IHPCSS Kobe 2024

SOFTWARE ENGINEERING

ERIK LINDAHL

WHO'S ERIK?

Professor of Biophysics

PhD KTH 2001

Postdocs in NL, US, FR

Faculty at Stockholm University 2005

2nd appointment at KTH 2010

30-person research division; experiments & theory work

Research: Membrane proteins, molecular dynamics simulation

Management of large research environments and projects - recruitment/strategy

Main worldwide research impact of my team is SOFTWARE

INTENDED LEARNING OUTCOMES

Help you develop strategies to be systematic in your coding

Awareness of tools, services and documentation that’s available

Improve the quality of your code

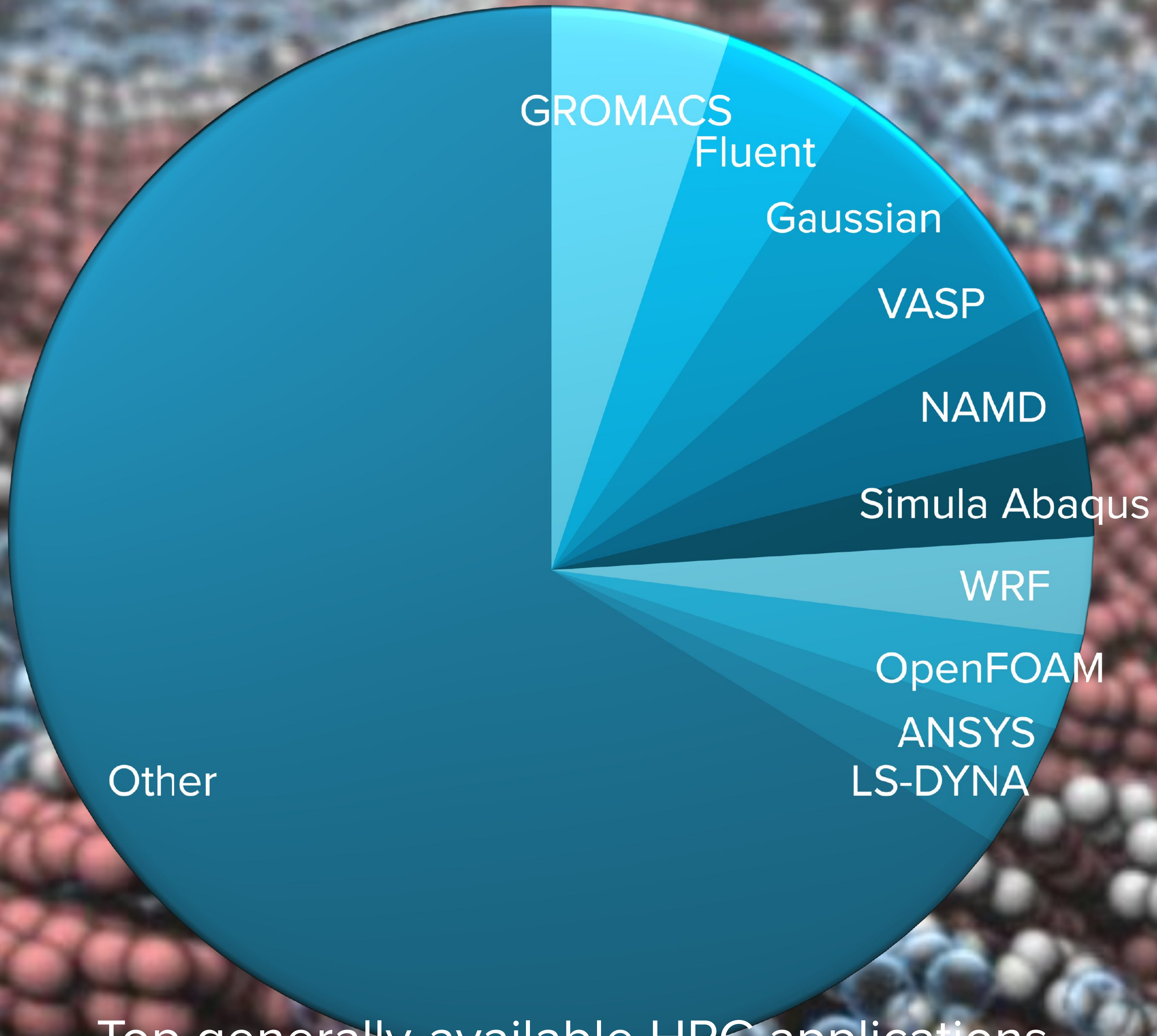
Improve your software productivity

Learning how to work in teams on software

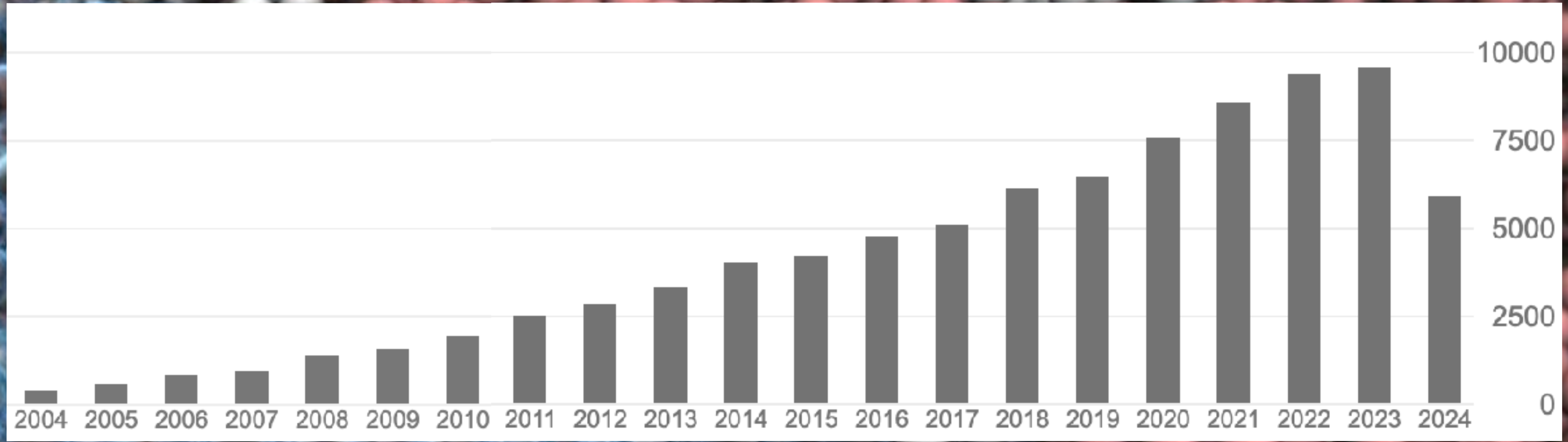
Getting other people to trust and use your code

Achieving impact from your work

Improving your career opportunities



Top generally-available HPC applications
[Intersect360 Research 2017]

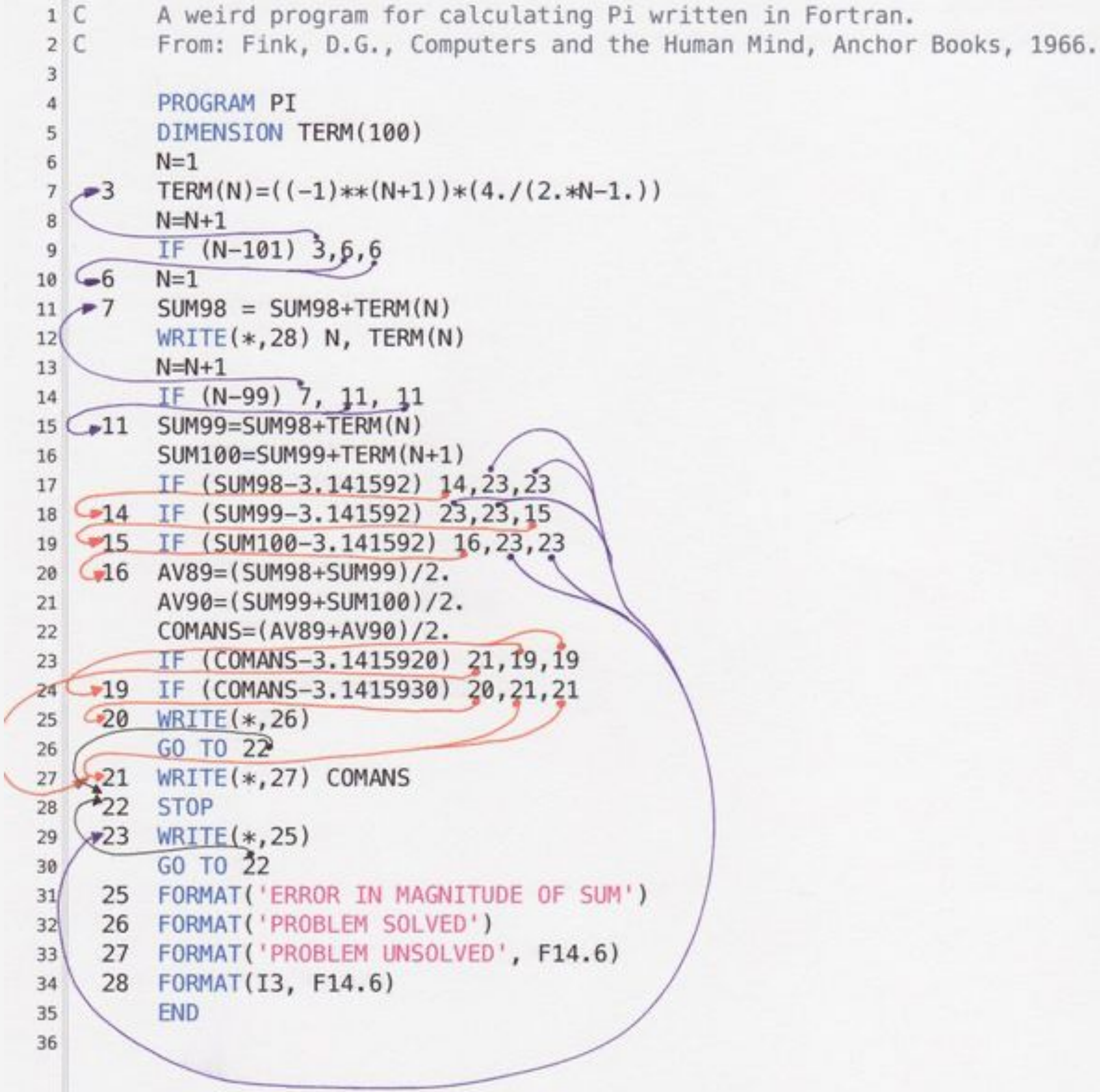


EXPERIENCE FROM 25 YEARS OF GROMACS DEVELOPMENT

- Simulation hardware project, turned software
- Early development based on our own needs
- Turned GPL in 2001, LGPL in 2012
- Organic growth: ~15 core developers, 15-25 contributors
- Currently 2600450 lines of C++17 code
  - For certain definitions of “C++17”
  - Over the years we’ve used Fortran, C, Assembly
- Lots of old code. Lots of new code. Lots of complicated (read: bad) code written by scientists

2011: Successful,
but increasingly painful

Source code control:
CVS
Build chain:
Automake/autoconf/libtool
Bug tracking:
Bugzilla
Testing:



```
C     A weird program for calculating Pi written in Fortran.
C     From: Fink, D.G., Computers and the Human Mind, Anchor Books, 1966.

      PROGRAM PI
      DIMENSION TERM(100)
      N=1
  3   TERM(N)=((-1)**(N+1))*(4./(2.*N-1.))
      N=N+1
      IF (N-101) 3,6,6
  6   N=1
  7   SUM98 = SUM98+TERM(N)
      WRITE(*,28) N, TERM(N)
      N=N+1
      IF (N-99) 7, 11, 11
  11  SUM99=SUM98+TERM(N)
      SUM100=SUM99+TERM(N+1)
      IF (SUM98-3.141592) 14,23,23
  14  IF (SUM99-3.141592) 23,23,15
  15  IF (SUM100-3.141592) 16,23,23
  16  AV89=(SUM98+SUM99)/2.
      AV90=(SUM99+SUM100)/2.
      COMANS=(AV89+AV90)/2.
      IF (COMANS-3.1415920) 21,19,19
  19  IF (COMANS-3.1415930) 20,21,21
  20  WRITE(*,26)
      GO TO 22
  21  WRITE(*,27) COMANS
  22  STOP
  23  WRITE(*,25)
      GO TO 22
  25  FORMAT('ERROR IN MAGNITUDE OF SUM')
  26  FORMAT('PROBLEM SOLVED')
  27  FORMAT('PROBLEM UNSOLVED', F14.6)
  28  FORMAT(I3, F14.6)
      END
```

> “The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software, and the study of these approaches, that is, the application of engineering to software.”

PIERRE BOURQUE & ROBERT DUPUIS / IEEE

SCIENTIST MENTALITY	SOFTWARE ENGINEER MENTALITY
Trained in physics, chemistry, etc.	Trained in CS/software
Cares about their problem	Cares about their code
Cares about short-term deadlines (next paper!)	Cares about long-term maintenance (next machine)
New code = asset	New code = liability
Often codes on their own	Often codes in a team
Writes more code than she reads	Reads much more code than she writes

Without proper software engineering, you are taking on **technical debt** that sooner or later will have to be repaid

““““Technical Debt is a wonderful metaphor developed by Ward Cunningham to help us think about this problem. In this metaphor, doing things the quick and dirty way sets us up with a technical debt, which is similar to a financial debt. **Like a financial debt, the technical debt incurs interest payments**, which come in the form of the extra effort that we have to do in future development because of the quick and dirty design choice. **We can choose to continue paying the interest, or we can pay down the principal by refactoring** the quick and dirty design into the better design. Although it costs to pay down the principal, we gain by reduced interest payments in the future.””””

MARTIN FOWLER

OPEN SOURCE & GOOD SOFTWARE ENGINEERING MATTER

Physics Today, Aug 22, 2018: *Recollection of Chandler/Limmer vs. Debenedetti 7-year fight over supercooled water; turned out to be algorithm implementation issue in code authors resisted sharing.*

22 Aug 2018 in Research & Technology

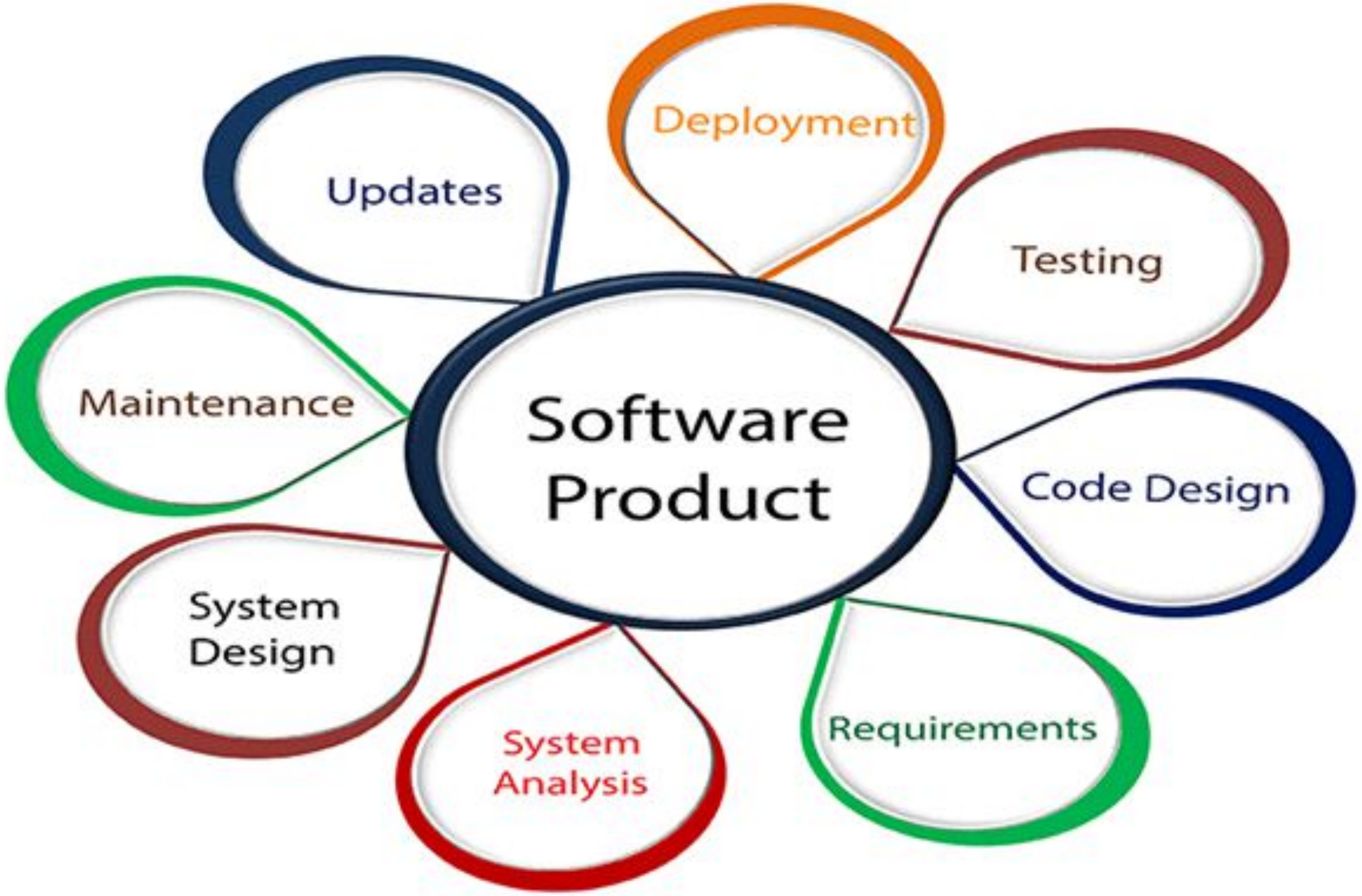
The war over supercooled water

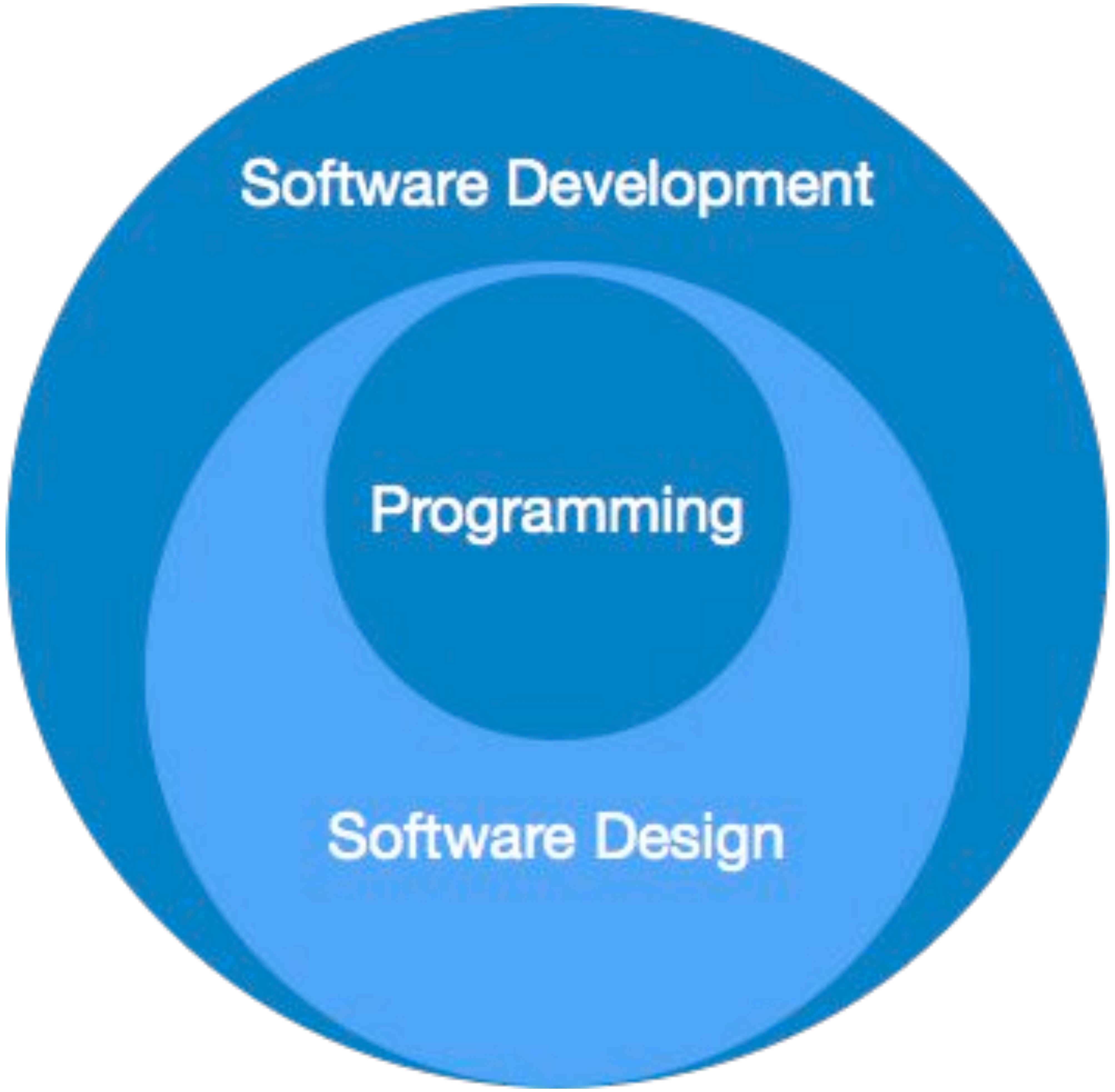
How a hidden coding error fueled a seven-year dispute between two of condensed matter's top theorists.

Ashley G. Smart

Limmer maintains that he and his mentor weren't trying to hide anything. "I had and was very willing to share the code," he says. What he didn't have, he says, was the time or personnel to prepare the code in a form that could be useful to an outsider. "When Debenedetti's group was making their code available," Limmer explains, "he brought people in to clean up the code and document and run tests on it. He had resources available to him to be able to do that." At Berkeley, "it was just me trying to get that done myself."

"One of the real travesties is that there's no way you could have reproduced [the Berkeley team's] algorithm—the way they had implemented their code—from reading their paper. If this had been disclosed, this saga might not have gone on for seven years."


Software Product
Deployment
Updates
Testing
Maintenance
Code Design
System Design
System Analysis
Requirements


Software Development
Programming
Software Design

LAWS OF EVOLUTION OF GENERAL/E-TYPE SOFTWARE (LEHMAN)

- **Continuing change:** An E-type software system must continue to adapt to the real world changes, else it becomes progressively less useful.
- **Increasing complexity:** As an E-type software system evolves, its complexity tends to increase unless work is done to maintain or reduce it.
- **Conservation of familiarity:** The familiarity with the software or the knowledge about how it was developed, why was it developed in that particular manner etc. must be retained at any cost, to implement the changes in the system.
- **Continuing growth:** In order for an E-type system intended to resolve some business problem, its size of implementing the changes grows according to the lifestyle changes of the business.
- **Reducing quality:** An E-type software system declines in quality unless rigorously maintained and adapted to a changing operational environment.
- **Feedback systems:** The E-type software systems constitute multi-loop, multi-level feedback systems and must be treated as such to be successfully modified or improved.
- **Self-regulation:** E-type system evolution processes are self-regulating with the distribution of product and process measures close to normal.
- **Organizational stability:** The average effective global activity rate in an evolving E-type system is invariant over the lifetime of the product.

