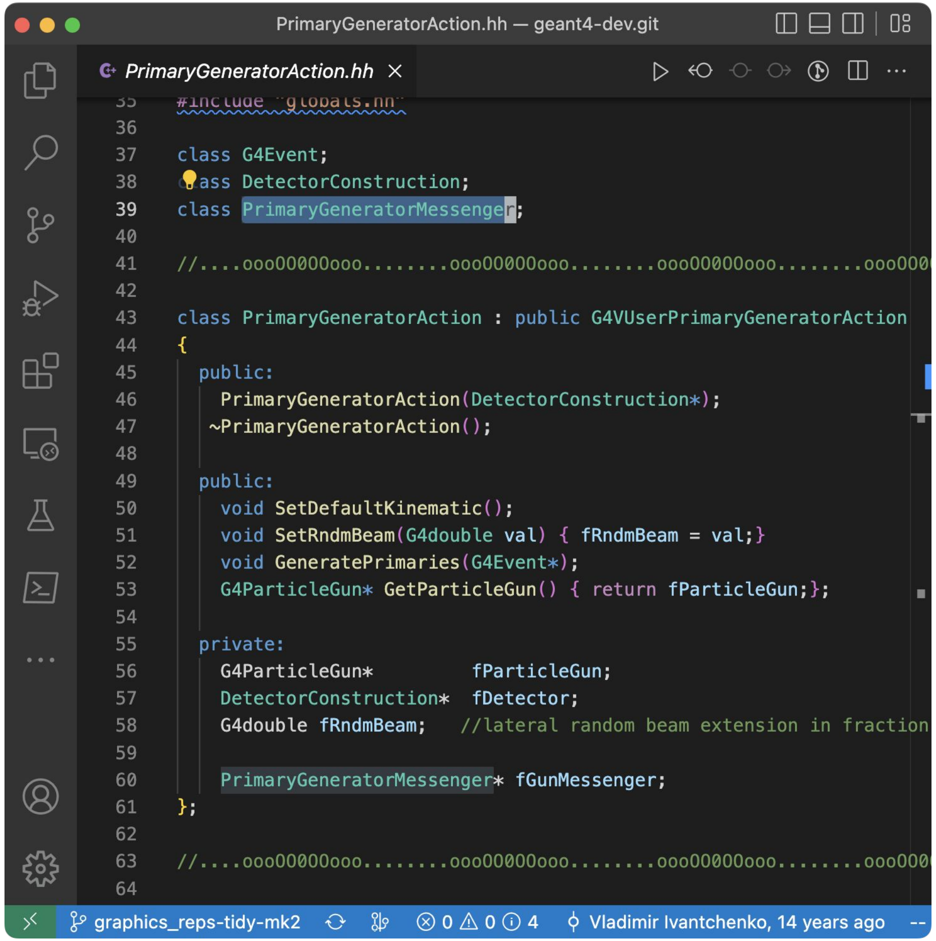
Return uppercase copy of string

Parameters:
`str` – the string to upper case

Returns:
uppercased copy of `str`

Hiding Implementation Details

- ***How many of our ~3500 headers are for interfaces never used/for use outside of the same module?***
  - *Canonical example: messengers*
- Hiding such “private” interfaces:
  - *Clarifies interface to library consumers (don’t need to understand that .hh)*
  - *Potential for (small) performance improvements from symbol hiding*
  - *Greater freedom in implementation: no user interface change!*
- Start reviewing your modules to see if it has private interfaces.


PrimaryGeneratorAction.hh — geant4-dev.git
PrimaryGeneratorAction.hh
graphics_reps-tidy-mk2
0 0 4
Vladimir Ivantchenko, 14 years ago

```
#include "globals.hh"

class G4Event;
class DetectorConstruction;
class PrimaryGeneratorMessenger;

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0

class PrimaryGeneratorAction : public G4VUserPrimaryGeneratorAction
{
  public:
    PrimaryGeneratorAction(DetectorConstruction*);
   ~PrimaryGeneratorAction();

  public:
    void SetDefaultKinematic();
    void SetRndmBeam(G4double val) { fRndmBeam = val;}
    void GeneratePrimaries(G4Event*);
    G4ParticleGun* GetParticleGun() { return fParticleGun;};

  private:
    G4ParticleGun*         fParticleGun;
    DetectorConstruction*  fDetector;
    G4double fRndmBeam;    //lateral random beam extension in fraction

    PrimaryGeneratorMessenger* fGunMessenger;
};

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0
```

Clarifying Ownership, reducing Globals

- Toolkit uses raw pointers (+new/delete) extensively, canonical use case being inheritance (ptr-to-base), some others - all valid, but…
  - *Ownership of new-d resource not always obvious, even confusing, especially when passing raw pointers around*
  - *Some awkwardness in code with collections of raw pointers (const-ness/de-refing)*
- A few things we could start doing/investigating/profiling:
  - *Values/`std::unique_ptr`/`std::optional` provide as cleaner solution in some places?*
  - *Docstrings to clarify ownership of in/out raw pointers?*
  - *Add interfaces for, e.g. collection of owned raw pointers, or use/import tools like GuidelineSupportLibrary or ranges (range-v3/C++20)?*
- A related topic is the toolkit's extensive use of globals/statics (as raw ptrs!), which can lead to unexpected behaviour, so where and how could we reduce use of these?
  - *Long term, but doesn't mean we shouldn't start thinking about it!*
  - ***GitLab #140 for initial discussion***

Questions, Discussion