CHARACTERISTICS OF GOOD SOFTWARE

Operational	Transitional	Maintenance
Budget	Portability	Modularity
Usability	Interoperability	Maintainability
Efficiency	Reusability	Flexibility
Correctness	Adaptability	Scalability
Functionality		
Dependability		
Security		
Safety		

PLAN
BUILD
CONTINUOUS
INTEGRATION
DEPLOY
OPERATE
CONTINUOUS FEEDBACK
REAL-TIME COMMUNICATION
REAL-TIME COMMUNICATION
draw.io
asana
PivotalTracker
box
Lucidchart
planio
flowdock
Google Drive
Wrike
Google Docs
smartsheet
Office
gliffy
Basecamp
Trello
Dropbox
Microsoft Teams
(x) matters
zoominfo.
CODE CLIMATE
git
GitLab
JFrog
npm
kubernetes
docker
Visual Studio Team Foundation Server
TeamCity
shippable
HashiCorp
amazon web services
ANSIBLE
CODESHIP
Jenkins
buddybuild
circleci
Travis CI
Google Cloud Platform
Microsoft Azure
XL RELEASE
heroku
split
XL DEPLOY
puppet
CHEF
rackspace
OpsGenie
VictorOps
pagerduty
BlueJeans
slack
New Relic.
snyk
bugsnag
Nagios
splunk>
LOGGLY
ZABBIX
salesforce
SUBVERSION
GitHub
Se
Sonatype
MOCHA
FitNesse
node
INTERCOM
SAUCELABS
runnable
ZOHO CRM
ZEPHYR
BrowserStack
freshdesk
QMETRY
servicenow
TestFairy
zendesk
Jasmine
cucumber
Instabug
QASymphony
OMNIDESK
RAYGUN
SENTRY
DATADOG
bigpanda
dynatrace
Rollbar
APPDYNAMICS
[:] SourceClear
LogicMonitor

LESSER ARTISTS BORROW, GREAT ARTISTS STEAL

https://github.com/IHPCSS/software-engineering

Please DO steal this and use it as a template for your own project if you want to!

When the simple repo is not advanced enough, find inspiration from large codes:

https://gitlab.com/gromacs/gromacs

(ADVANCED) SOURCE CODE REVISION CONTROL

What changed in our code last week?

- 49 commits
- 183 files changed
- 5,375 line insertions
- 3,320 line deletions

What changed in our code since January 1?

- 671 commits
- 5,752 files changes
- 157,177 line insertions
- 1,622,410 line deletions (removed assembly kernels)

How do you start debugging when the code now crashes?
Your program worked last week, but now there’s something wrong…
What if it crashes with -O3, but works fine when you add the debug flag?

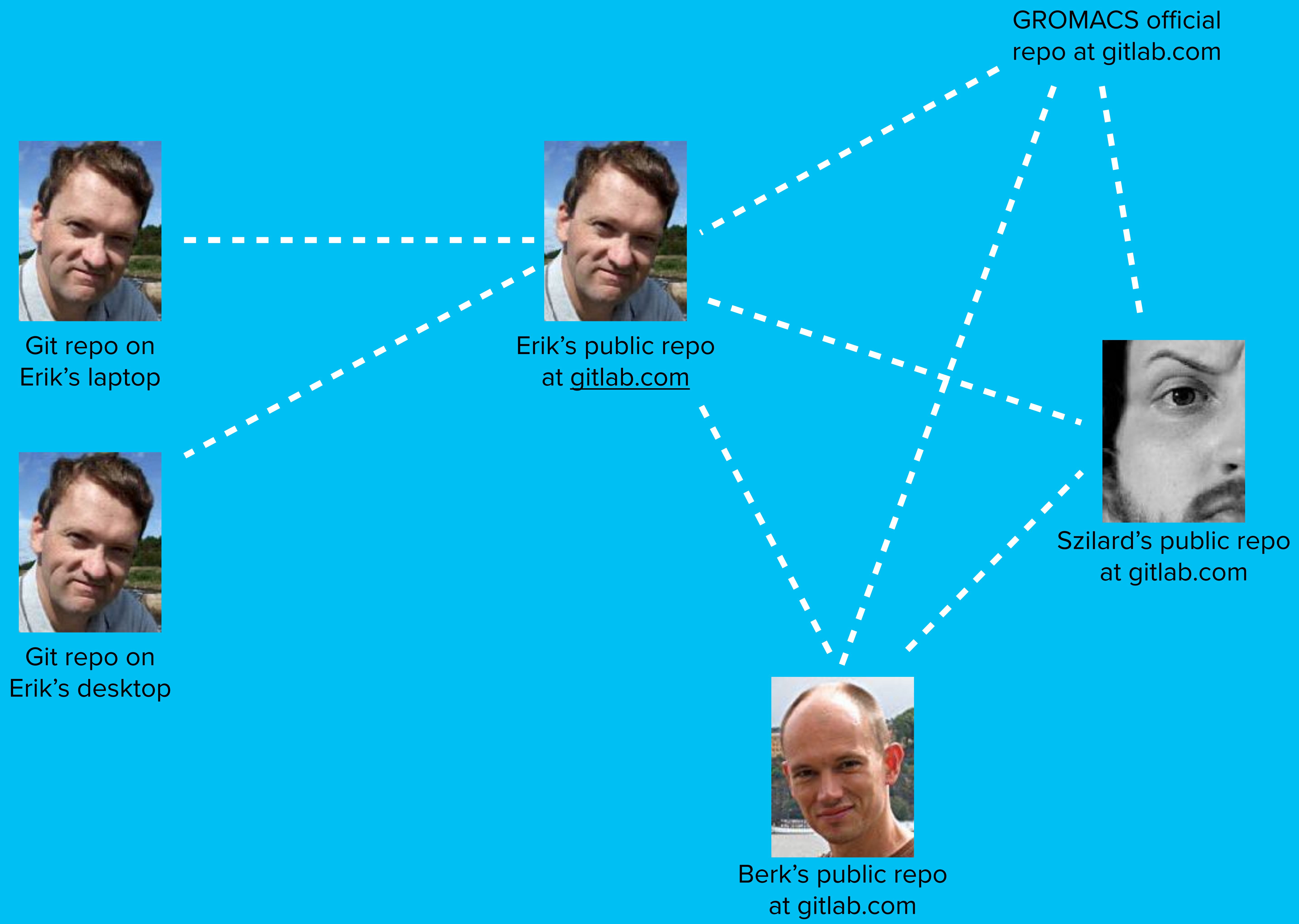
WHAT GIT WILL GIVE YOU

- Handles multiple developers beautifully
- Handles multiple feature branches in parallel with a stable production-quality one
- Develop based on features, not source files
- Pull/push patches between branches
- Revert a specific stupid thing I did 6 months ago, without changing subsequent patches
- Bisect changes to find which one of (say) 1,500 patches caused a bug
- Drawback: VERY steep learning curve to move beyond trivial usage

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

GIT CULTURE

Recommendation: Use gitlab.com



GROMACS official repo at gitlab.com

Git repo on Erik's laptop

Erik's public repo at gitlab.com

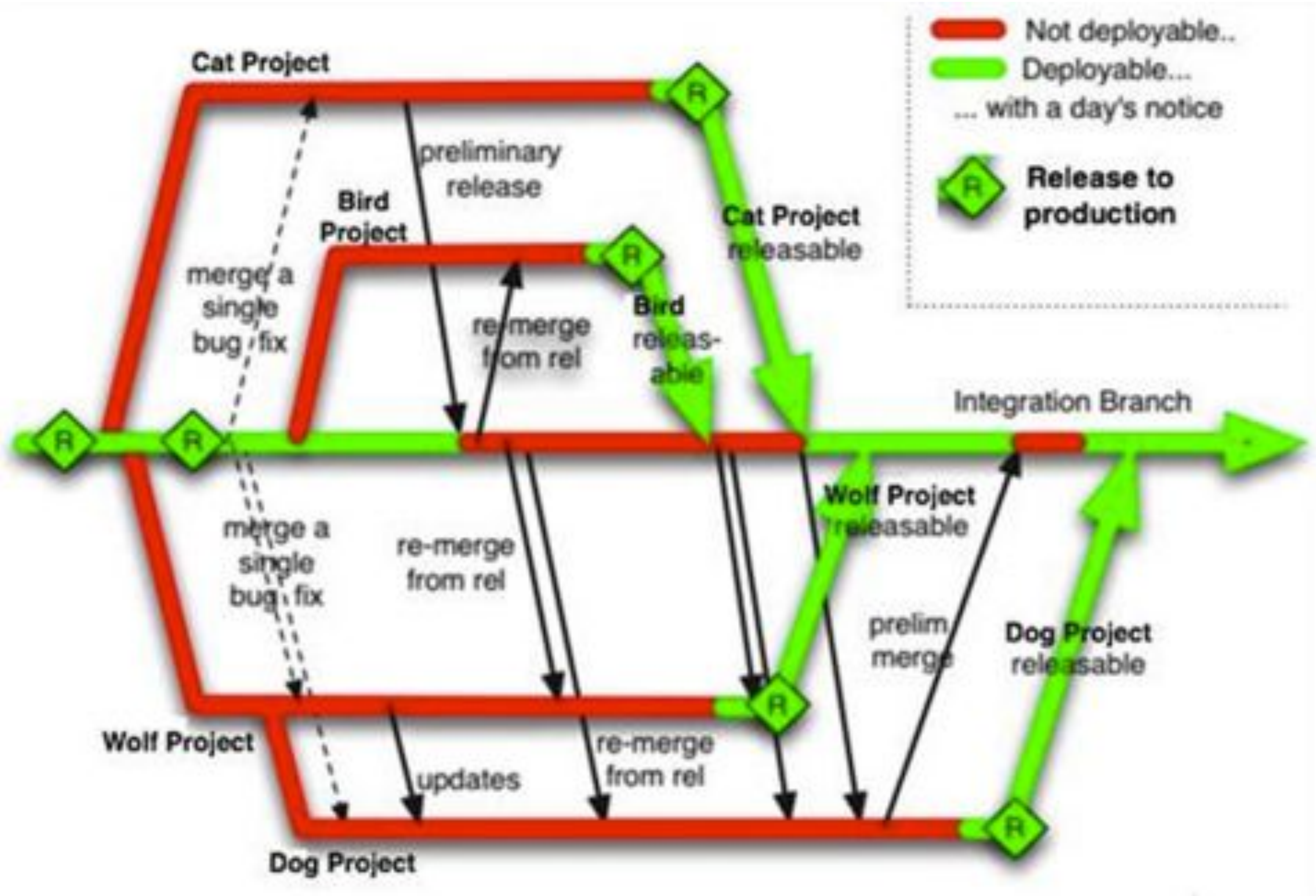
Git repo on Erik's desktop

Szilard's public repo at gitlab.com

Berk's public repo at gitlab.com

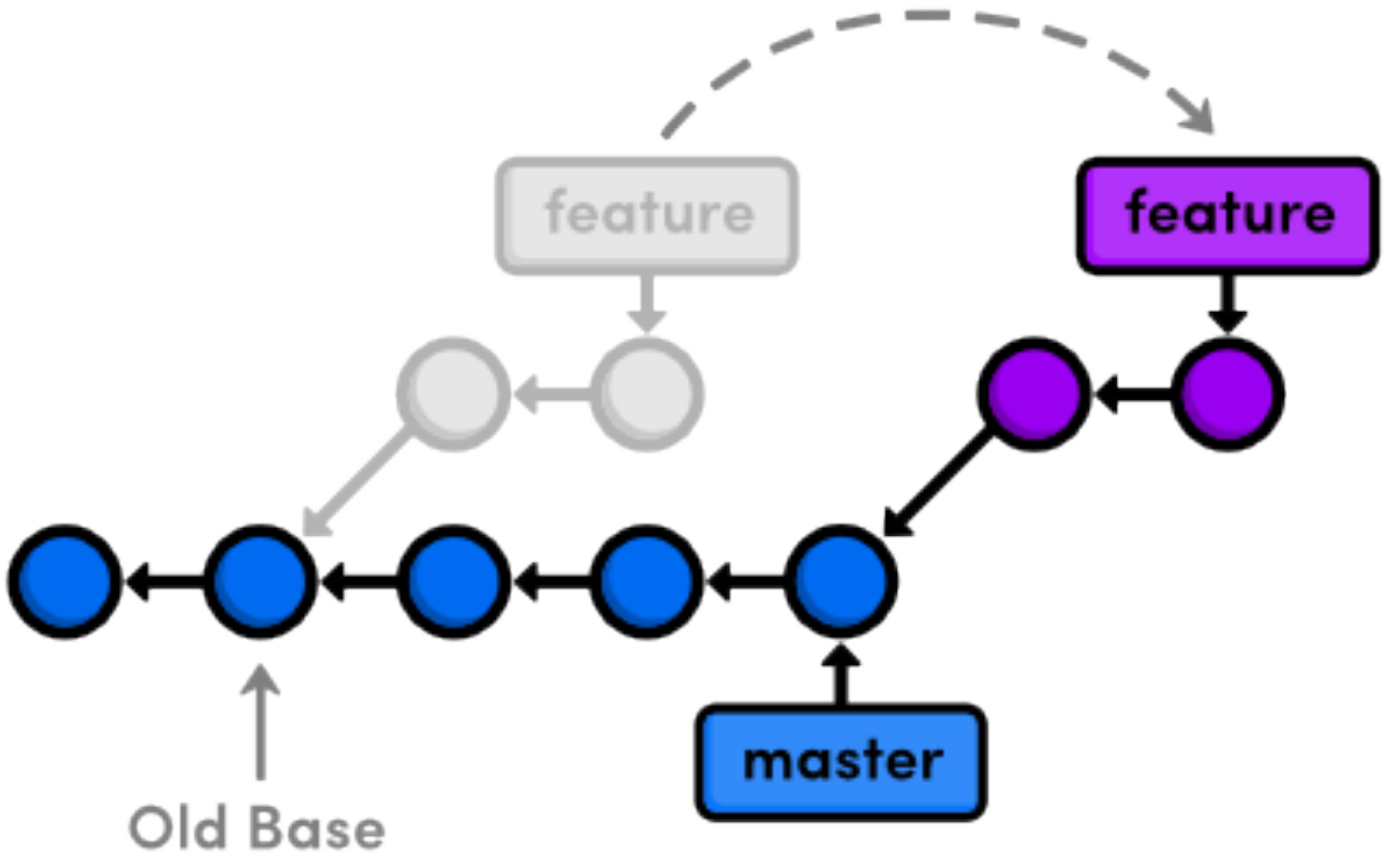
- **The power of public repos**
- **Share work-in-progress**
- **Test others' changes**
- **Merge / pull requests**
- **Combine multiple features under development**
- **Share our development histories as small changes - why did I do things this way?**
- **To contribute a change to somebody else, I first 'clone' their repo, make the change in my own version, and then ask them to merge in my change**

GIT WORKFLOWS: BRANCHES & MERGING



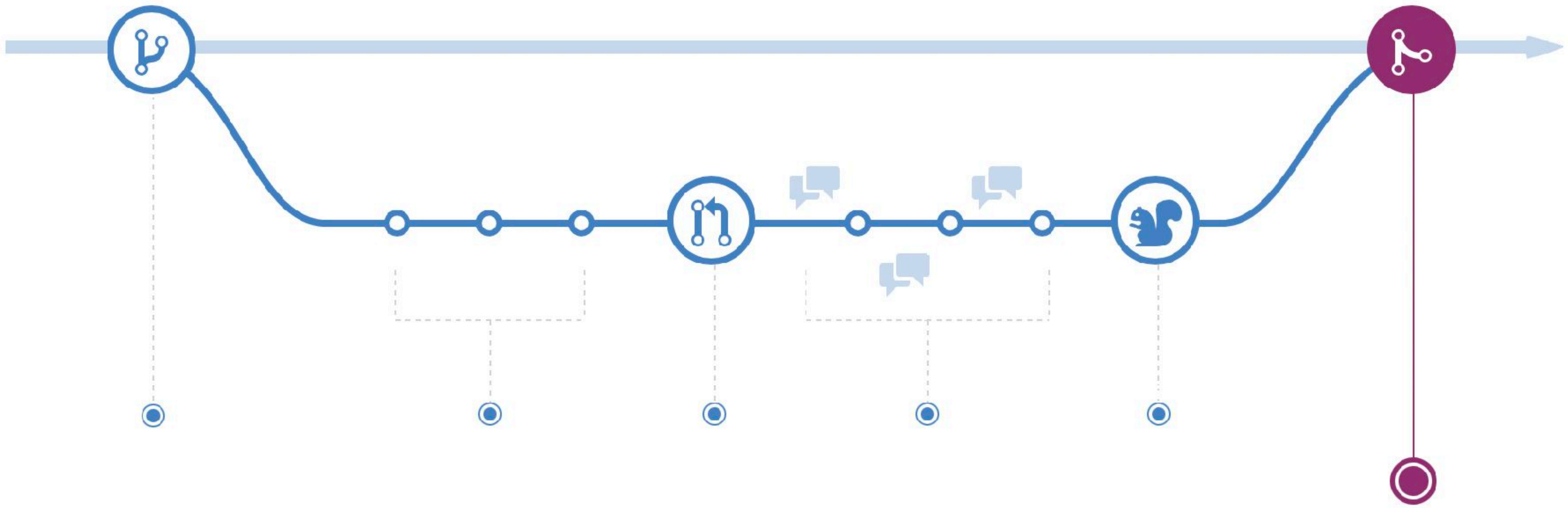
- Each feature is a new branch
- All your different versions of the HPC challenge!
- Common base is the scalar version
- Feature 1: MPI
- Feature 2: OpenMP
- Feature 3: OpenACC, CUDA, SYCL, etc.
- Imagine that these features have now been developed/improved over 3 months.
- Each feature branch works great, but major pains when you need to combine them & release

A BETTER (?) WORKFLOW: CONSTANT REBASING



- Think of feature commits as work-in-progress (e.g. on my laptop) that have not yet made it into our common master branch
- A large project like GROMACS can have hundreds of such work-in-progress commits; each of them is independent of all other feature commits
- When one feature commit is ready & merged into master, the other features should rebase to instead be a difference relative to the updated master state
- You can continue to work with the old base while developing, but before committing your feature it has to be rebased
- Advantage: Clean changes, rapid deployment

GitHub Flow:

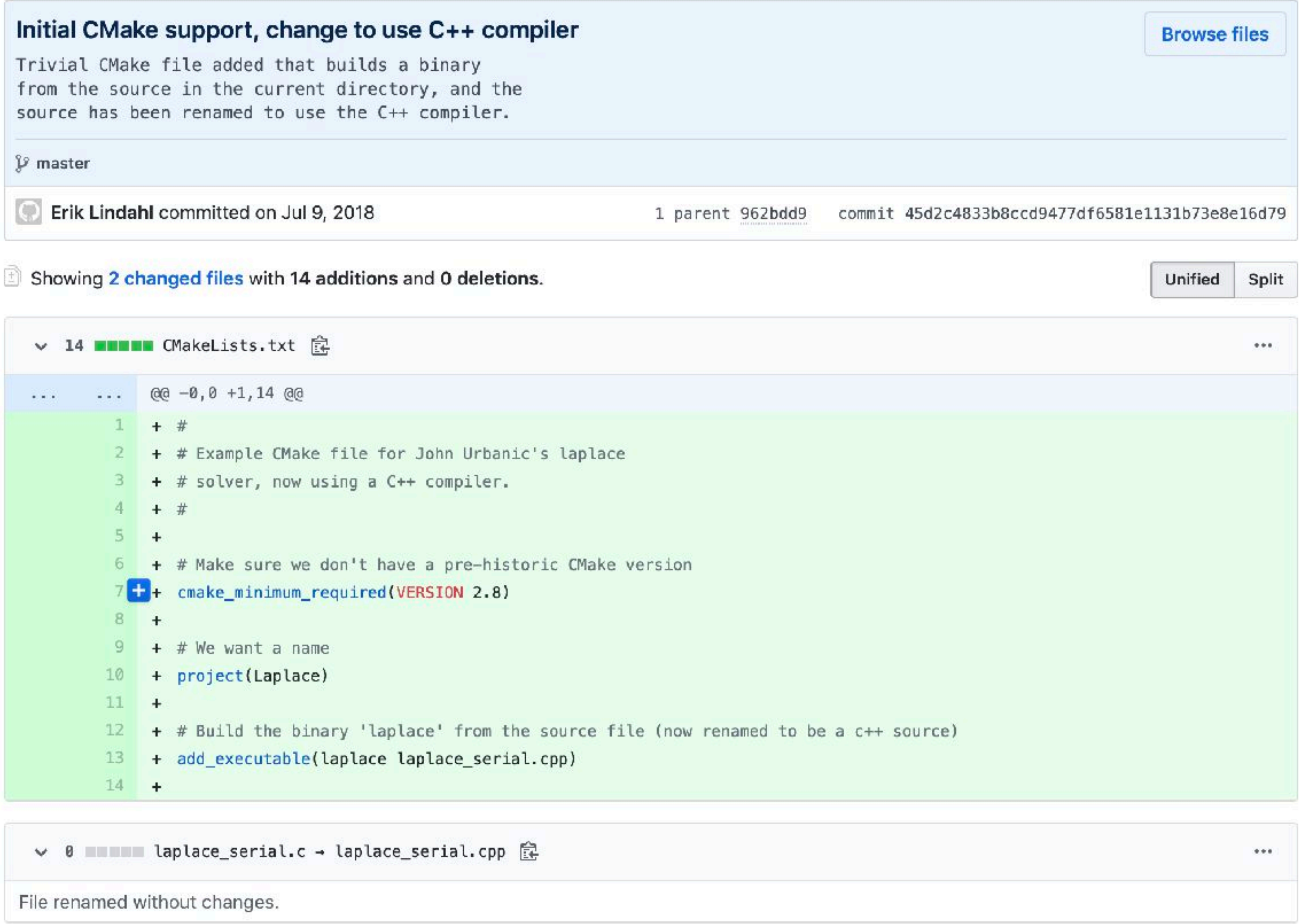


Merge

GOOD GIT COMMITS ARE...

- Small
  - 10-100 lines
  - NOT 1000 lines...
  - Decomposed as far as possible
- Limited to address a single issue
- Well documented
- Tested to work

This type of commit will also be close to trivial to rebase!



Initial CMake support, change to use C++ compiler

```
Trivial CMake file added that builds a binary
from the source in the current directory, and the
source has been renamed to use the C++ compiler.
```

Erik Lindahl committed on Jul 9, 2018 — 1 parent 962bdd9 — commit 45d2c4833b8ccd9477df6581e1131b73e8e16d79

Showing 2 changed files with 14 additions and 0 deletions.

14 CMakeLists.txt

```
@@ -0,0 +1,14 @@
+ #
+ # Example CMake file for John Urbanic's laplace
+ # solver, now using a C++ compiler.
+ #
+
+ # Make sure we don't have a pre-historic CMake version
+ cmake_minimum_required(VERSION 2.8)
+
+ # We want a name
+ project(Laplace)
+
+ # Build the binary 'laplace' from the source file (now renamed to be a c++ source)
+ add_executable(laplace laplace_serial.cpp)
+
```

0 laplace_serial.c → laplace_serial.cpp

File renamed without changes.

BUILD CHAINS

IS YOUR CODE PORTABLE? DOES IT COMPILE ON...

- Does your code compile on
- windows (MSVC)?
- PGI (Now NVIDIA) Compilers? Pathscale?
- Blue Gene?
- Fugaku (Both GCC and Fujitsu compilers)?
- ARM? AArch64? With the Arm compiler? Clang? Gcc?
- PowerPC (big endian)? OpenPower (little endian)?
- Google NativeClient?
- RISC V?

TYPICAL USER PROGRESSION

- Issue compiler commands manually
- Start using Makefiles, edit Makefiles, give up
- Automate the generation of Makefiles

```
CC=gcc
CFLAGS=-I.
DEPS = hellomake.h
OBJ = hellomake.o
hellofunc.o

%.o: %.c $(DEPS)
  $(CC) -c -o $@ $< $(CFLAGS)

hellomake: $(OBJ)
  $(CC) -o $@ $^ $(CFLAGS)
```

Think about edits you have to do just to compile trivial program in the summer school.

Friends don't let friends write makefiles.

CONFIGURING FOR THE REAL WORLD IS HORRIBLE - AND SO IS CMAKE(?)

- “Where is the X11 library? MKL? LibXML?”
- “Is this the buggy version 3.3.7 of the FFTW library?”
- “Is the Intel Math Kernel Library installed?”
- “Do we use that buggy gcc version?”
- “Does this compiler understand Xeon Phi AVX512?”
- “Which flags should be used to enable C++11 for this compiler?”
- “Is this a big or small endian system?”
- “Is a long integer 4 or 8 bytes on this host?”
- “How do we build a shared library here?”
- “How do we turn on OpenMP? OpenACC?”
- “What library should I link with to have gettimeofday() available?”
- “What C backend compiler is used with CUDA-8.0?”
- “What underscore naming standard does this Fortran compiler use?”
- “Is Doxygen available? Sphinx? Dot?”

Welcome to **CMake**, the cross-platform, open-source build system. CMake is a family of tools designed to build, test and package software. CMake is used to control the software compilation process using simple platform and compiler independent configuration files. CMake generates native makefiles and workspaces that can be used in the compiler environment of your choice.

CMake 2.8.8 Available

See what's new with the CMake 2.8.8 release

News

04.19.2012 CMake 2.8.8 is Now Available

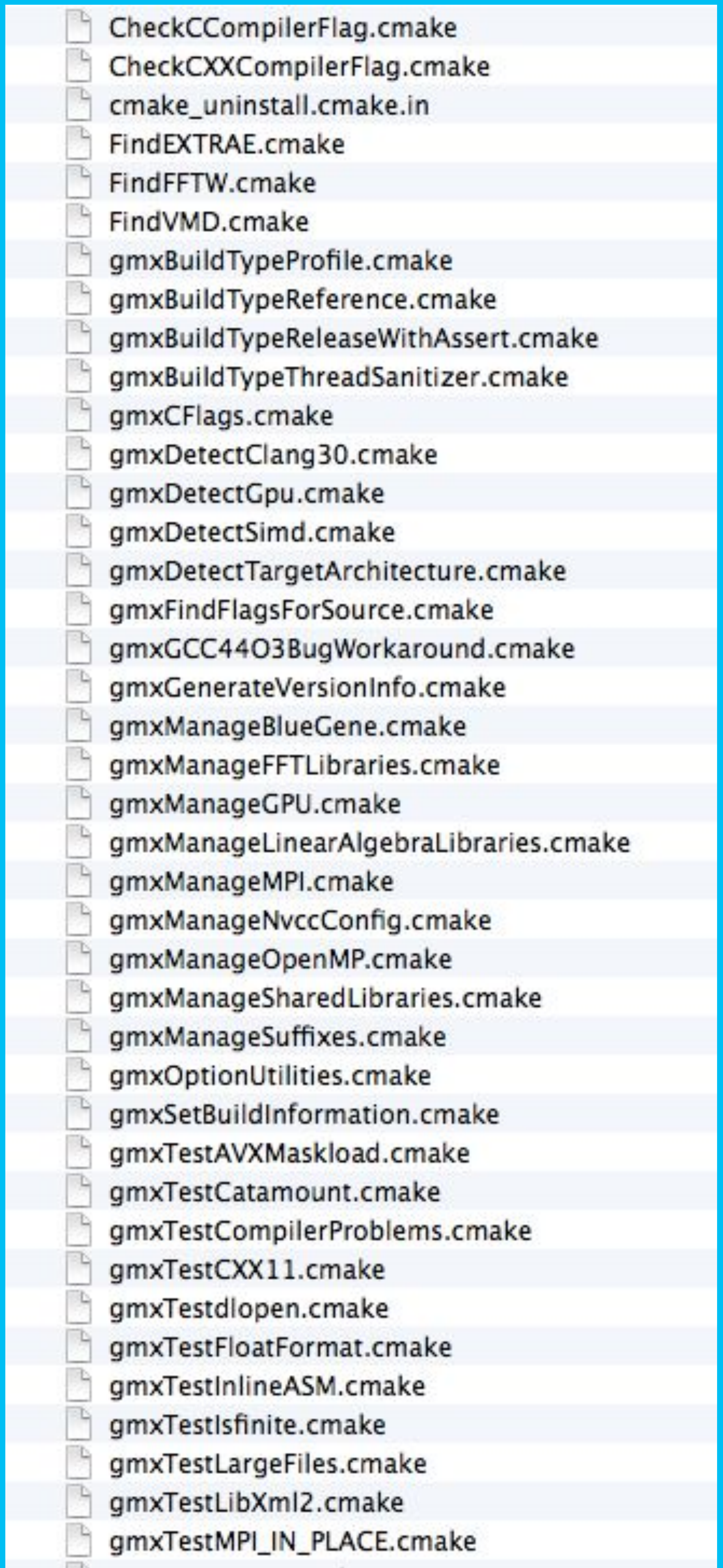
03.02.2012 CDash 2.0.2 Now Available

01.12.2012 Kitware Collaborates with NREL on Software Process for Radiance P...

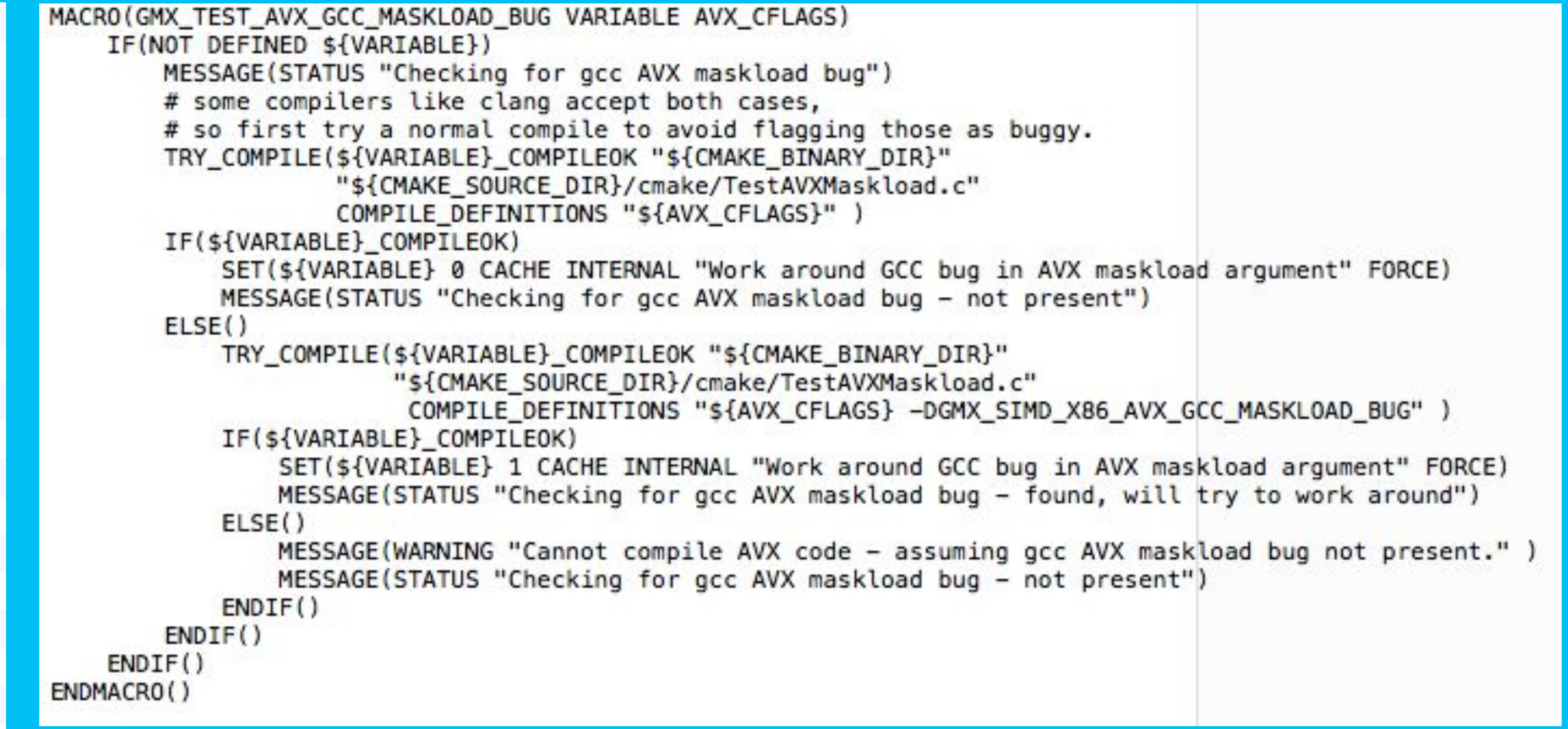
01.02.2012 CMake 2.8.7 Now Available

10.31.2011 Kitware Courses Move to Webinar Format

GROMACS has ~100 CMake tests for features/bugs/libraries/compilers



- CheckCCompilerFlag.cmake
- CheckCXXCompilerFlag.cmake
- cmake_uninstall.cmake.in
- FindEXTRAE.cmake
- FindFFTW.cmake
- FindVMD.cmake
- gmxBuildTypeProfile.cmake
- gmxBuildTypeReference.cmake
- gmxBuildTypeReleaseWithAssert.cmake
- gmxBuildTypeThreadSanitizer.cmake
- gmxCFlags.cmake
- gmxDetectClang30.cmake
- gmxDetectGpu.cmake
- gmxDetectSimd.cmake
- gmxDetectTargetArchitecture.cmake
- gmxFindFlagsForSource.cmake
- gmxGCC44O3BugWorkaround.cmake
- gmxGenerateVersionInfo.cmake
- gmxManageBlueGene.cmake
- gmxManageFFTLibraries.cmake
- gmxManageGPU.cmake
- gmxManageLinearAlgebraLibraries.cmake
- gmxManageMPI.cmake
- gmxManageNvccConfig.cmake
- gmxManageOpenMP.cmake
- gmxManageSharedLibraries.cmake
- gmxManageSuffixes.cmake
- gmxOptionUtilities.cmake
- gmxSetBuildInformation.cmake
- gmxTestAVXMaskload.cmake
- gmxTestCatamount.cmake
- gmxTestCompilerProblems.cmake
- gmxTestCXX11.cmake
- gmxTestdlopen.cmake
- gmxTestFloatFormat.cmake
- gmxTestInlineASM.cmake
- gmxTestIsfinite.cmake
- gmxTestLargeFiles.cmake
- gmxTestLibXml2.cmake
- gmxTestMPI_IN_PLACE.cmake



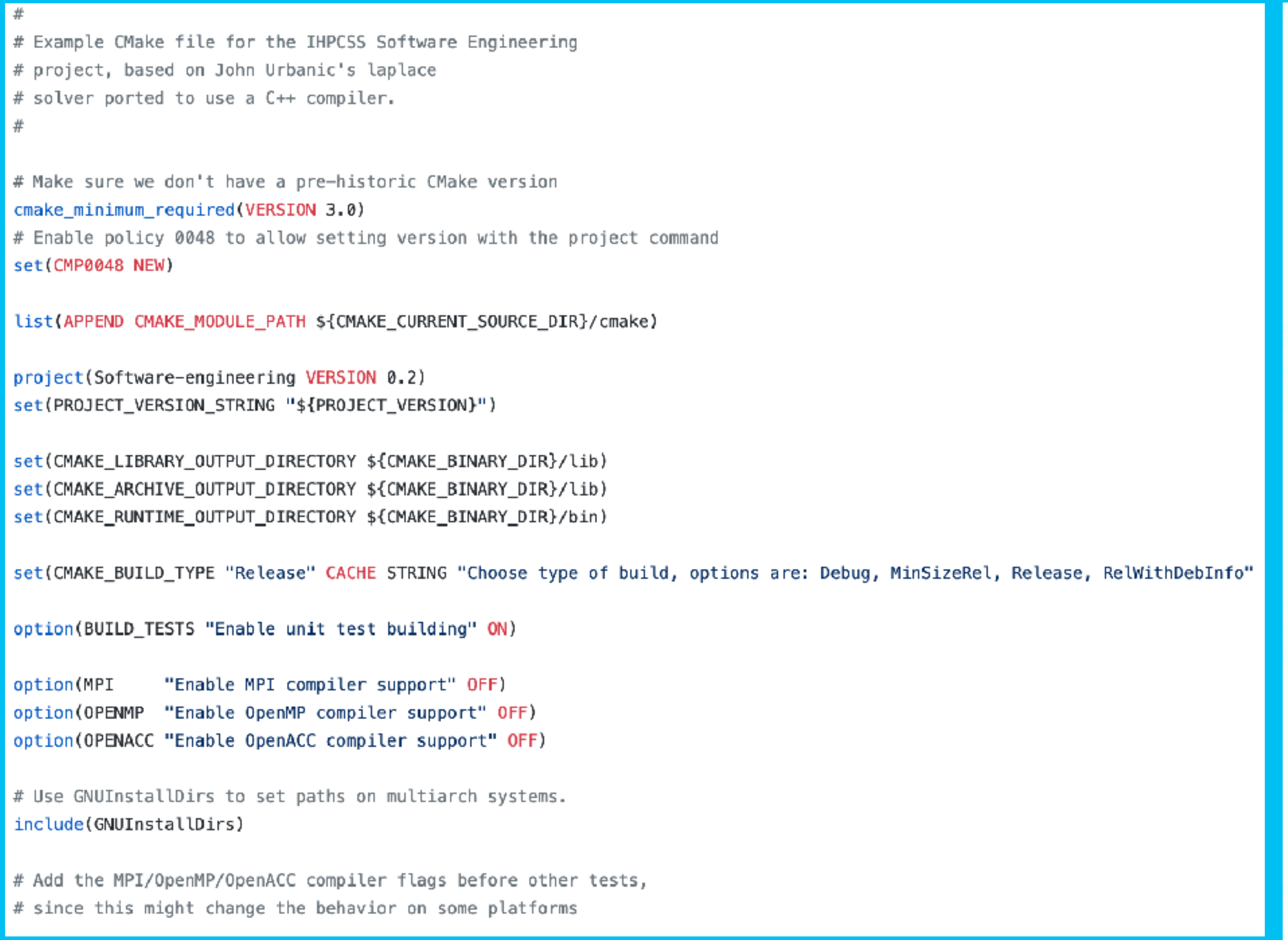
```
MACRO(GMX_TEST_AVX_GCC_MASKLOAD_BUG VARIABLE AVX_CFLAGS)
    IF(NOT DEFINED ${VARIABLE})
        MESSAGE(STATUS "Checking for gcc AVX maskload bug")
        # some compilers like clang accept both cases,
        # so first try a normal compile to avoid flagging those as buggy.
        TRY_COMPILE(${VARIABLE}_COMPILEOK "${CMAKE_BINARY_DIR}"
                    "${CMAKE_SOURCE_DIR}/cmake/TestAVXMaskload.c"
                    COMPILE_DEFINITIONS "${AVX_CFLAGS}" )
        IF(${VARIABLE}_COMPILEOK)
            SET(${VARIABLE} 0 CACHE INTERNAL "Work around GCC bug in AVX maskload argument" FORCE)
            MESSAGE(STATUS "Checking for gcc AVX maskload bug - not present")
        ELSE()
            TRY_COMPILE(${VARIABLE}_COMPILEOK "${CMAKE_BINARY_DIR}"
                        "${CMAKE_SOURCE_DIR}/cmake/TestAVXMaskload.c"
                         COMPILE_DEFINITIONS "${AVX_CFLAGS} -DGMX_SIMD_X86_AVX_GCC_MASKLOAD_BUG" )
            IF(${VARIABLE}_COMPILEOK)
                SET(${VARIABLE} 1 CACHE INTERNAL "Work around GCC bug in AVX maskload argument" FORCE)
                MESSAGE(STATUS "Checking for gcc AVX maskload bug - found, will try to work around")
            ELSE()
                MESSAGE(WARNING "Cannot compile AVX code - assuming gcc AVX maskload bug not present." )
                MESSAGE(STATUS "Checking for gcc AVX maskload bug - not present")
            ENDIF()
        ENDIF()
    ENDIF()
ENDMACRO()
```

Optional components (FFT libs) and extensive regressiontests can be downloaded automatically

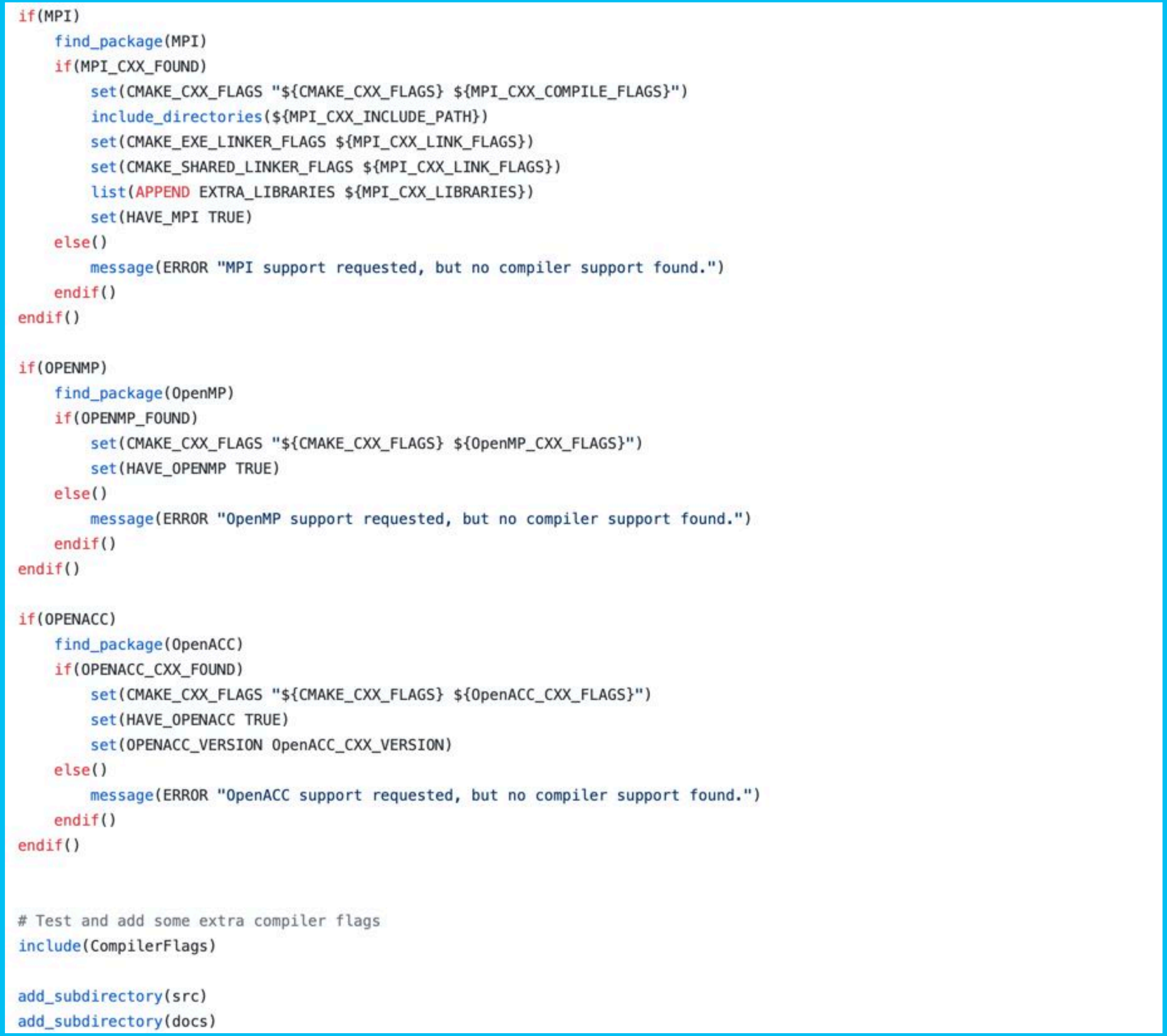
Generators: Makefiles, Eclipse, Xcode, VisualStudio, nmake, CodeBlocks, KDevelop3, etc.

But don’t start with GROMACS: Look at the CMakeLists.txt in the IHPCSS/software-engineering example: 75 lines and a few modules for complete detection of compilers, OpenMP, OpenACC, MPI, and everything else you’ll see on the next few slides!

The complete CMakeLists.txt source for the IHPCSS Laplace code

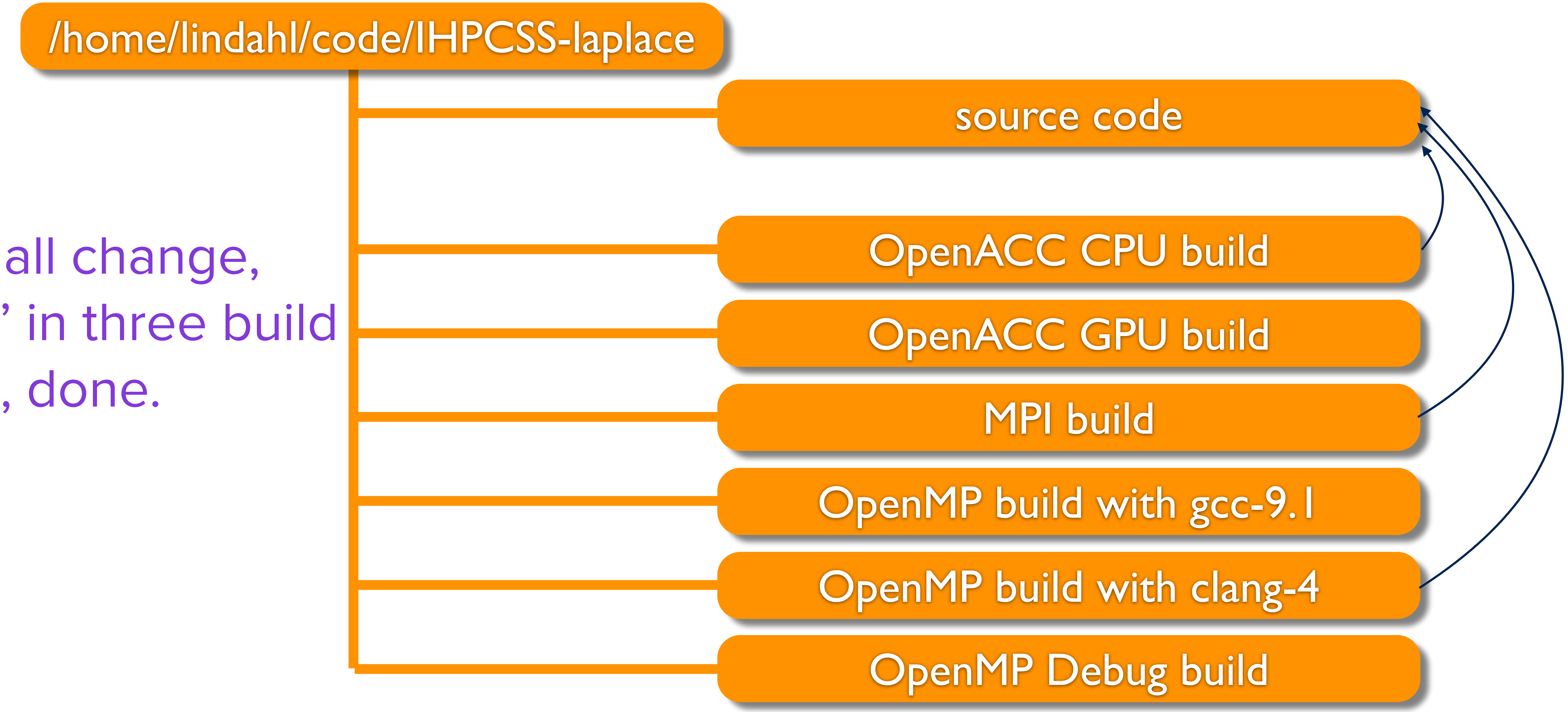


```
#
# Example CMake file for the IHPCSS Software Engineering
# project, based on John Urbanic's laplace
# solver ported to use a C++ compiler.
#

# Make sure we don't have a pre-historic CMake version
cmake_minimum_required(VERSION 3.0)
# Enable policy 0048 to allow setting version with the project command
set(CMP0048 NEW)

list(APPEND CMAKE_MODULE_PATH ${CMAKE_CURRENT_SOURCE_DIR}/cmake)

project(Software-engineering VERSION 0.2)
set(PROJECT_VERSION_STRING "${PROJECT_VERSION}")

set(CMAKE_LIBRARY_OUTPUT_DIRECTORY ${CMAKE_BINARY_DIR}/lib)
set(CMAKE_ARCHIVE_OUTPUT_DIRECTORY ${CMAKE_BINARY_DIR}/lib)
set(CMAKE_RUNTIME_OUTPUT_DIRECTORY ${CMAKE_BINARY_DIR}/bin)

set(CMAKE_BUILD_TYPE "Release" CACHE STRING "Choose type of build, options are: Debug, MinSizeRel, Release, RelWithDebInfo"

option(BUILD_TESTS "Enable unit test building" ON)

option(MPI     "Enable MPI compiler support" OFF)
option(OPENMP  "Enable OpenMP compiler support" OFF)
option(OPENACC "Enable OpenACC compiler support" OFF)

# Use GNUInstallDirs to set paths on multiarch systems.
include(GNUInstallDirs)

# Add the MPI/OpenMP/OpenACC compiler flags before other tests,
# since this might change the behavior on some platforms
```



```
if(MPI)
    find_package(MPI)
    if(MPI_CXX_FOUND)
        set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} ${MPI_CXX_COMPILE_FLAGS}")
        include_directories(${MPI_CXX_INCLUDE_PATH})
        set(CMAKE_EXE_LINKER_FLAGS ${MPI_CXX_LINK_FLAGS})
        set(CMAKE_SHARED_LINKER_FLAGS ${MPI_CXX_LINK_FLAGS})
        list(APPEND EXTRA_LIBRARIES ${MPI_CXX_LIBRARIES})
        set(HAVE_MPI TRUE)
    else()
        message(ERROR "MPI support requested, but no compiler support found.")
    endif()
endif()

if(OPENMP)
    find_package(OpenMP)
    if(OPENMP_FOUND)
        set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} ${OpenMP_CXX_FLAGS}")
        set(HAVE_OPENMP TRUE)
    else()
        message(ERROR "OpenMP support requested, but no compiler support found.")
    endif()
endif()

if(OPENACC)
    find_package(OpenACC)
    if(OPENACC_CXX_FOUND)
        set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} ${OpenACC_CXX_FLAGS}")
        set(HAVE_OPENACC TRUE)
        set(OPENACC_VERSION OpenACC_CXX_VERSION)
    else()
        message(ERROR "OpenACC support requested, but no compiler support found.")
    endif()
endif()


# Test and add some extra compiler flags
include(CompilerFlags)

add_subdirectory(src)
add_subdirectory(docs)
```

MAKE A HABIT OF USING OUT-OF-SOURCE BUILDS


/home/lindahl/code/IHPCSS-laplace
source code
OpenACC CPU build
OpenACC GPU build
MPI build
OpenMP build with gcc-9.1
OpenMP build with clang-4
OpenMP Debug build

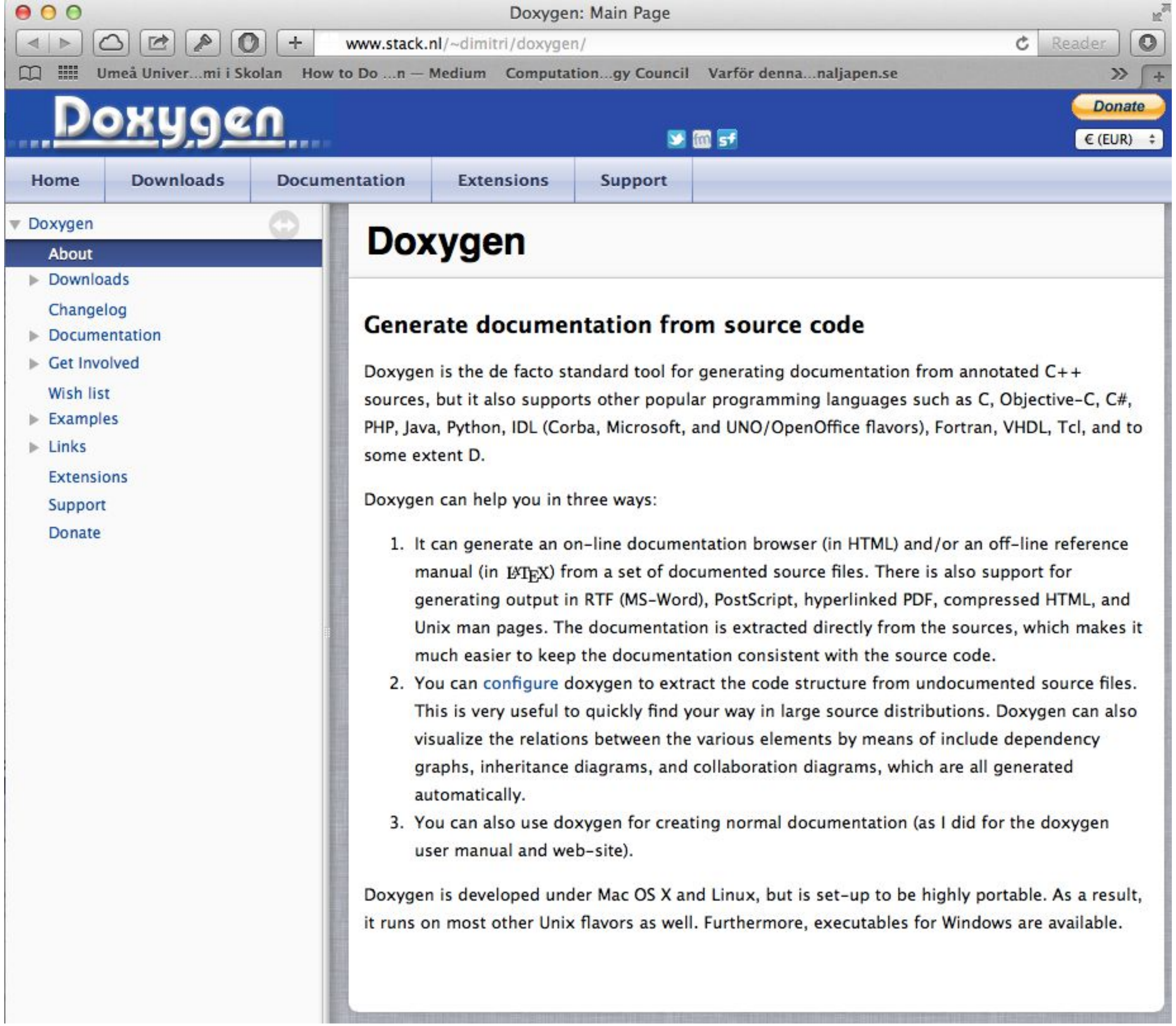
Make a small change, run “make” in three build directories, done.

```
$ ~> mkdir build-openacc
$ ~> cd build-openacc
$ build-openacc> cmake -DOPENACC=ON ../path/to/source/directory
```

LIVING WITH YOUR CODE FOR YEARS (OR DECADES):

DOCUMENTATION

IF DOCUMENTATION ISN'T IN THE SOURCE, IT WON'T BE UPDATED



Doxygen

Generate documentation from source code

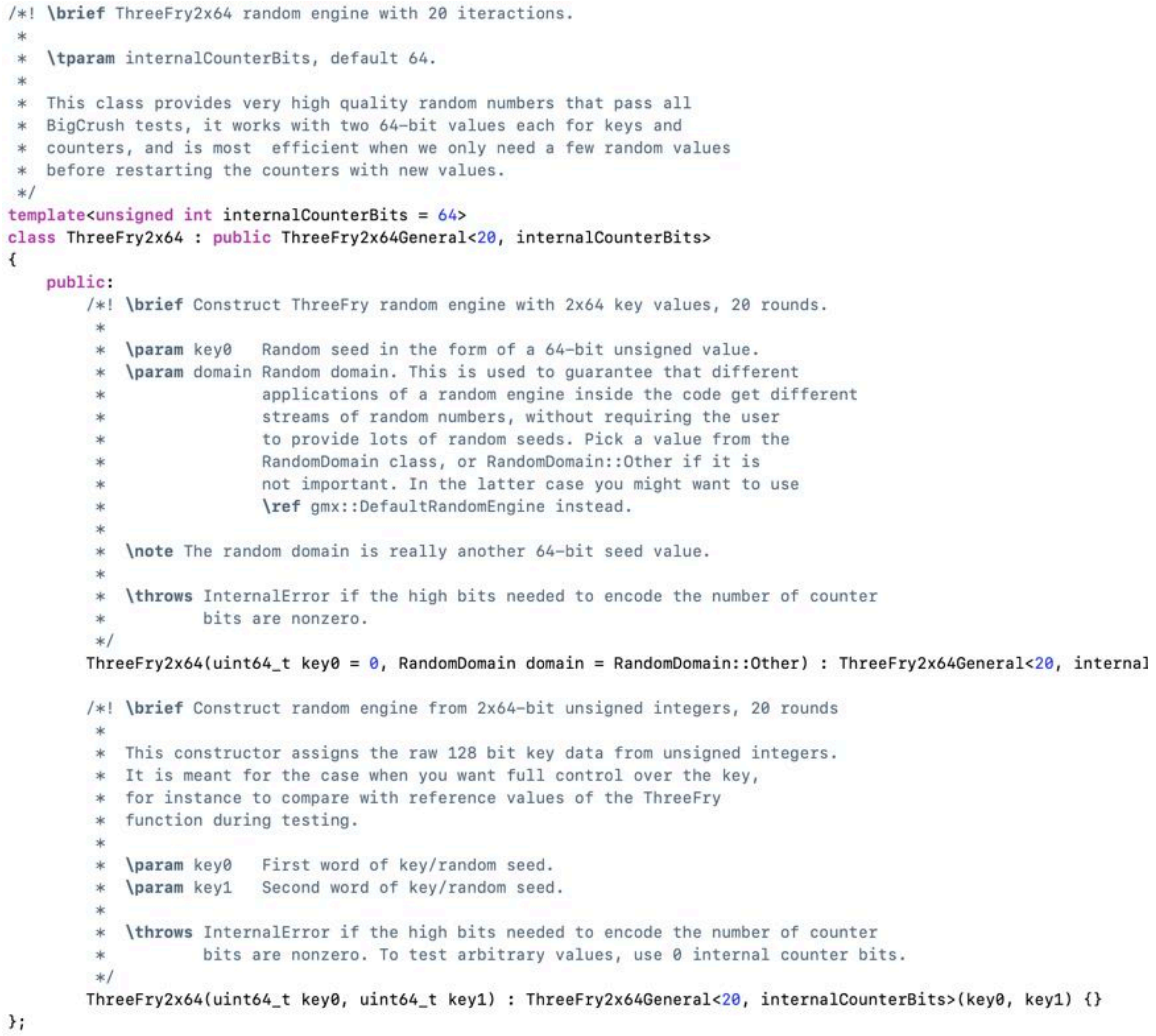
Doxygen is the de facto standard tool for generating documentation from annotated C++ sources, but it also supports other popular programming languages such as C, Objective-C, C#, PHP, Java, Python, IDL (Corba, Microsoft, and UNO/OpenOffice flavors), Fortran, VHDL, Tcl, and to some extent D.

Doxygen can help you in three ways:

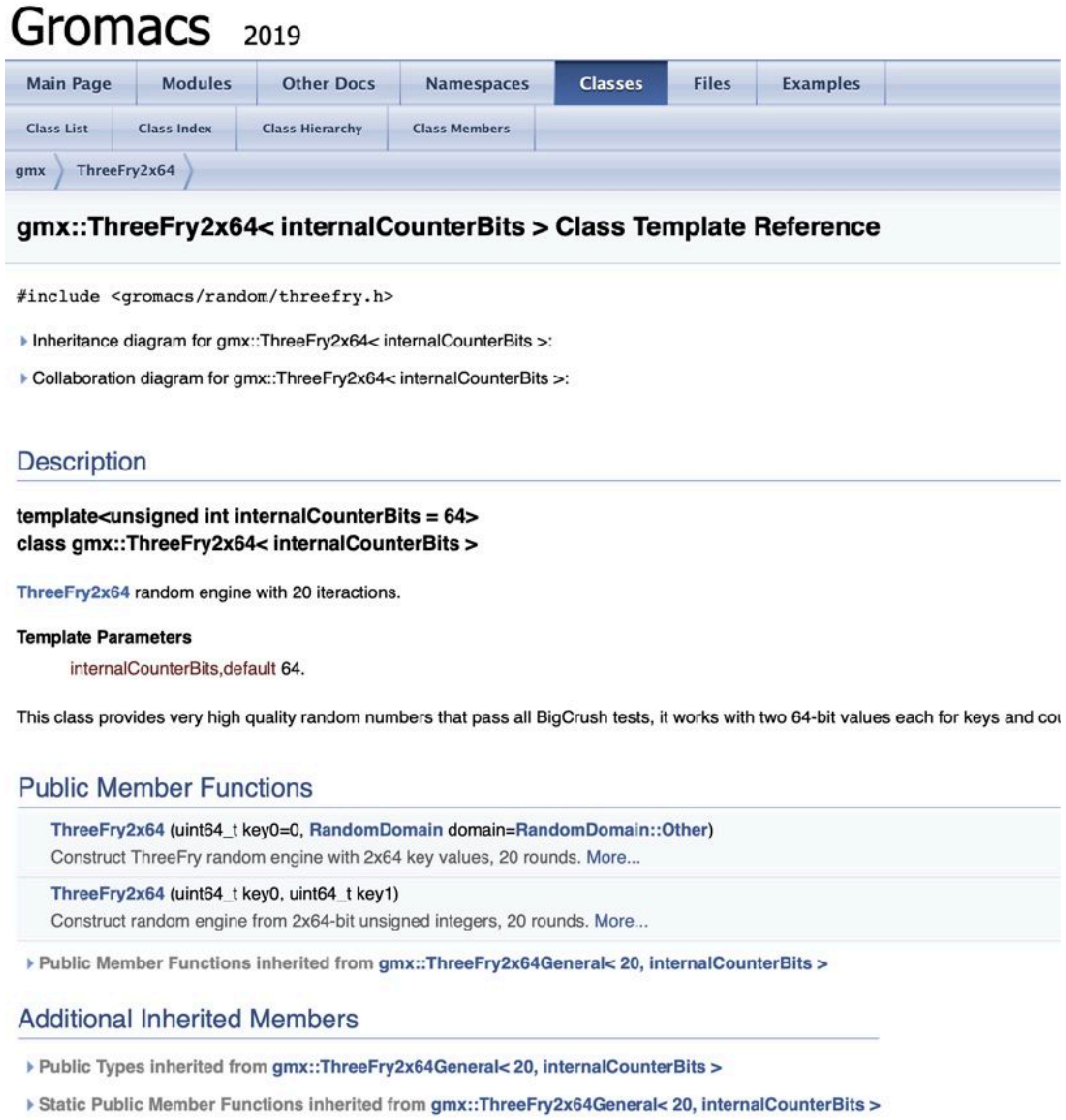
1. It can generate an on-line documentation browser (in HTML) and/or an off-line reference manual (in LaTeX) from a set of documented source files. There is also support for generating output in RTF (MS-Word), PostScript, hyperlinked PDF, compressed HTML, and Unix man pages. The documentation is extracted directly from the sources, which makes it much easier to keep the documentation consistent with the source code.
2. You can configure doxygen to extract the code structure from undocumented source files. This is very useful to quickly find your way in large source distributions. Doxygen can also visualize the relations between the various elements by means of include dependency graphs, inheritance diagrams, and collaboration diagrams, which are all generated automatically.
3. You can also use doxygen for creating normal documentation (as I did for the doxygen user manual and web-site).

Doxygen is developed under Mac OS X and Linux, but is set-up to be highly portable. As a result, it runs on most other Unix flavors as well. Furthermore, executables for Windows are available.

DOXYGEN EXAMPLE - THE GROMACS 'RANDOM' MODULE



```
/*! \brief ThreeFry2x64 random engine with 20 iteractions.
 *
 *  \tparam internalCounterBits, default 64.
 *
 *  This class provides very high quality random numbers that pass all
 *  BigCrush tests, it works with two 64-bit values each for keys and
 *  counters, and is most  efficient when we only need a few random values
 *  before restarting the counters with new values.
 */
template<unsigned int internalCounterBits = 64>
class ThreeFry2x64 : public ThreeFry2x64General<20, internalCounterBits>
{
    public:
        /*! \brief Construct ThreeFry random engine with 2x64 key values, 20 rounds.
         *
         *  \param key0   Random seed in the form of a 64-bit unsigned value.
         *  \param domain Random domain. This is used to guarantee that different
         *                applications of a random engine inside the code get different
         *                streams of random numbers, without requiring the user
         *                to provide lots of random seeds. Pick a value from the
         *                RandomDomain class, or RandomDomain::Other if it is
         *                not important. In the latter case you might want to use
         *                \ref gmx::DefaultRandomEngine instead.
         *
         *  \note The random domain is really another 64-bit seed value.
         *
         *  \throws InternalError if the high bits needed to encode the number of counter
         *          bits are nonzero.
         */
        ThreeFry2x64(uint64_t key0 = 0, RandomDomain domain = RandomDomain::Other) : ThreeFry2x64General<20, internal

        /*! \brief Construct random engine from 2x64-bit unsigned integers, 20 rounds
         *
         *  This constructor assigns the raw 128 bit key data from unsigned integers.
         *  It is meant for the case when you want full control over the key,
         *  for instance to compare with reference values of the ThreeFry
         *  function during testing.
         *
         *  \param key0   First word of key/random seed.
         *  \param key1   Second word of key/random seed.
         *
         *  \throws InternalError if the high bits needed to encode the number of counter
         *          bits are nonzero. To test arbitrary values, use 0 internal counter bits.
         */
        ThreeFry2x64(uint64_t key0, uint64_t key1) : ThreeFry2x64General<20, internalCounterBits>(key0, key1) {}
};
```



gmx::ThreeFry2x64< internalCounterBits > Class Template Reference

`#include <gromacs/random/threefry.h>`

▸ Inheritance diagram for gmx::ThreeFry2x64< internalCounterBits >:

▸ Collaboration diagram for gmx::ThreeFry2x64< internalCounterBits >:

Description

template<unsigned int internalCounterBits = 64>
class gmx::ThreeFry2x64< internalCounterBits >

ThreeFry2x64 random engine with 20 iteractions.

Template Parameters

internalCounterBits,default 64.

This class provides very high quality random numbers that pass all BigCrush tests, it works with two 64-bit values each for keys and cou

Public Member Functions

ThreeFry2x64 (uint64_t key0=0, RandomDomain domain=RandomDomain::Other)
Construct ThreeFry random engine with 2x64 key values, 20 rounds. More...

ThreeFry2x64 (uint64_t key0, uint64_t key1)
Construct random engine from 2x64-bit unsigned integers, 20 rounds. More...

▸ Public Member Functions inherited from gmx::ThreeFry2x64General< 20, internalCounterBits >

Additional Inherited Members

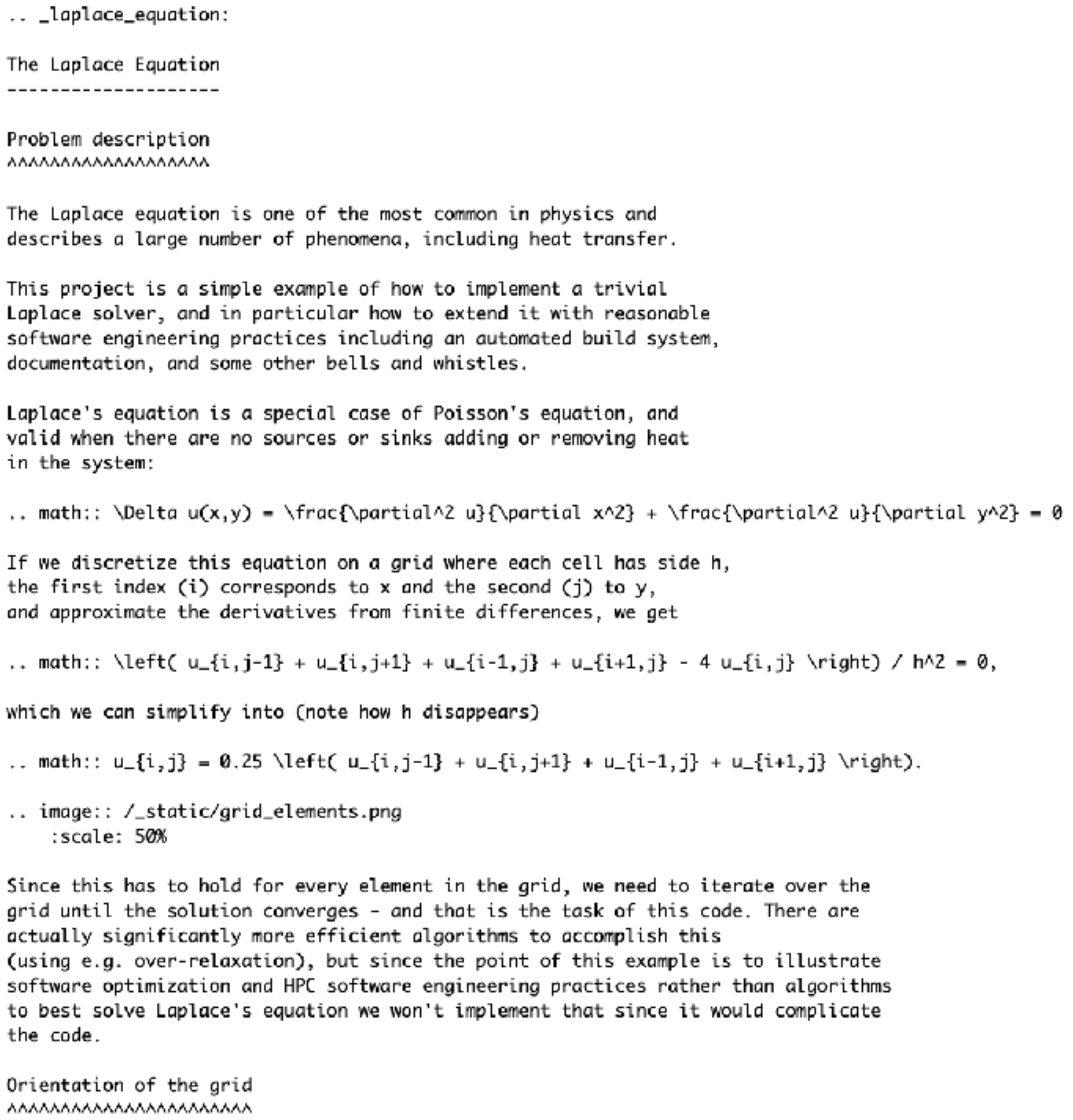
▸ Public Types inherited from gmx::ThreeFry2x64General< 20, internalCounterBits >

▸ Static Public Member Functions inherited from gmx::ThreeFry2x64General< 20, internalCounterBits >

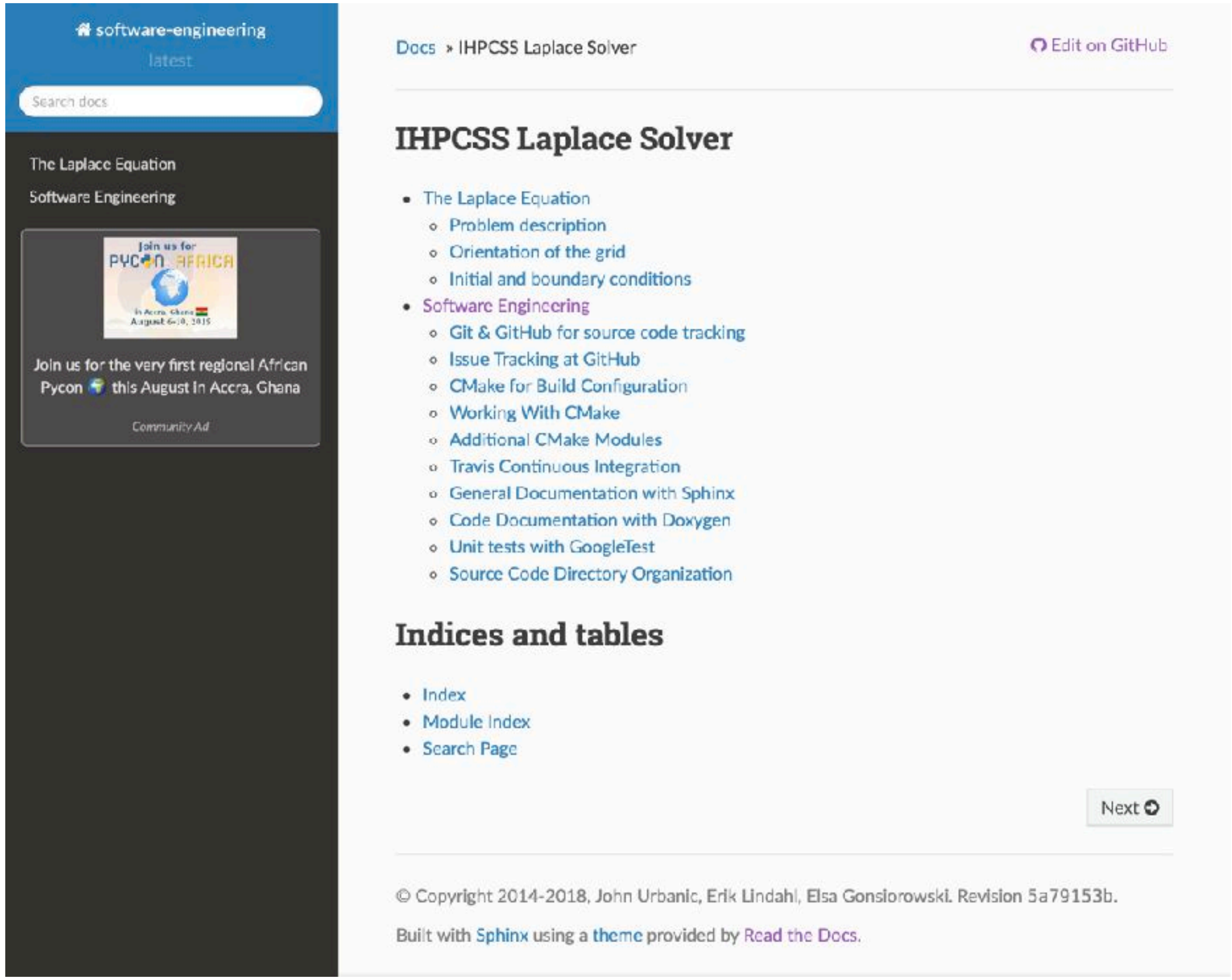
API - APPLICATION PROGRAMMING INTERFACES

- Each module/part should have a clear Application Programming Interface
- An API is a long-term promise to the user/other developers - it cannot change randomly
- Document the API from the user's point-of-view, not the implementer's!
- What input is valid? Can the user expect a certain algorithm or not?
- Always fully separate interfaces from implementations
- With a well-defined interface, you should be able to change the implementation without the interface changing - you should never have to look at my implementation
- Interfaces need *extensive* documentation - implementations can get away with less

HIGH-LEVEL NON-SOURCE DOCUMENTATION: SPHINX



```
.. _laplace_equation:

The Laplace Equation
--------------------

Problem description
^^^^^^^^^^^^^^^^^^^

The Laplace equation is one of the most common in physics and
describes a large number of phenomena, including heat transfer.

This project is a simple example of how to implement a trivial
Laplace solver, and in particular how to extend it with reasonable
software engineering practices including an automated build system,
documentation, and some other bells and whistles.

Laplace's equation is a special case of Poisson's equation, and
valid when there are no sources or sinks adding or removing heat
in the system:

.. math:: \Delta u(x,y) = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0

If we discretize this equation on a grid where each cell has side h,
the first index (i) corresponds to x and the second (j) to y,
and approximate the derivatives from finite differences, we get

.. math:: \left( u_{i,j-1} + u_{i,j+1} + u_{i-1,j} + u_{i+1,j} - 4 u_{i,j} \right) / h^2 = 0,

which we can simplify into (note how h disappears)

.. math:: u_{i,j} = 0.25 \left( u_{i,j-1} + u_{i,j+1} + u_{i-1,j} + u_{i+1,j} \right).

.. image:: /_static/grid_elements.png
   :scale: 50%

Since this has to hold for every element in the grid, we need to iterate over the
grid until the solution converges - and that is the task of this code. There are
actually significantly more efficient algorithms to accomplish this
(using e.g. over-relaxation), but since the point of this example is to illustrate
software optimization and HPC software engineering practices rather than algorithms
to best solve Laplace's equation we won't implement that since it would complicate
the code.

Orientation of the grid
^^^^^^^^^^^^^^^^^^^^^^^
```



IHPCSS Laplace Solver

- The Laplace Equation
  - Problem description
  - Orientation of the grid
  - Initial and boundary conditions
- Software Engineering
  - Git & GitHub for source code tracking
  - Issue Tracking at GitHub
  - CMake for Build Configuration
  - Working With CMake
  - Additional CMake Modules
  - Travis Continuous Integration
  - General Documentation with Sphinx
  - Code Documentation with Doxygen
  - Unit tests with GoogleTest
  - Source Code Directory Organization

Indices and tables

- Index
- Module Index
- Search Page

© Copyright 2014-2018, John Urbanic, Erik Lindahl, Elsa Gonsiorowski. Revision 5a79153b.

Built with Sphinx using a theme provided by Read the Docs.

Fully integrated into IHPCSS-laplace.
Check out the docs folder, and if you have sphinx/latex installed you can type "make sphinx-html" or "make sphinx-pdf".

Integrated it with readthedocs.org!
Any time a new change is pushed to the repo, documentation is built automatically at http://software-engineering.readthedocs.org

LANGUAGES

THE CASE FOR MODERN C++

- Modern: Threads, atomics, etc. part of C++11
- Very powerful library with containers, algorithms
- Strongly typed language
- Still a low-level language - you control data exactly
- Modern C++ has gotten rid of pointers, memory errors
- Templates avoid code duplication
- Some very advanced parallelization libraries: Intel TBB
- Rapidly developing language, large ISO committee
- Parallel Standard Template Library (STL) in C++17/20
- A lot of momentum from vendors and large-scale projects (e.g. ECP)
- Negative: It is a VERY complex language to master

ALL LANGUAGES ARE BAD, BUT SOME ARE USEFUL

- “REAL PROGRAMMERS CAN WRITE FORTRAN IN ANY LANGUAGE”
- "C combines the flexibility and power of assembly language with the user-friendliness of assembly language."
- “C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do it blows your whole leg off.”
- The actual C++ nightmare: You accidentally create a dozen instances of yourself and shoot them all in the foot. Providing emergency medical care is impossible since you can't tell which are bitwise copies and which are just pointing at others and saying, "That's me over there."

C++ Core guidelines (Herb Sutter & Bjarne Stroustrup):
https://github.com/isocpp/CppCoreGuidelines/blob/master/CppCoreGuidelines.md

Example: If you have ever worked with mutex:es to make sure only one thread accesses a critical region, you have likely bumped into race conditions or deadlocks e.g. when you forget to release a mutex in complex code.

These errors are insanely difficult to debug, since it depends in dynamic timing events - when you run it in the debugger there won't be any error!

Definition:

```
class Lock {
public:
    explicit Lock(Mutex *pm)
    : mutexPtr(pm)
    { lock(*mutexPtr); }

    ~Lock() { unlock(*mutexPtr) };

private:
    Mutex *mutexPtr;
}
```

Usage in client code:

```
Mutex m;

…

{
    Lock ml(&m);
    …
}
```

One more problem: What happens if you copy that class? Then the first object to go out of scope will release the mutex, while the second thinks it's still locked (=bad)!

Easy to fix in C++11: Just use a reference-counted shared pointer.
Note: no change to the client code.

Definition:

```
class Lock {
public:
    explicit Lock(Mutex *pm)
    : mutexPtr(pm, unlock)
    { lock(mutexPtr.get()); }

    ~~~Lock() { unlock(*mutexPtr) };~~

private:
    std::shared_ptr<Mutex> mutexPtr;
}
```

Usage in client code:

```
Mutex m;

…

{
    Lock ml(&m);
    …
}
```

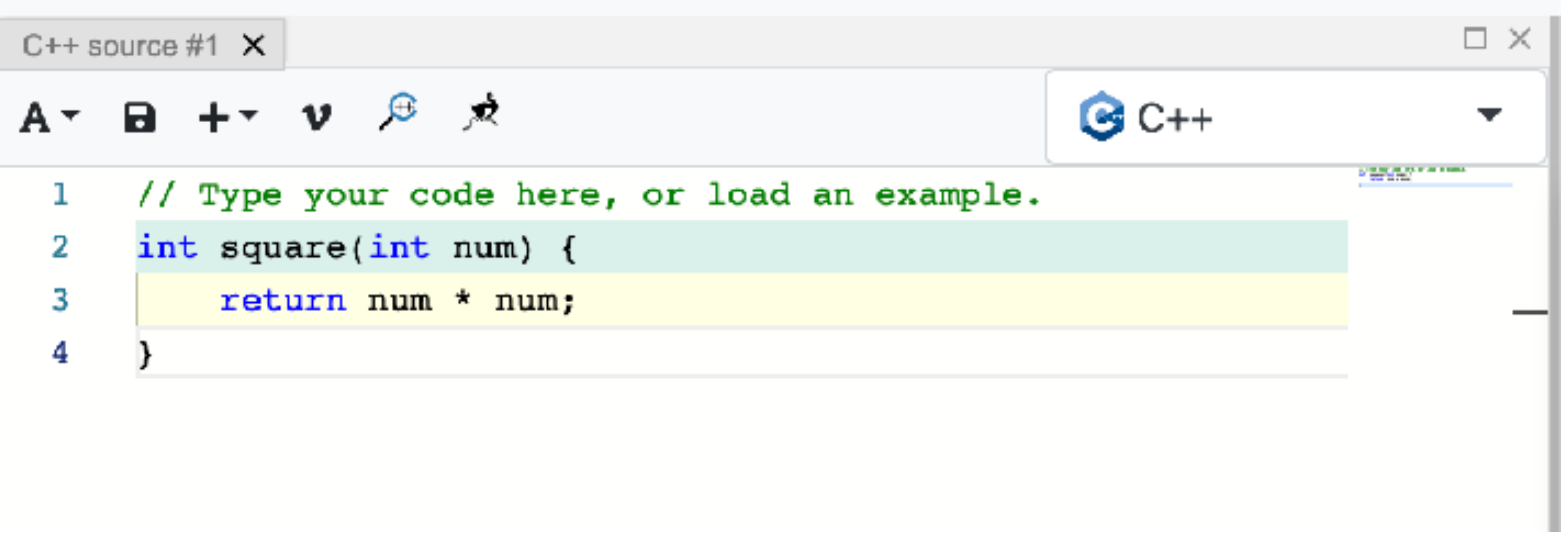
TEMPLATE METAPROGRAMMING:- C++ CAN BE MUCH FASTER THAN C OR FORTRAN

```
int
myFunc(obj_t obj, int choiceA, int choice B)
{
    for(int i=0;i<obj.N;i++)
    {
        if(choiceA==1)
        {
            if(choiceB==1)
            {
                kernelcode1;
            }
            else if(choiceB==2)
            {
                kernelcode2;
            }
        }
        else if(choiceA==2)
        {
            if(choiceB==1)
            {
                kernelcode3;
            }
            else if(choiceB==2)
            {
                kernelcode4;
            }
        }
    }
}

calling code in different translation unit:

myFunc(obj,2,3);
```

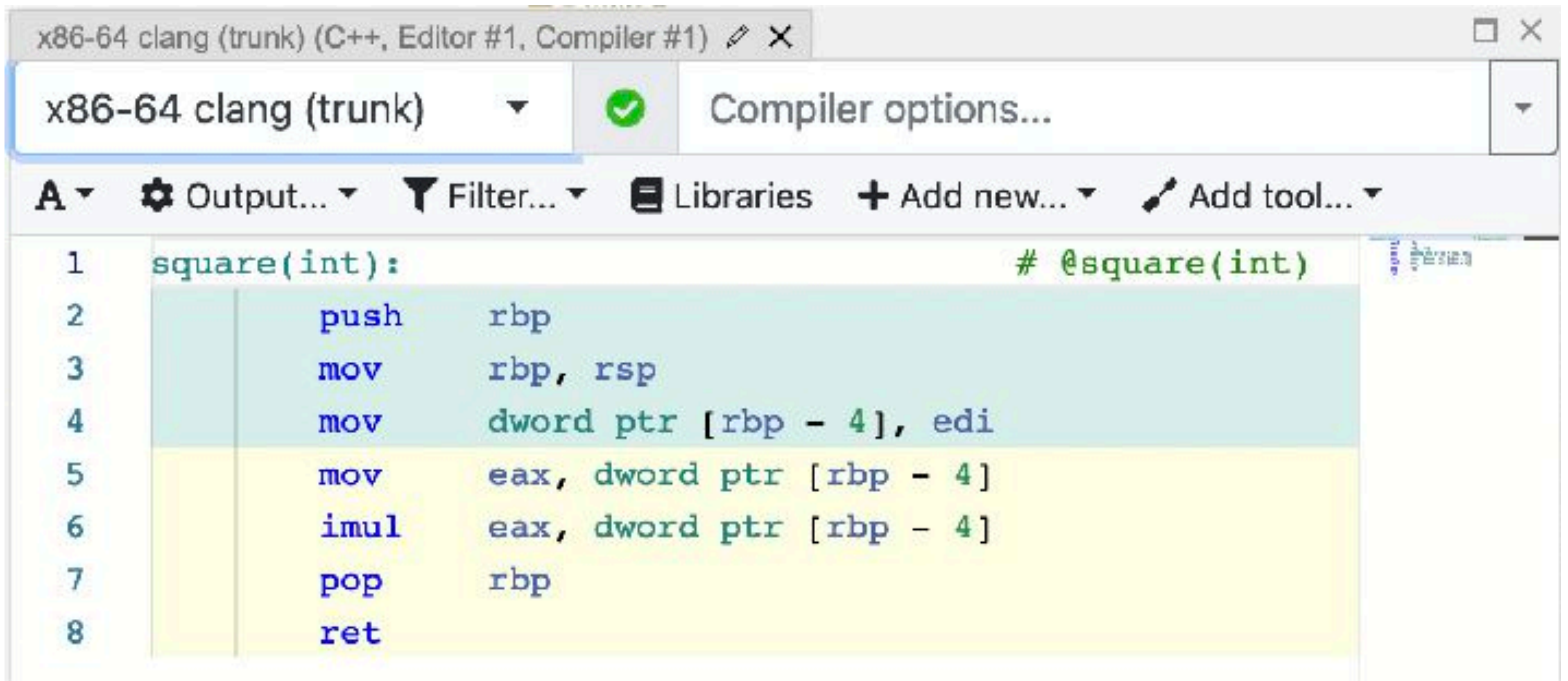
```
template <int choiceA, int choice B>
int
myFunc(obj_t obj)
{
    for(int i=0;i<obj.N;i++)
    {
        if(choiceA==1)
        {
            if(choiceB==1)
            {
                kernelcode1;
            }
            else if(choiceB==2)
            {
                kernelcode2;
            }
        }
        else if(choiceA==2)
        {
            if(choiceB==1)
            {
                kernelcode3;
            }
            else if(choiceB==2)
            {
                kernelcode4;
            }
        }
    }
}

calling code in different translation unit:

extern template int myFunc<2,3>(obj_t obj)
myFunc<2,3>(obj);
```

This C++ code will be fully expanded by the compiler. No conditionals present in the generated assembly code.

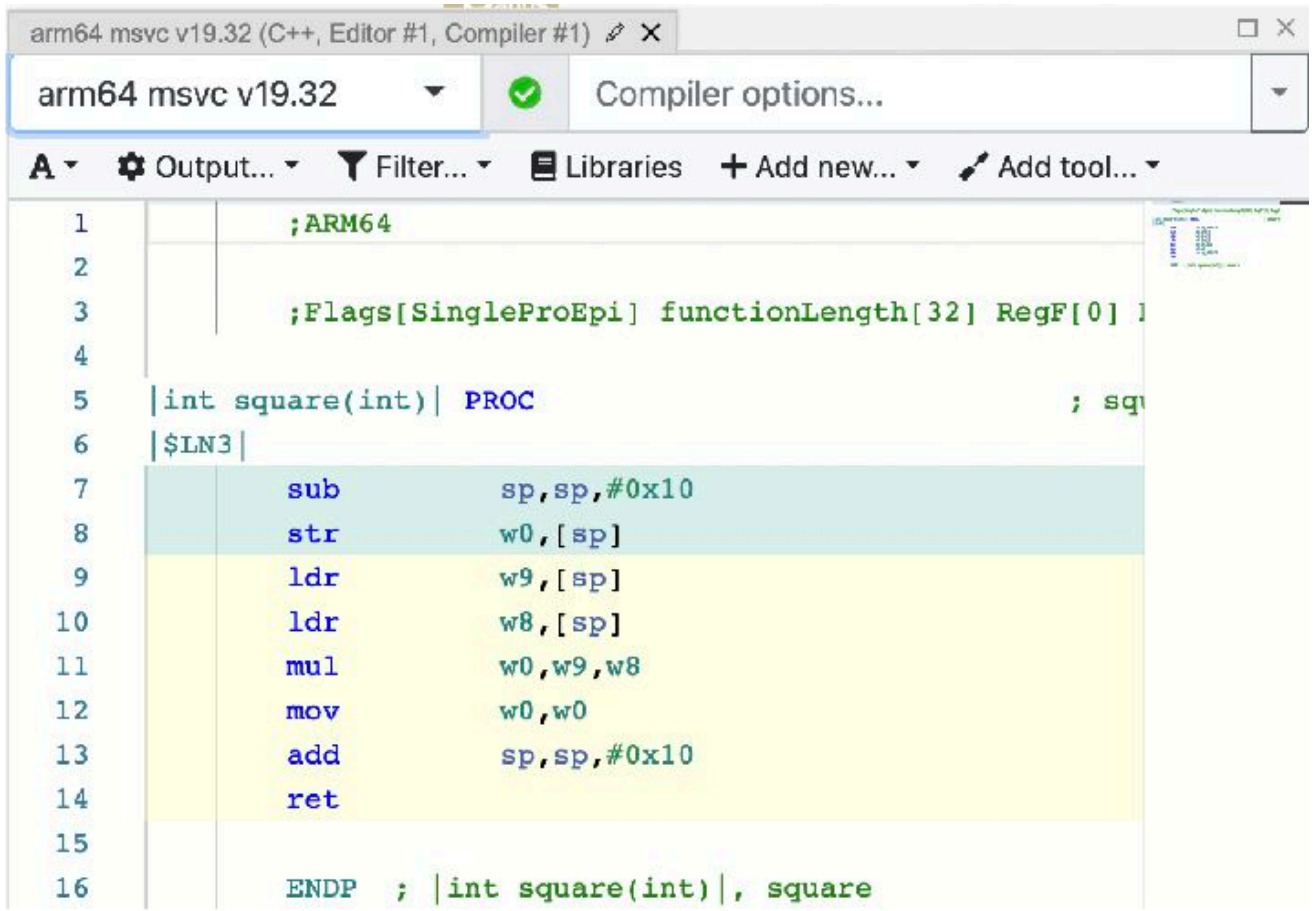
GODBOLT.ORG - COMPILER AUTOPSIES



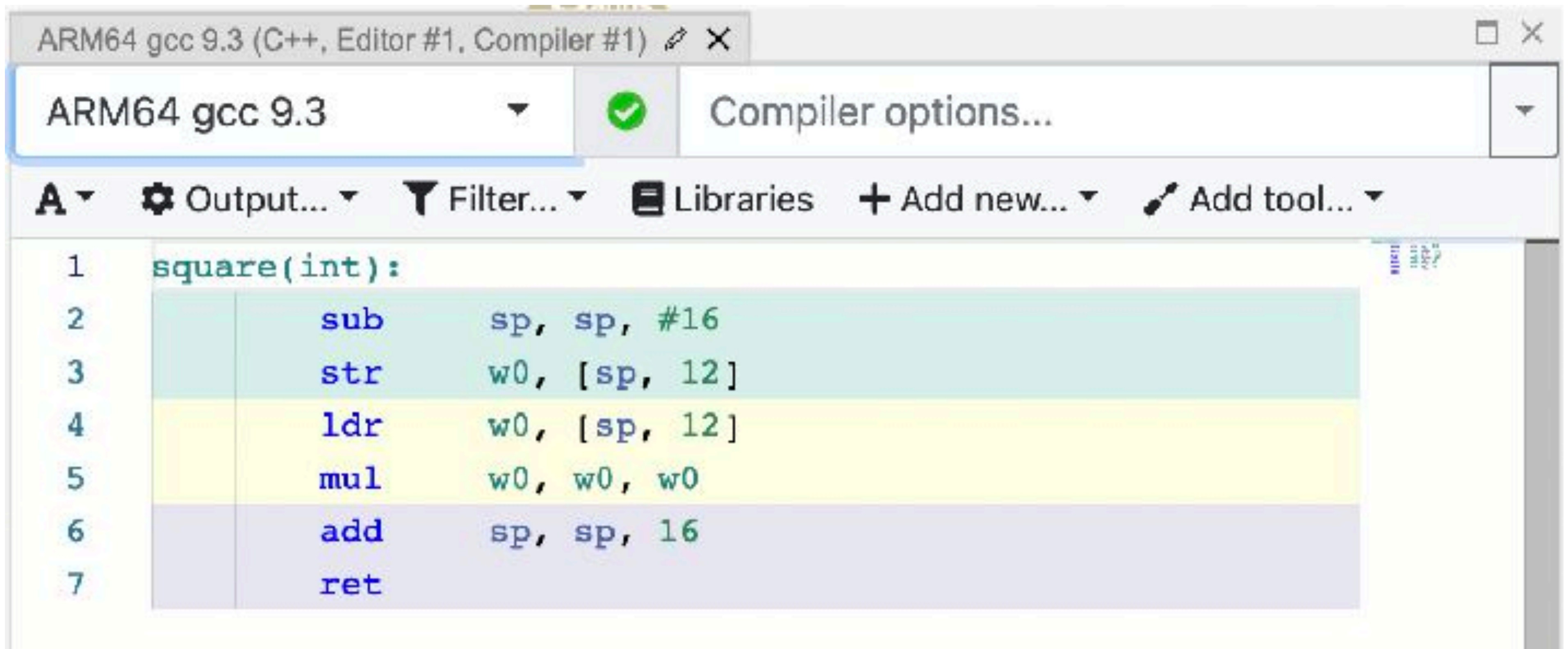
```
// Type your code here, or load an example.
int square(int num) {
    return num * num;
}
```



```
square(int):                            # @square(int)
        push    rbp
        mov     rbp, rsp
        mov     dword ptr [rbp - 4], edi
        mov     eax, dword ptr [rbp - 4]
        imul    eax, dword ptr [rbp - 4]
        pop     rbp
        ret
```



```
        ;ARM64

        ;Flags[SingleProEpi] functionLength[32] RegF[0] I

|int square(int)| PROC                          ; squ
|$LN3|
        sub         sp,sp,#0x10
        str         w0,[sp]
        ldr         w9,[sp]
        ldr         w8,[sp]
        mul         w0,w9,w8
        mov         w0,w0
        add         sp,sp,#0x10
        ret

        ENDP  ; |int square(int)|, square
```



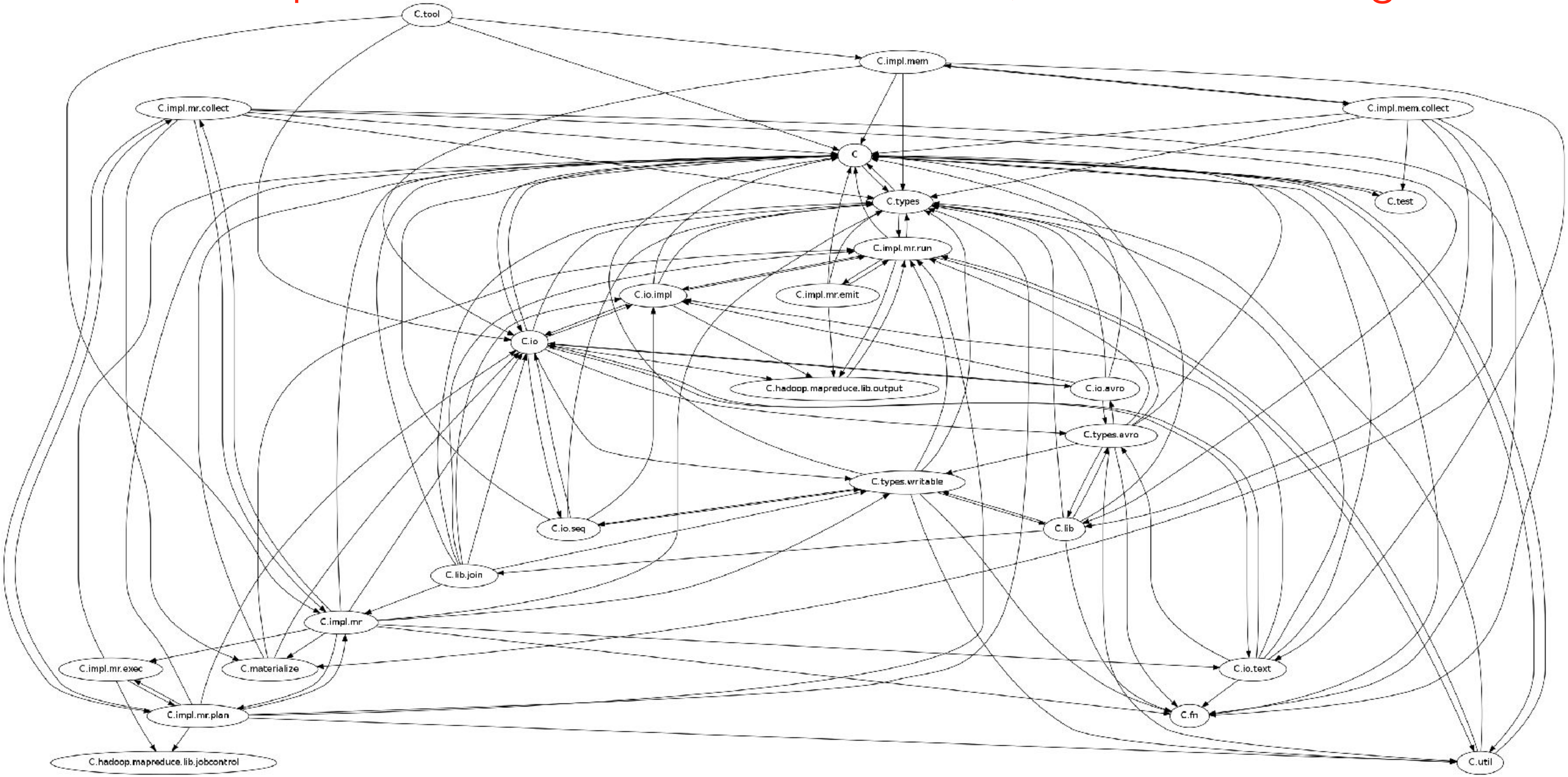
```
square(int):
        sub     sp, sp, #16
        str     w0, [sp, 12]
        ldr     w0, [sp, 12]
        mul     w0, w0, w0
        add     sp, sp, 16
        ret
```

FINDING & PREVENTING BUGS

MODULARIZATION

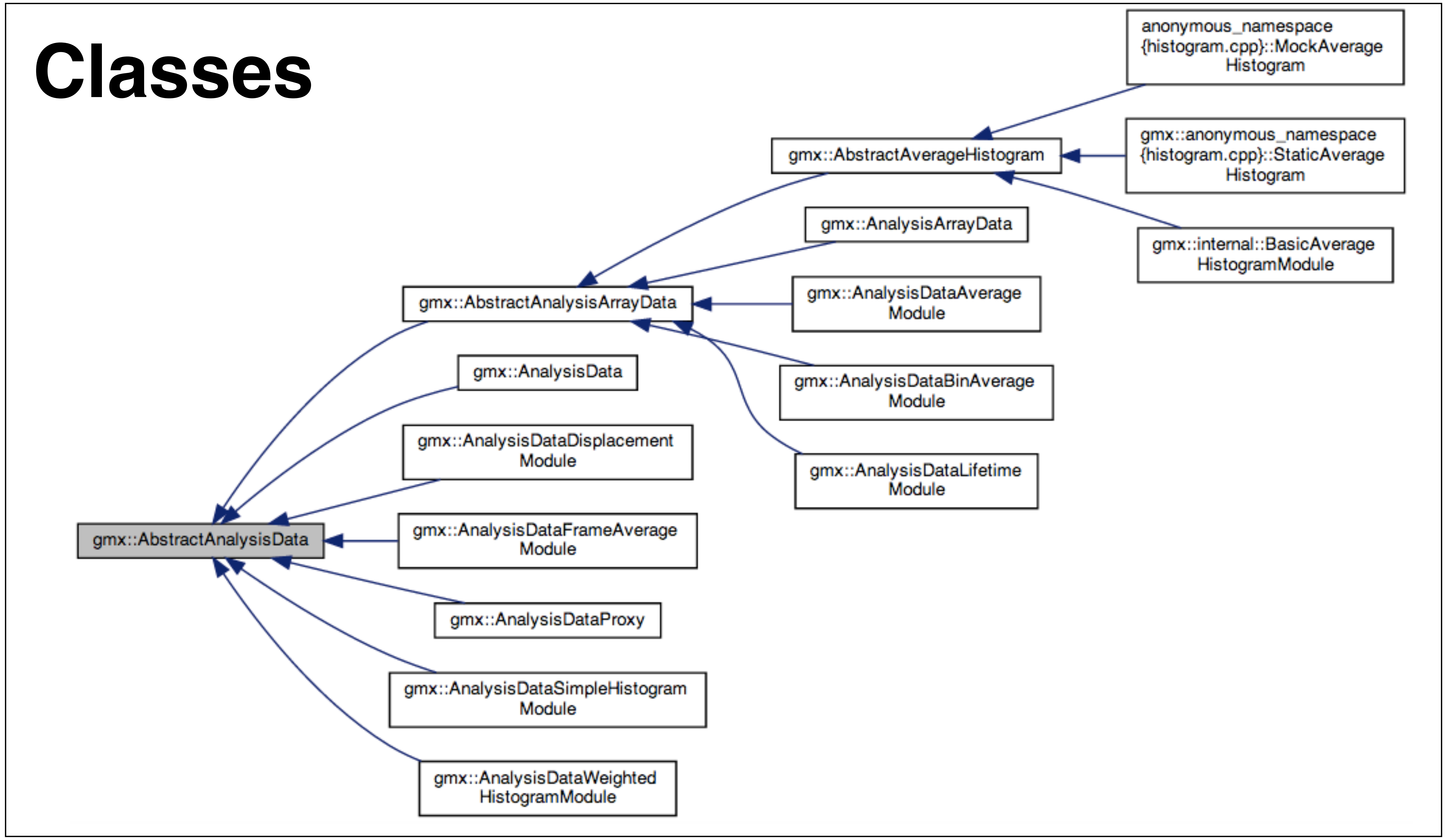
- Avoid code inter-dependencies
- Have modules doing clearly separate tasks
- Have a clear (documented) API for each module
- Make sure all code is thread-safe!
- Strict code organization:
- One directory per module, e.g. src/foo - with documentation for that module
- The ‘bar’ class is declared in src/foo/bar.h, implemented in src/foo/bar.cpp
- Write unit tests, not only regression tests
- Unit tests for ‘bar’ class are placed in src/foo/tests/bar.cpp
- Design-for-Testability (DFT):
  Write unit test first, then the code implementation
- Controversial (?): Move to C++

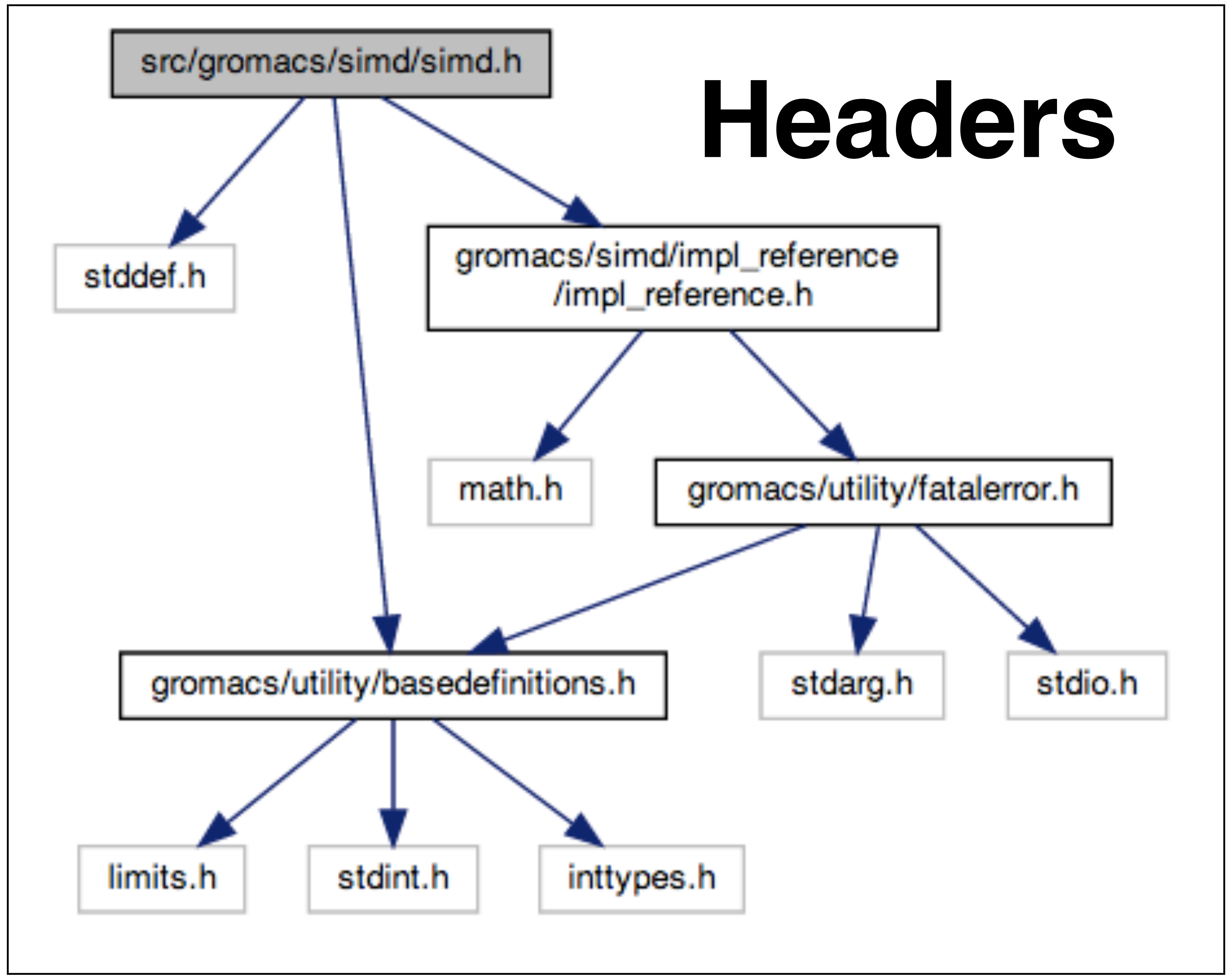
Circular dependencies are bad. If a test fails, where is the bug here?



"It has been discovered that C++ provides a remarkable facility for concealing the trivial details of a program - such as where its bugs are." (David Keppel)

JUST SAY ‘NO’ TO CIRCULAR DEPENDENCIES


Classes
anonymous_namespace {histogram.cpp}::MockAverage Histogram
gmx::AbstractAverageHistogram
gmx::anonymous_namespace {histogram.cpp}::StaticAverage Histogram
gmx::internal::BasicAverage HistogramModule
gmx::AnalysisArrayData
gmx::AbstractAnalysisArrayData
gmx::AnalysisDataAverage Module
gmx::AnalysisDataBinAverage Module
gmx::AnalysisDataLifetime Module
gmx::AnalysisData
gmx::AnalysisDataDisplacement Module
gmx::AbstractAnalysisData
gmx::AnalysisDataFrameAverage Module
gmx::AnalysisDataProxy
gmx::AnalysisDataSimpleHistogram Module
gmx::AnalysisDataWeighted HistogramModule


src/gromacs/simd/simd.h
Headers
stddef.h
gromacs/simd/impl_reference /impl_reference.h
math.h
gromacs/utility/fatalerror.h
gromacs/utility/basedefinitions.h
stdarg.h
stdio.h
limits.h
stdint.h
inttypes.h

This is hard, but Doxygen helps you detect it

For GROMACS, our code management system will not allow any developer to submit a file with a circular dependency.

UNIT TESTING

BE AGGRESSIVE IN TESTING: "ДОВЕРЯЙ, НО ПРОВЕРЯЙ" (TRUST, BUT VERIFY)

googletest - Google C++ Testing Framework - Google Project Hosting
https code.google.com/p/googletest/
Umeå Univer...mi i Skolan
How to Do ...n — Medium
Computation...gy Council
Varför denna...naljapen.se
How to make...s Technica
Low Resolut...lar Domains
Examining p...Med
erik.lindahl@gmail.com ▾ | My favorites ▾ |
googletest
Google C++ Testing Framework
Project Home
Downloads
Wiki
Issues
Source
Summary
People
Project Information
Starred by 2339 users
Project feeds
Code license
New BSD License
Labels
Cplusplus, Testing, Framework, Tests, Unittests, Cpp, Google
Members
j...@google.com, zhanyong...@gmail.com, w...@google.com, ko...@google.com,
g+1
1k

Google's framework for writing C++ tests on a variety of platforms (Linux, Mac OS X, Windows, Cygwin, Windows CE, and Symbian). Based on the xUnit architecture. Supports automatic test discovery, a rich set of assertions, user-defined assertions, death tests, fatal and non-fatal failures, value- and type-parameterized tests, various options for running the tests, and XML test report generation.

Getting Started ¶

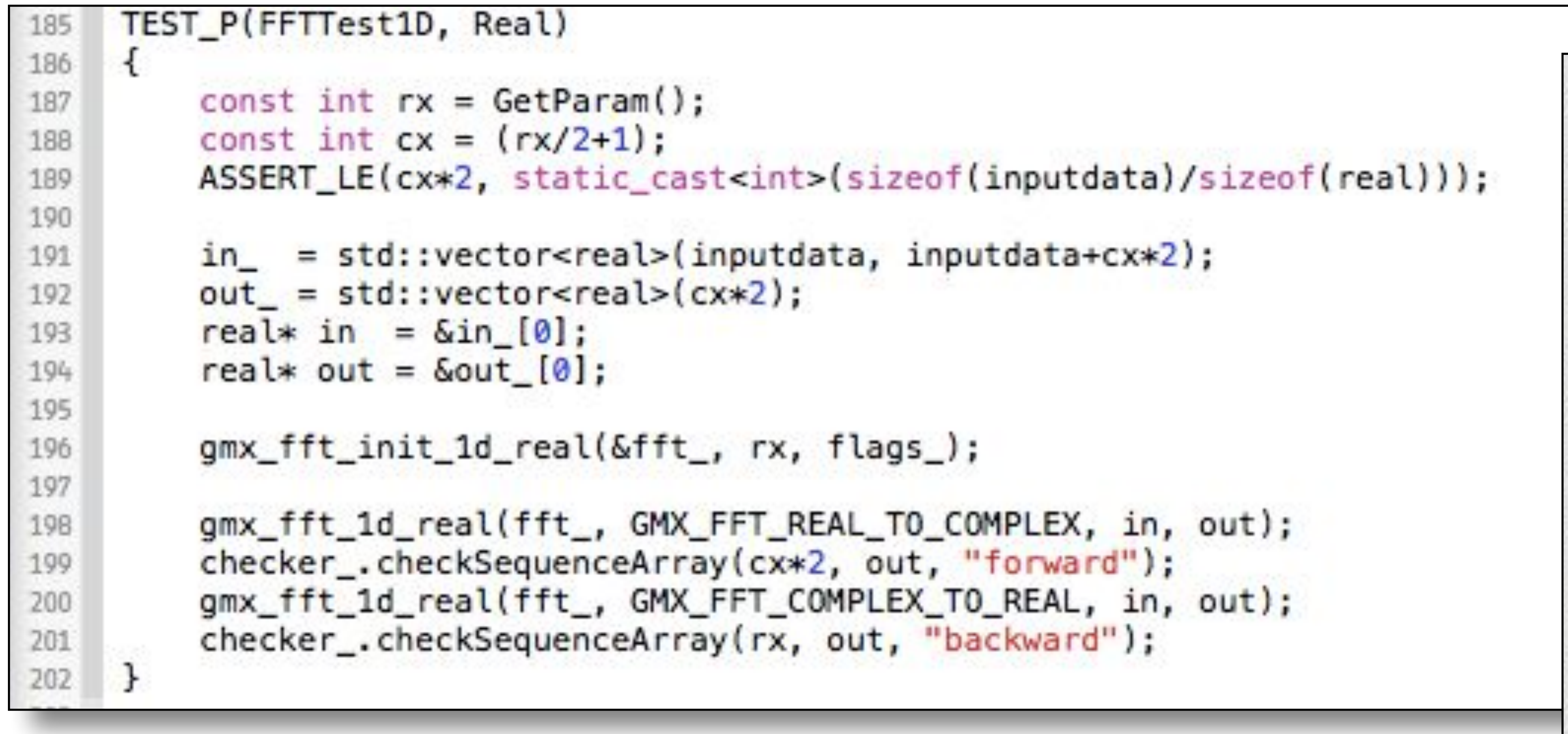
After downloading Google Test, unpack it, read the README file and the documentation wiki pages (listed on the right side of this front page).

Who Is Using Google Test?

In addition to many internal projects at Google, Google Test is also used by the following notable projects:

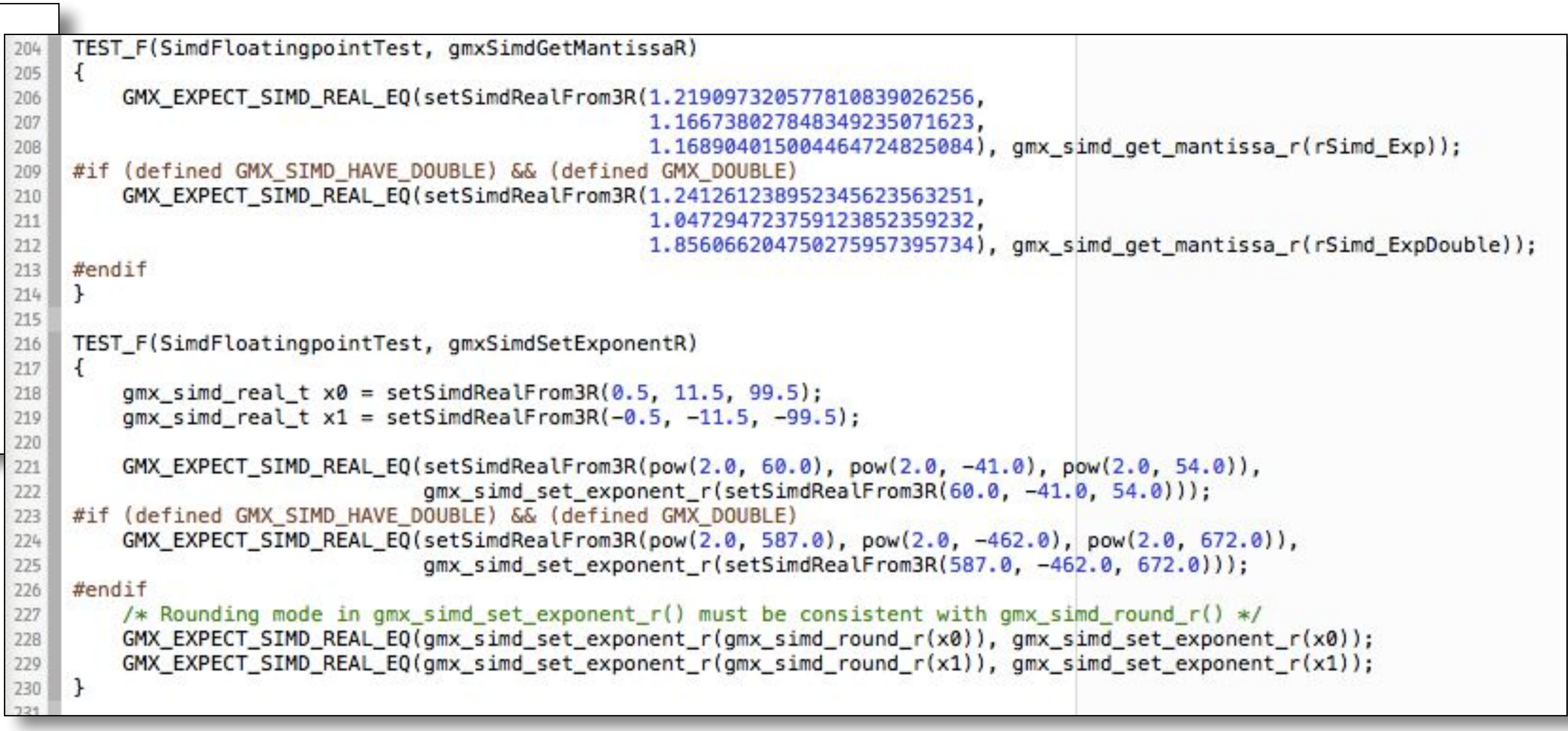
- The Chromium projects (behind the Chrome browser and Chrome OS)
- The LLVM compiler
- Protocol Buffers (Google's data interchange format)

Example Gromacs unit tests:



```
TEST_P(FFTTest1D, Real)
{
    const int rx = GetParam();
    const int cx = (rx/2+1);
    ASSERT_LE(cx*2, static_cast<int>(sizeof(inputdata)/sizeof(real)));

    in_  = std::vector<real>(inputdata, inputdata+cx*2);
    out_ = std::vector<real>(cx*2);
    real* in  = &in_[0];
    real* out = &out_[0];

    gmx_fft_init_1d_real(&fft_, rx, flags_);

    gmx_fft_1d_real(fft_, GMX_FFT_REAL_TO_COMPLEX, in, out);
    checker_.checkSequenceArray(cx*2, out, "forward");
    gmx_fft_1d_real(fft_, GMX_FFT_COMPLEX_TO_REAL, in, out);
    checker_.checkSequenceArray(rx, out, "backward");
}
```

The idea is that you should test *everything*



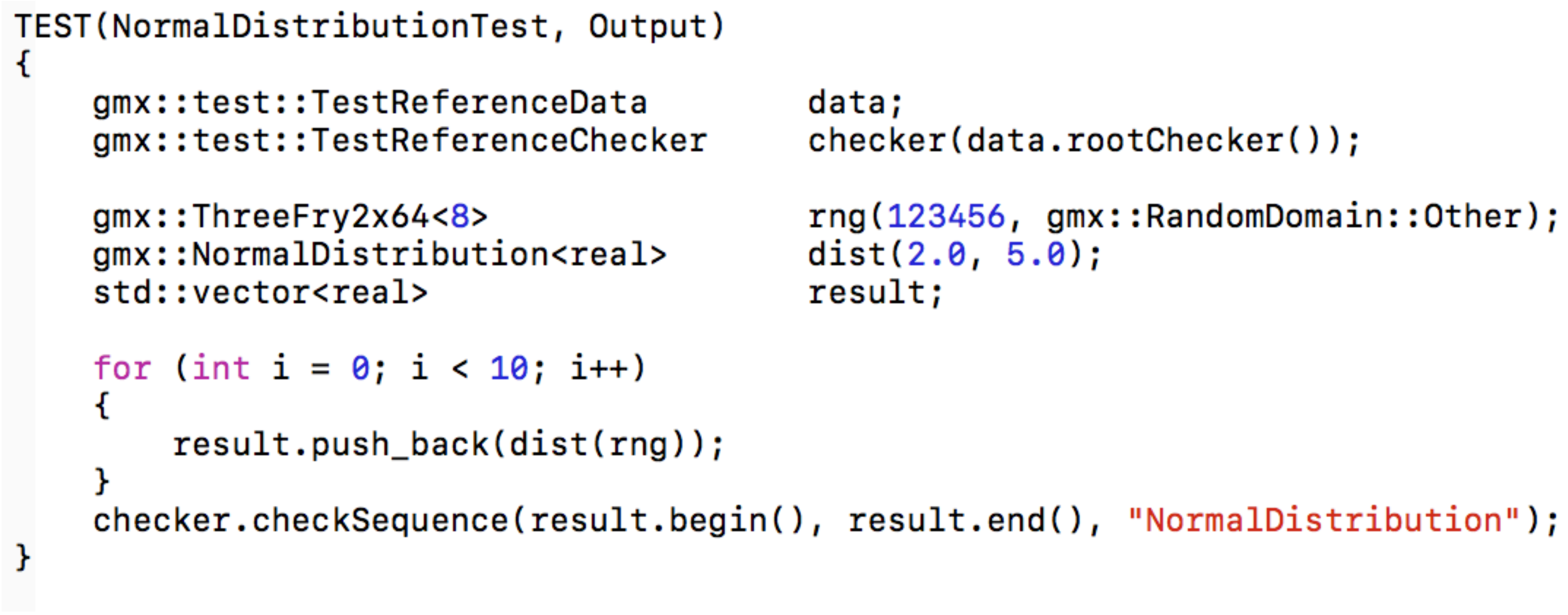
```
TEST_F(SimdFloatingpointTest, gmxSimdGetMantissaR)
{
    GMX_EXPECT_SIMD_REAL_EQ(setSimdRealFrom3R(1.219097320577810839026256,
                                              1.166738027848349235071623,
                                              1.168904015004464724825084), gmx_simd_get_mantissa_r(rSimd_Exp));
#if (defined GMX_SIMD_HAVE_DOUBLE) && (defined GMX_DOUBLE)
    GMX_EXPECT_SIMD_REAL_EQ(setSimdRealFrom3R(1.241261238952345623563251,
                                              1.047294723759123852359232,
                                              1.856066204750275957395734), gmx_simd_get_mantissa_r(rSimd_ExpDouble));
#endif
}

TEST_F(SimdFloatingpointTest, gmxSimdSetExponentR)
{
    gmx_simd_real_t x0 = setSimdRealFrom3R(0.5, 11.5, 99.5);
    gmx_simd_real_t x1 = setSimdRealFrom3R(-0.5, -11.5, -99.5);

    GMX_EXPECT_SIMD_REAL_EQ(setSimdRealFrom3R(pow(2.0, 60.0), pow(2.0, -41.0), pow(2.0, 54.0)),
                            gmx_simd_set_exponent_r(setSimdRealFrom3R(60.0, -41.0, 54.0)));
#if (defined GMX_SIMD_HAVE_DOUBLE) && (defined GMX_DOUBLE)
    GMX_EXPECT_SIMD_REAL_EQ(setSimdRealFrom3R(pow(2.0, 587.0), pow(2.0, -462.0), pow(2.0, 672.0)),
                            gmx_simd_set_exponent_r(setSimdRealFrom3R(587.0, -462.0, 672.0)));
#endif
    /* Rounding mode in gmx_simd_set_exponent_r() must be consistent with gmx_simd_round_r() */
    GMX_EXPECT_SIMD_REAL_EQ(gmx_simd_set_exponent_r(gmx_simd_round_r(x0)), gmx_simd_set_exponent_r(x0));
    GMX_EXPECT_SIMD_REAL_EQ(gmx_simd_set_exponent_r(gmx_simd_round_r(x1)), gmx_simd_set_exponent_r(x1));
}
```

Do you think it’s overkill to test that hardware rounding works? In March 2014, this very test caught that IBM Power7 VMX uses different rounding modes for SIMD and normal floating-point to integer conversions…

Spring 2018: Our unit tests caught that IBM had semi-silently had to change their *binary* ABI for Power8/9 since their compiler specifications partly violated the C++ standard. Fedora running all our unit tests caught it immediately, and a few hours later we had a workaround in the code.

Spring 2019: Our unit tests failed on the specific combination of gcc-7 and Intel AVX-512 hardware, but only with -O3 flags. Turned out to be a bug in the gcc-7 AVX-512 loop unrolling optimization (Godbolt!)

In C++, each method in a class should ideally have exhaustive unit tests



```cpp
TEST(NormalDistributionTest, Output)
{
    gmx::test::TestReferenceData        data;
    gmx::test::TestReferenceChecker     checker(data.rootChecker());

    gmx::ThreeFry2x64<8>                rng(123456, gmx::RandomDomain::Other);
    gmx::NormalDistribution<real>       dist(2.0, 5.0);
    std::vector<real>                   result;

    for (int i = 0; i < 10; i++)
    {
        result.push_back(dist(rng));
    }
    checker.checkSequence(result.begin(), result.end(), "NormalDistribution");
}
```

Are *you* aware of the peculiarities of rounding differences depending on whether your CPU hardware uses fused multiply-add (FMA) vs. separate multiply & add?

Test that a simple call to a normal distribution random generator returns the expected 10 numbers.

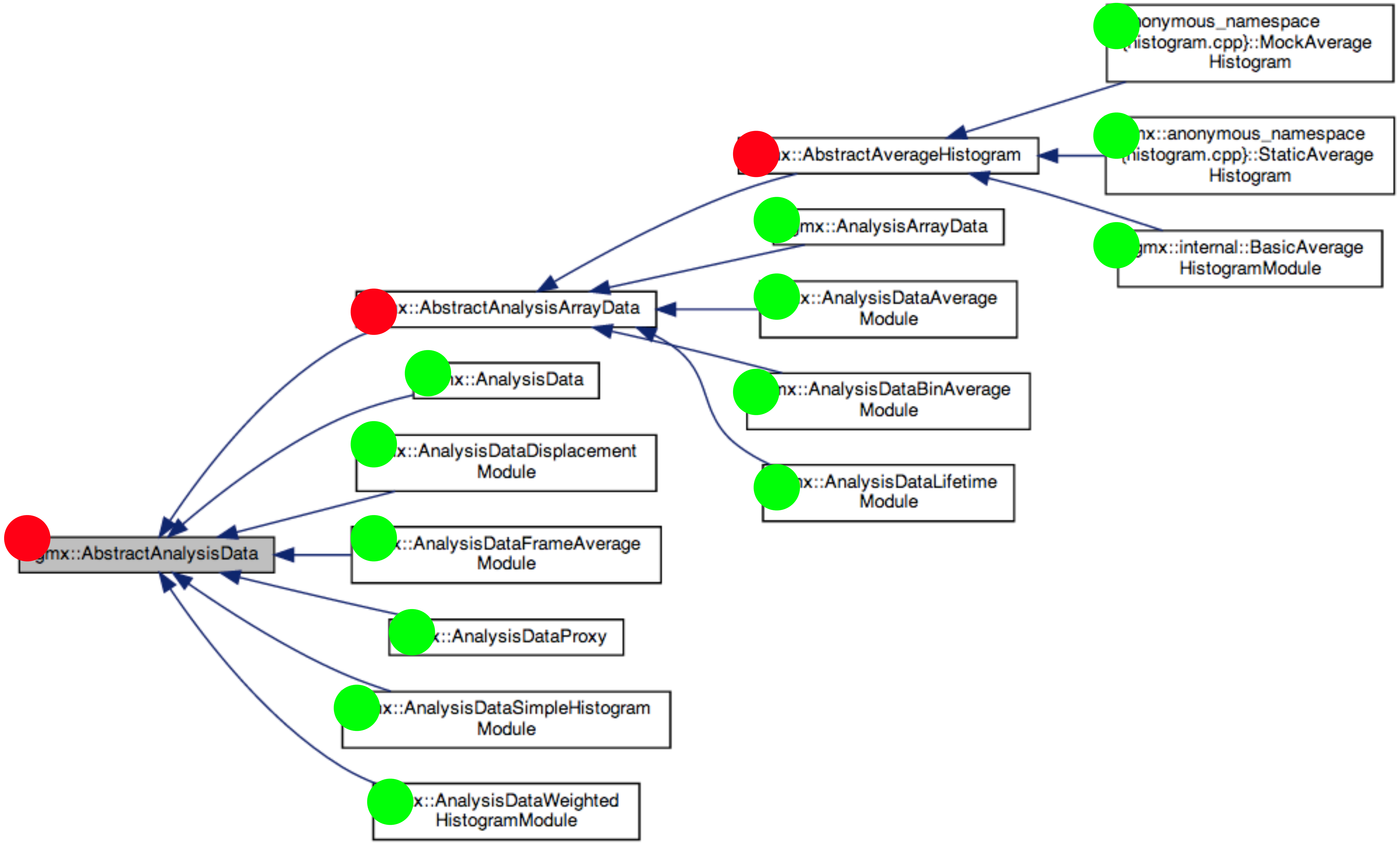
Why? Because we found that libstdc++ and libcxx do not use the same algorithm, so code will not produce the same results. We now use our own algorithm implementation - make sure it keeps working.

No need to ask: Of course we have integrated GoogleTest support into the IHPCSS/software-engineering repo - but I have not had time to write the actual tests. However, as you add more tests, they will all execute if you just issue “make check”.

Imagine a project with ~1000 classes, and that the class diagram below is a small excerpt (it’s from Gromacs).

All classes have close-to-exhaustive unit tests - but your latest build now fails the unit test.
Green means the unit test for this class was OK, red means it failed.

Where do you look for the bug?

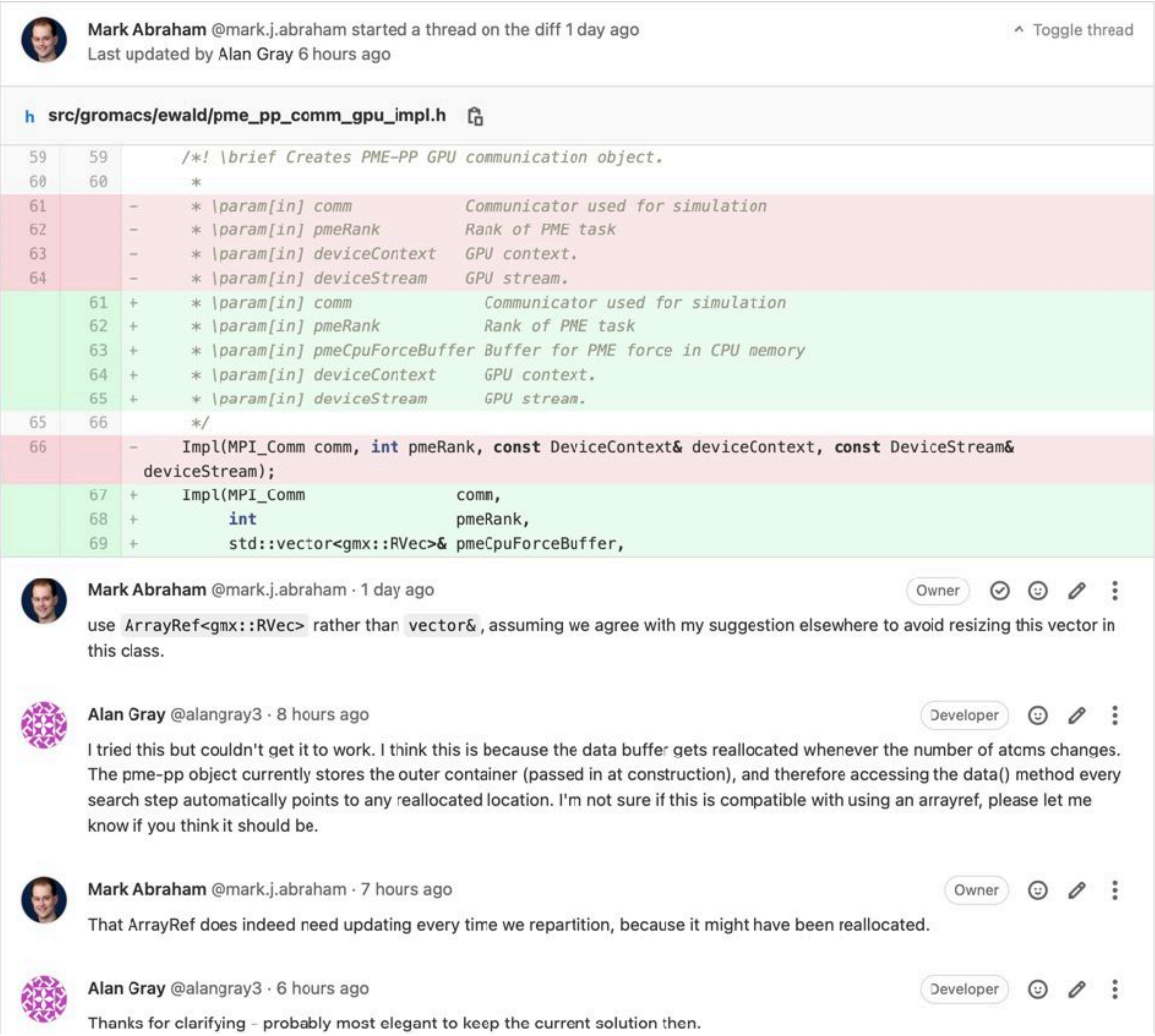

onymous_namespace
histogram.cpp}::MockAverage
Histogram
x::AbstractAverageHistogram
nx::anonymous_namespace
histogram.cpp}::StaticAverage
Histogram
gmx::internal::BasicAverage
HistogramModule
gmx::AnalysisArrayData
x::AnalysisDataAverage
Module
x::AbstractAnalysisArrayData
nx::AnalysisDataBinAverage
Module
x::AnalysisDataLifetime
Module
x::AnalysisData
x::AnalysisDataDisplacement
Module
gmx::AbstractAnalysisData
x::AnalysisDataFrameAverage
Module
x::AnalysisDataProxy
x::AnalysisDataSimpleHistogram
Module
x::AnalysisDataWeighted
HistogramModule

If each unit test targets a small method/function, you have isolated the bug to within ~50 lines-of-code before even opening your editor.

COMMITS - HOW CHANGES MAKE IT INTO A LARGE CODEBASE

- Who is allowed to write to your code repository?
- Especially problematic if you suspect some less talented developers might submit buggy code
- Such as this one:

FORMAL CODE REVIEW - NOBODY CAN COMMIT DIRECTLY!



Mark Abraham @mark.j.abraham started a thread on the diff 1 day ago
Last updated by Alan Gray 6 hours ago

src/gromacs/ewald/pme_pp_comm_gpu_impl.h

```
      /*! \brief Creates PME-PP GPU communication object.
       *
-      * \param[in] comm           Communicator used for simulation
-      * \param[in] pmeRank        Rank of PME task
-      * \param[in] deviceContext   GPU context.
-      * \param[in] deviceStream    GPU stream.
+      * \param[in] comm             Communicator used for simulation
+      * \param[in] pmeRank          Rank of PME task
+      * \param[in] pmeCpuForceBuffer Buffer for PME force in CPU memory
+      * \param[in] deviceContext    GPU context.
+      * \param[in] deviceStream     GPU stream.
       */
-     Impl(MPI_Comm comm, int pmeRank, const DeviceContext& deviceContext, const DeviceStream&
 deviceStream);
+     Impl(MPI_Comm                    comm,
+          int                         pmeRank,
+          std::vector<gmx::RVec>& pmeCpuForceBuffer,
```

Mark Abraham @mark.j.abraham · 1 day ago

use `ArrayRef<gmx::RVec>` rather than `vector&`, assuming we agree with my suggestion elsewhere to avoid resizing this vector in this class.

Alan Gray @alangray3 · 8 hours ago

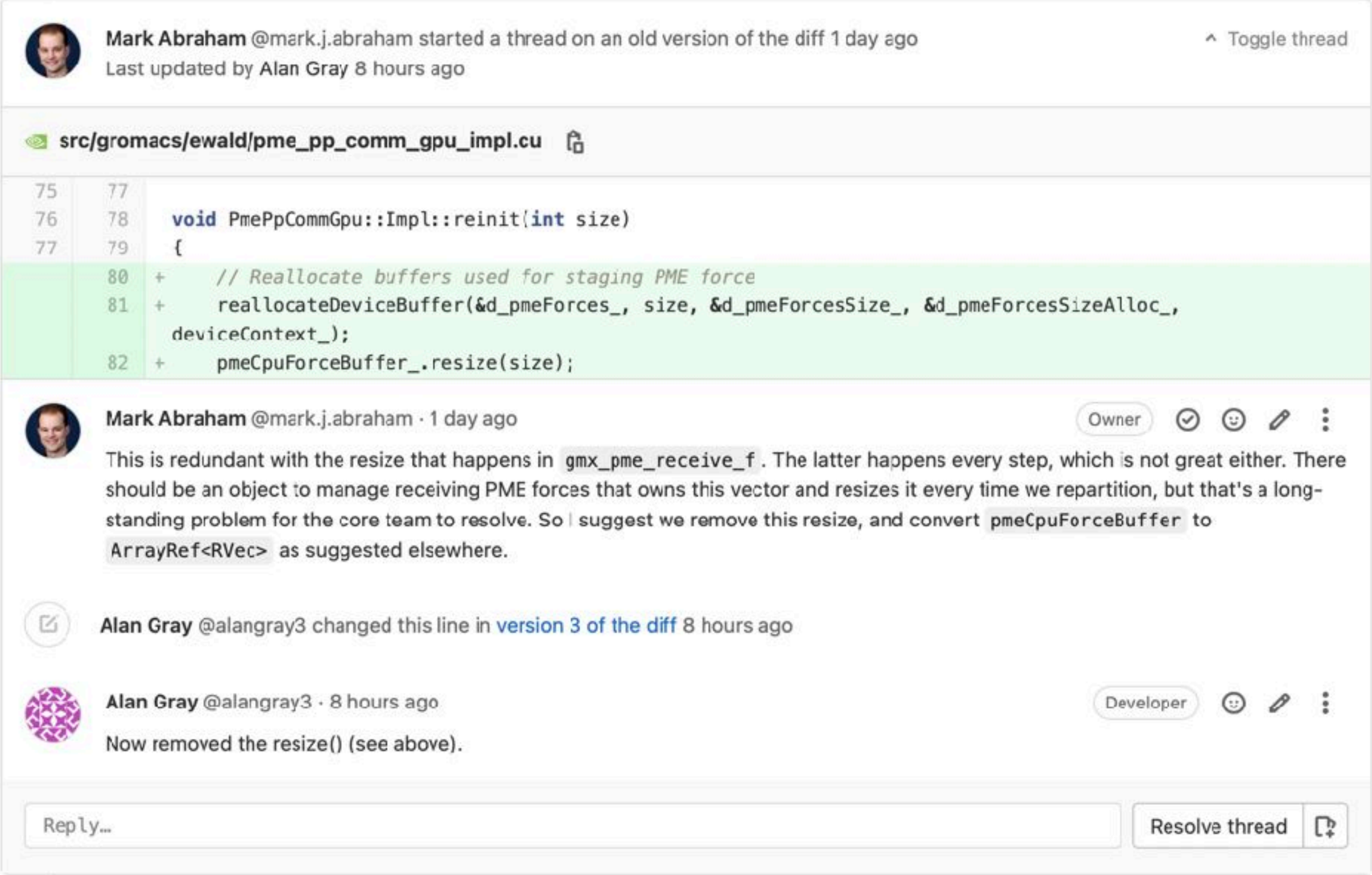
I tried this but couldn't get it to work. I think this is because the data buffer gets reallocated whenever the number of atoms changes. The pme-pp object currently stores the outer container (passed in at construction), and therefore accessing the data() method every search step automatically points to any reallocated location. I'm not sure if this is compatible with using an arrayref, please let me know if you think it should be.

Mark Abraham @mark.j.abraham · 7 hours ago

That ArrayRef does indeed need updating every time we repartition, because it might have been reallocated.

Alan Gray @alangray3 · 6 hours ago

Thanks for clarifying - probably most elegant to keep the current solution then.



Mark Abraham @mark.j.abraham started a thread on an old version of the diff 1 day ago
Last updated by Alan Gray 8 hours ago

src/gromacs/ewald/pme_pp_comm_gpu_impl.cu

```

   void PmePpCommGpu::Impl::reinit(int size)
   {
+      // Reallocate buffers used for staging PME force
+      reallocateDeviceBuffer(&d_pmeForces_, size, &d_pmeForcesSize_, &d_pmeForcesSizeAlloc_,
   deviceContext_);
+      pmeCpuForceBuffer_.resize(size);
```

Mark Abraham @mark.j.abraham · 1 day ago

This is redundant with the resize that happens in `gmx_pme_receive_f`. The latter happens every step, which is not great either. There should be an object to manage receiving PME forces that owns this vector and resizes it every time we repartition, but that's a long-standing problem for the core team to resolve. So I suggest we remove this resize, and convert `pmeCpuForceBuffer` to `ArrayRef<RVec>` as suggested elsewhere.

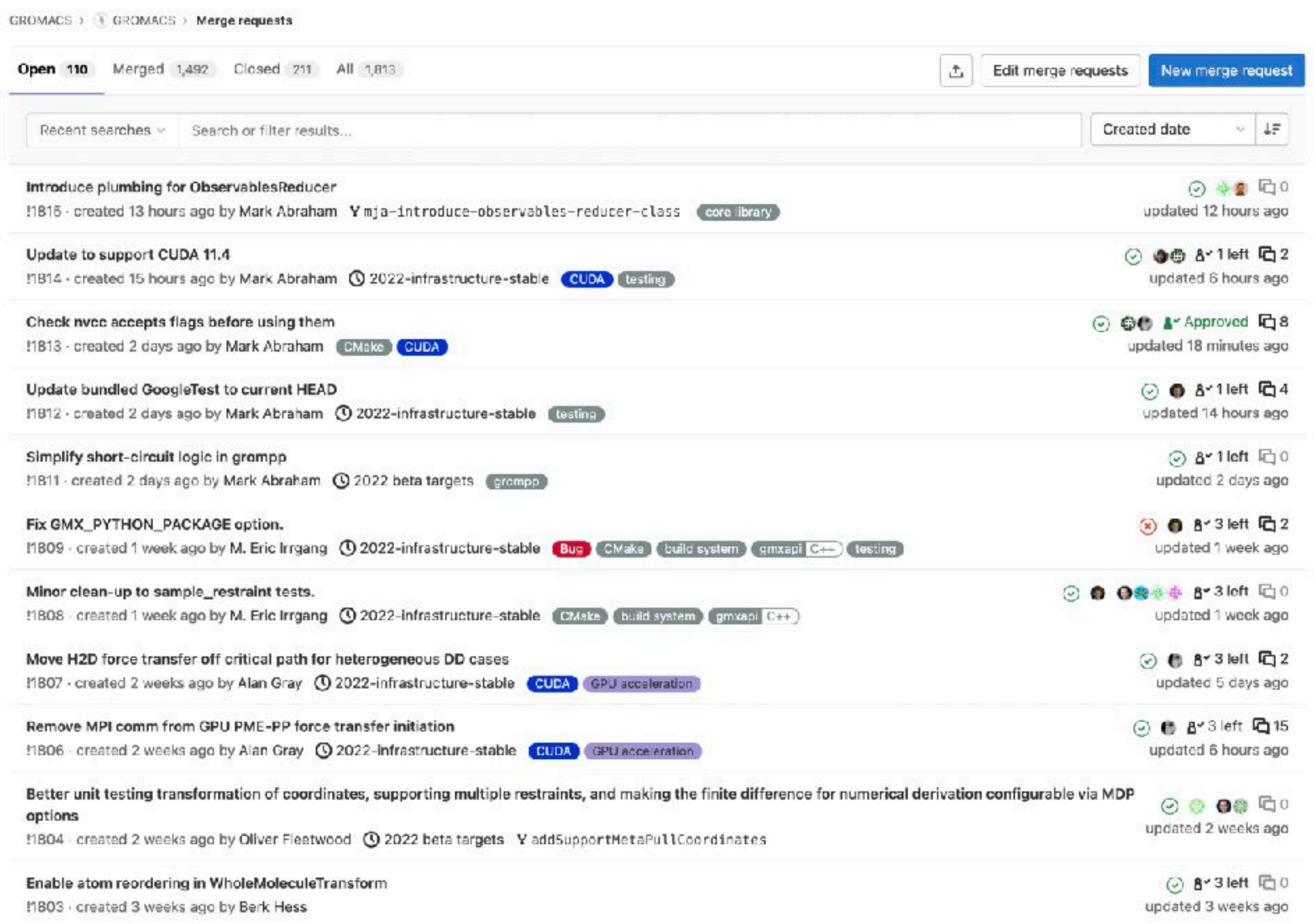
Alan Gray @alangray3 changed this line in version 3 of the diff 8 hours ago

Alan Gray @alangray3 · 8 hours ago

Now removed the resize() (see above).

Each new comment on the MR (merge request) in GitLab opens a “thread”, which is “resolved” when the commenter is happy.

We have a TON of merge requests in flight. With full dependency tracking, patches can be rebased onto others by hitting a rebase button, or even edited on-the-fly in the window



Introduce plumbing for ObservablesReducer
!1815 · created 13 hours ago by Mark Abraham mja-introduce-observables-reducer-class core library — 0 — updated 12 hours ago

Update to support CUDA 11.4
!1814 · created 15 hours ago by Mark Abraham 2022-infrastructure-stable CUDA testing — 1 left 2 — updated 6 hours ago

Check nvcc accepts flags before using them
!1813 · created 2 days ago by Mark Abraham CMake CUDA — Approved 8 — updated 18 minutes ago

Update bundled GoogleTest to current HEAD
!1812 · created 2 days ago by Mark Abraham 2022-infrastructure-stable testing — 1 left 4 — updated 14 hours ago

Simplify short-circuit logic in grompp
!1811 · created 2 days ago by Mark Abraham 2022 beta targets grompp — 1 left 0 — updated 2 days ago

Fix GMX_PYTHON_PACKAGE option.
!1809 · created 1 week ago by M. Eric Irrgang 2022-infrastructure-stable Bug CMake build system gmxapi C++ testing — 3 left 2 — updated 1 week ago

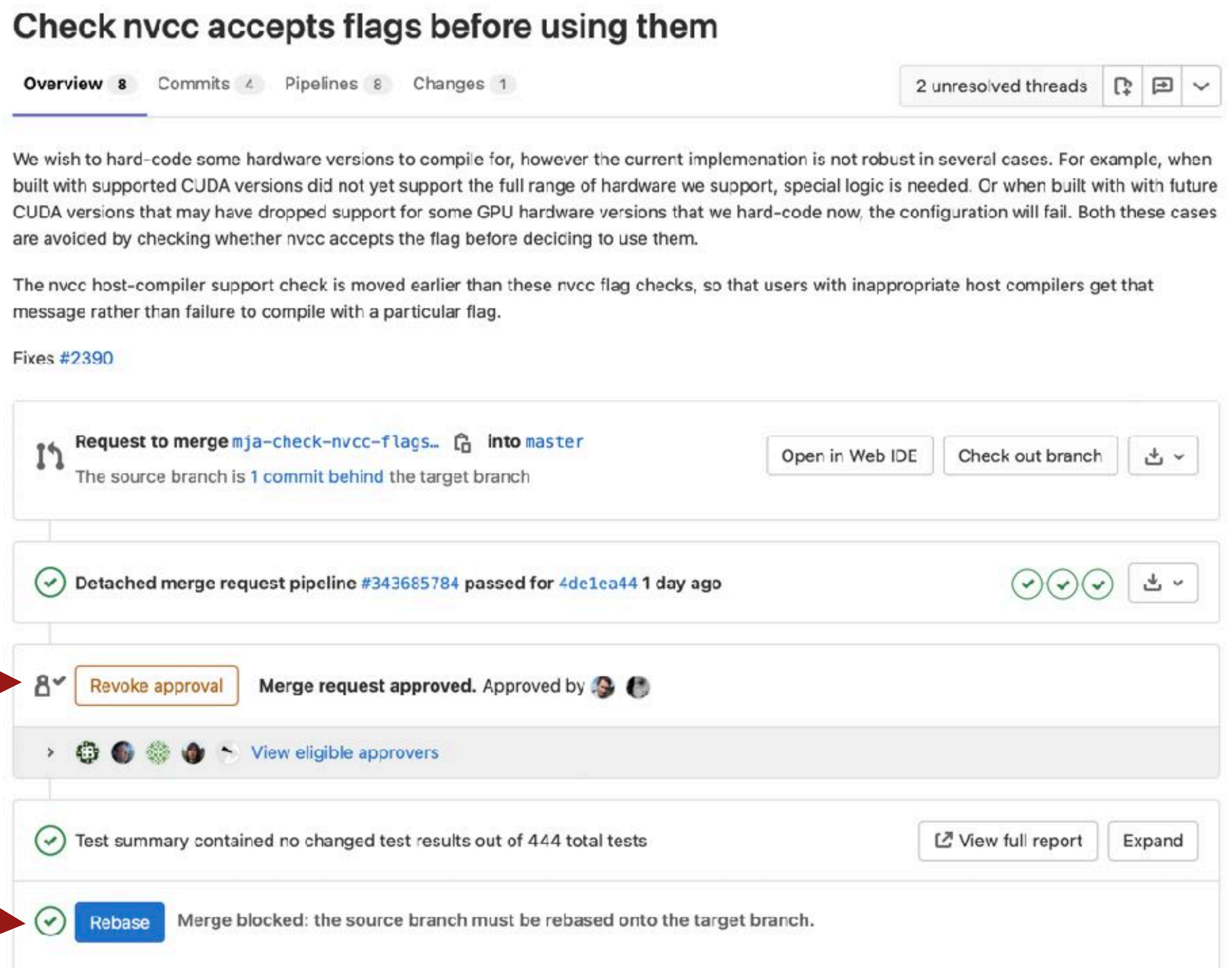
Minor clean-up to sample_restraint tests.
!1808 · created 1 week ago by M. Eric Irrgang 2022-infrastructure-stable CMake build system gmxapi C++ — 3 left 0 — updated 1 week ago

Move H2D force transfer off critical path for heterogeneous DD cases
!1807 · created 2 weeks ago by Alan Gray 2022-infrastructure-stable CUDA GPU acceleration — 3 left 2 — updated 5 days ago

Remove MPI comm from GPU PME-PP force transfer initiation
!1806 · created 2 weeks ago by Alan Gray 2022-infrastructure-stable CUDA GPU acceleration — 3 left 15 — updated 6 hours ago

Better unit testing transformation of coordinates, supporting multiple restraints, and making the finite difference for numerical derivation configurable via MDP options
!1804 · created 2 weeks ago by Oliver Fleetwood 2022 beta targets addSupportMetaPullCoordinates — 0 — updated 2 weeks ago

Enable atom reordering in WholeMoleculeTransform
!1803 · created 3 weeks ago by Berk Hess — 3 left 0 — updated 3 weeks ago



Check nvcc accepts flags before using them

We wish to hard-code some hardware versions to compile for, however the current implemenation is not robust in several cases. For example, when built with supported CUDA versions did not yet support the full range of hardware we support, special logic is needed. Or when built with with future CUDA versions that may have dropped support for some GPU hardware versions that we hard-code now, the configuration will fail. Both these cases are avoided by checking whether nvcc accepts the flag before deciding to use them.

The nvcc host-compiler support check is moved earlier than these nvcc flag checks, so that users with inappropriate host compilers get that message rather than failure to compile with a particular flag.

Fixes #2390

Request to merge mja-check-nvcc-flags... into master

The source branch is 1 commit behind the target branch

Detached merge request pipeline #343685784 passed for 4de1ca44 1 day ago

Merge request approved. Approved by

View eligible approvers

Test summary contained no changed test results out of 444 total tests

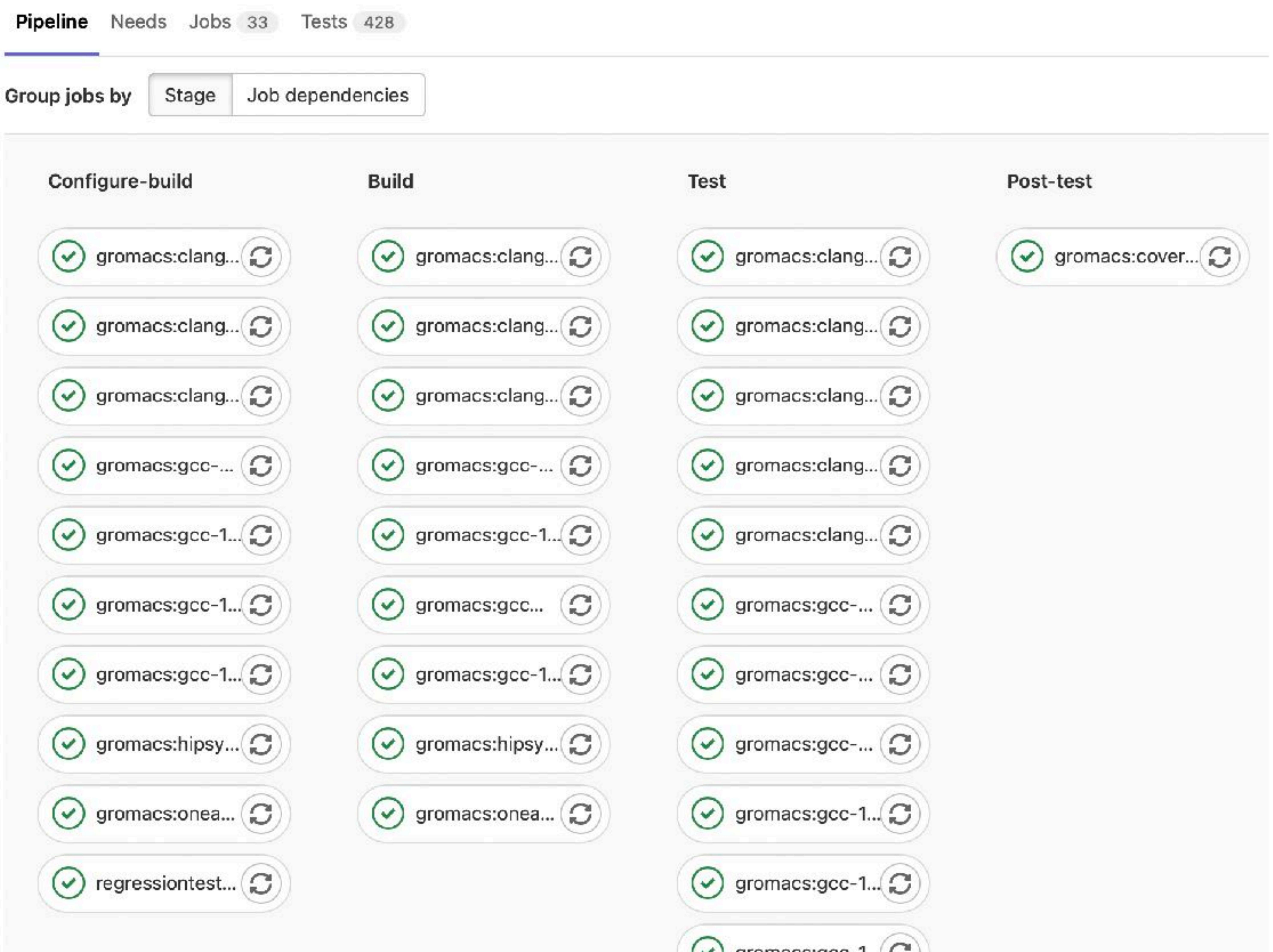
Merge blocked: the source branch must be rebased onto the target branch.

Anybody can add comments. When two eligible developers say OK (Erik + Szilard here), the patch can be merged into master by a maintainer - but note how GitLab blocks that until it's been rebased!

MAINTAINING QUALITY & AVOID BREAKING STUFF

CONTINUOUS INTEGRATION - PART OF OUR GITLAB ENVIRONMENT

- Every single merge request is tested automatically, including both builds & regression tests.
- Moderate usage is free both on GitHub (Travis) and GitLab
- For large usage, you can use your own servers or pay them
- Catches Cmake build errors
- Catches unit test failures
- We have separate nightly, weekly, monthly and release pipelines to do even more advanced checks (of physics)



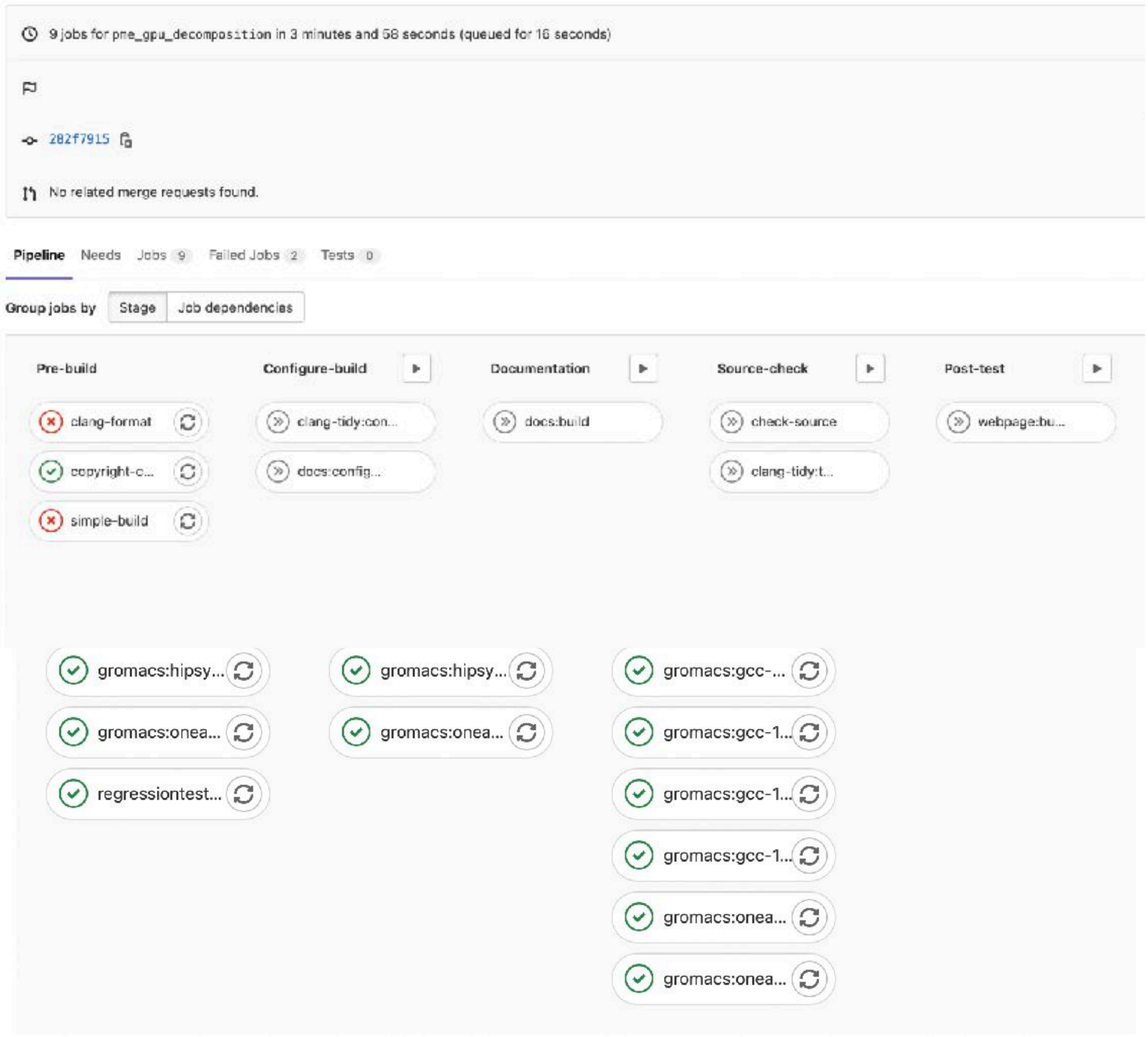
Pipeline Needs Jobs 33 Tests 428

Group jobs by Stage Job dependencies

Configure-build	Build	Test	Post-test
✓ gromacs:clang...	✓ gromacs:clang...	✓ gromacs:clang...	✓ gromacs:cover...
✓ gromacs:clang...	✓ gromacs:clang...	✓ gromacs:clang...	
✓ gromacs:clang...	✓ gromacs:clang...	✓ gromacs:clang...	
✓ gromacs:gcc-...	✓ gromacs:gcc-...	✓ gromacs:clang...	
✓ gromacs:gcc-1...	✓ gromacs:gcc-1...	✓ gromacs:clang...	
✓ gromacs:gcc-1...	✓ gromacs:gcc...	✓ gromacs:gcc-...	
✓ gromacs:gcc-1...	✓ gromacs:gcc-1...	✓ gromacs:gcc-...	
✓ gromacs:hipsy...	✓ gromacs:hipsy...	✓ gromacs:gcc-...	
✓ gromacs:onea...	✓ gromacs:onea...	✓ gromacs:gcc-1...	
✓ regressiontest...		✓ gromacs:gcc-1...	
		✓ gromacs:gcc-1	

GROMACS CI TESTS FOR EVERY SINGLE COMMIT

- Unit Tests: Do modules reproduce reference values?
  … on x86, Power 9, ARM, CUDA, OpenCL, SYCL CPU & SYCL GPU.
- Integration tests: Does a normal full run work?
- Regression tests: Are previous simulation results identical?
- Clang AddressSanitizer: Catch simple memory errors
- Clang MemorySanitizer: Like Valgrind - memory debugging
- Clang/GCC ThreadSanitizer: Thread synchronization errors
- Clang Static Analyzer: Logical execution dependency errors
- Cppcheck: Another static analyzer
- Clang-format: Proper code formatting, no tabs, brace standards?
- Doxygen: All classes/methods/arguments/variables documented?
- Weekly - physical validation tests: Do we reproduce statistical ensemble fluctuations?
- Weekly - performance tests: Ensure performance has not dropped (even by 1%) since last week



9 jobs for pre_gpu_decomposition in 3 minutes and 58 seconds (queued for 16 seconds)

282f7915

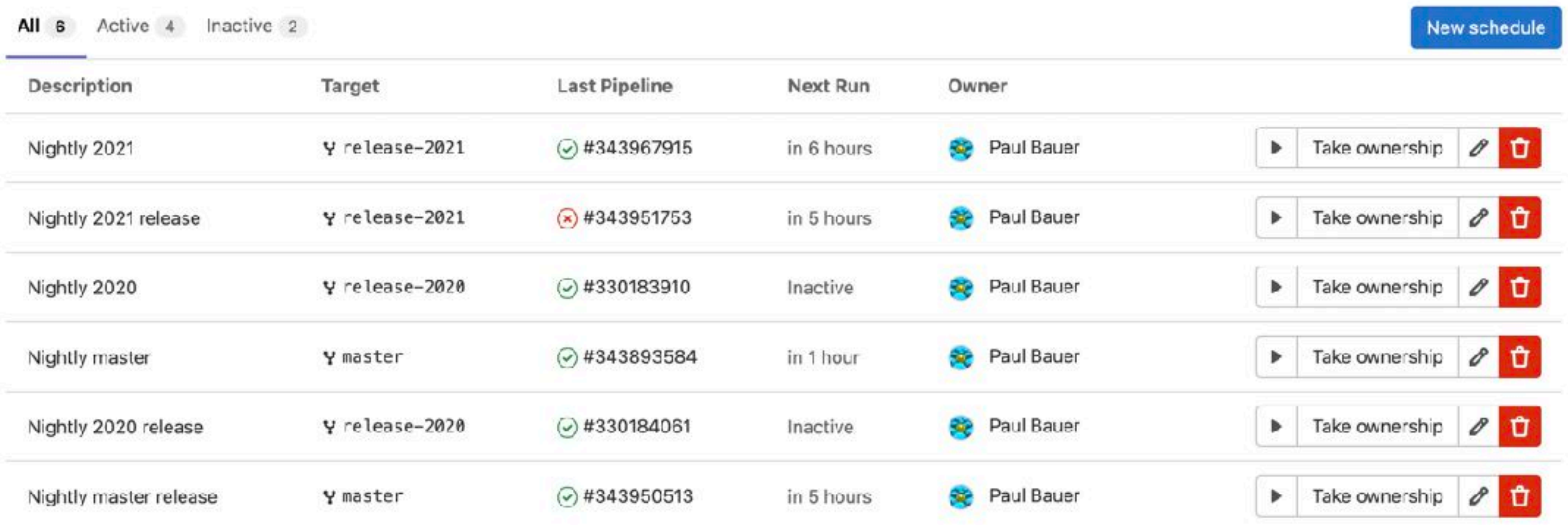
No related merge requests found.

Pre-submit GROMACS testing:

Changes cannot be committed until this entire pipeline is all green

Post-submit GROMACS testing:

Rare hardware and longer-running performance tests are performed once each patch has been approved, or nightly.



Description	Target	Last Pipeline	Next Run	Owner	
Nightly 2021	release-2021	✓ #343967915	in 6 hours	Paul Bauer	Take ownership
Nightly 2021 release	release-2021	✗ #343951753	in 5 hours	Paul Bauer	Take ownership
Nightly 2020	release-2020	✓ #330183910	Inactive	Paul Bauer	Take ownership
Nightly master	master	✓ #343893584	in 1 hour	Paul Bauer	Take ownership
Nightly 2020 release	release-2020	✓ #330184061	Inactive	Paul Bauer	Take ownership
Nightly master release	master	✓ #343950513	in 5 hours	Paul Bauer	Take ownership

TRAVIS CI: YOUR ALTERNATIVE FOR GITHUB

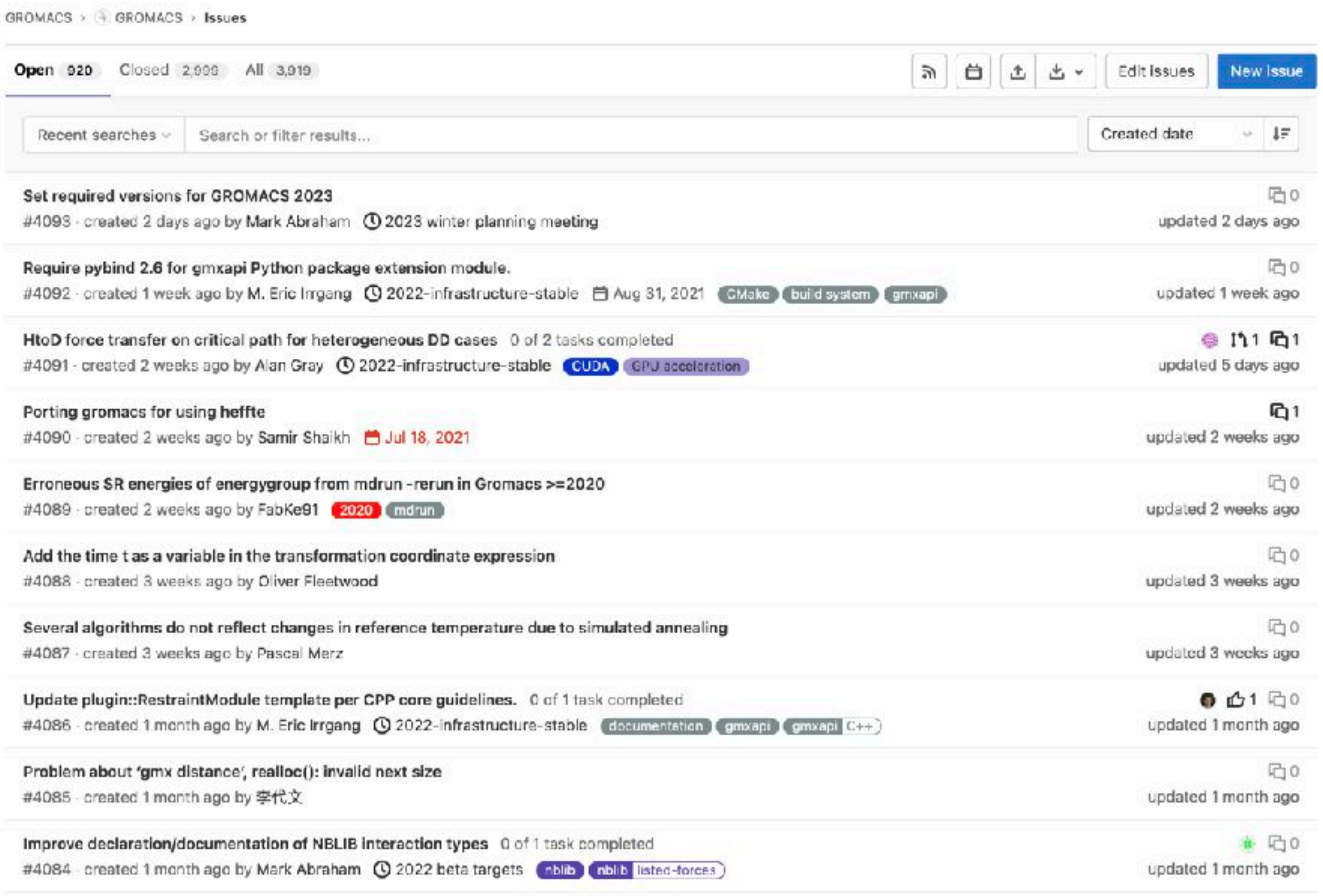
http://travis-ci.com

- Before moving to GitLab, GROMACS used Jenkins which is *very* powerful, but you need to set it up yourself to do advanced stuff, and/or arrange access to special hardware
- If your needs are more modest, Travis-CI is a much simpler environment that offers *free* CI testing of open source GitHub repositories
- Enabled for the IHPCSS-laplace repo: Every time I push an update, the code is built, followed by execution of the unit tests.
- If you look at the two badges at GitHub, green colors mean both the Travis CI and ReadTheDocs builds are OK.
- Suggested exercise: Clone/rename the repo, and turn on both Travis & ReadTheDocs automated builds in your version of it!
- Note: You will need to change travis-ci.org references to travis-ci.com

ISSUE/BUG TRACKING

- Version 1.2.3 has bug X!
- Windows builds broke
- How is the work going on refactoring module Y?
- Should we improve scaling by method Z or W?

Why did we decide to modify that loop in file F in git change Icfca5a?



Set required versions for GROMACS 2023
#4093 · created 2 days ago by Mark Abraham · 2023 winter planning meeting · 0 · updated 2 days ago

Require pybind 2.6 for gmxapi Python package extension module.
#4092 · created 1 week ago by M. Eric Irrgang · 2022-infrastructure-stable · Aug 31, 2021 · CMake · build system · gmxapi · 0 · updated 1 week ago

HtoD force transfer on critical path for heterogeneous DD cases 0 of 2 tasks completed
#4091 - created 2 weeks ago by Alan Gray · 2022-infrastructure-stable · CUDA · GPU acceleration · 1 · 1 · updated 5 days ago

Porting gromacs for using heffte
#4090 · created 2 weeks ago by Samir Shaikh · Jul 18, 2021 · 1 · updated 2 weeks ago

Erroneous SR energies of energygroup from mdrun -rerun in Gromacs >=2020
#4089 · created 2 weeks ago by FabKe91 · 2020 · mdrun · 0 · updated 2 weeks ago

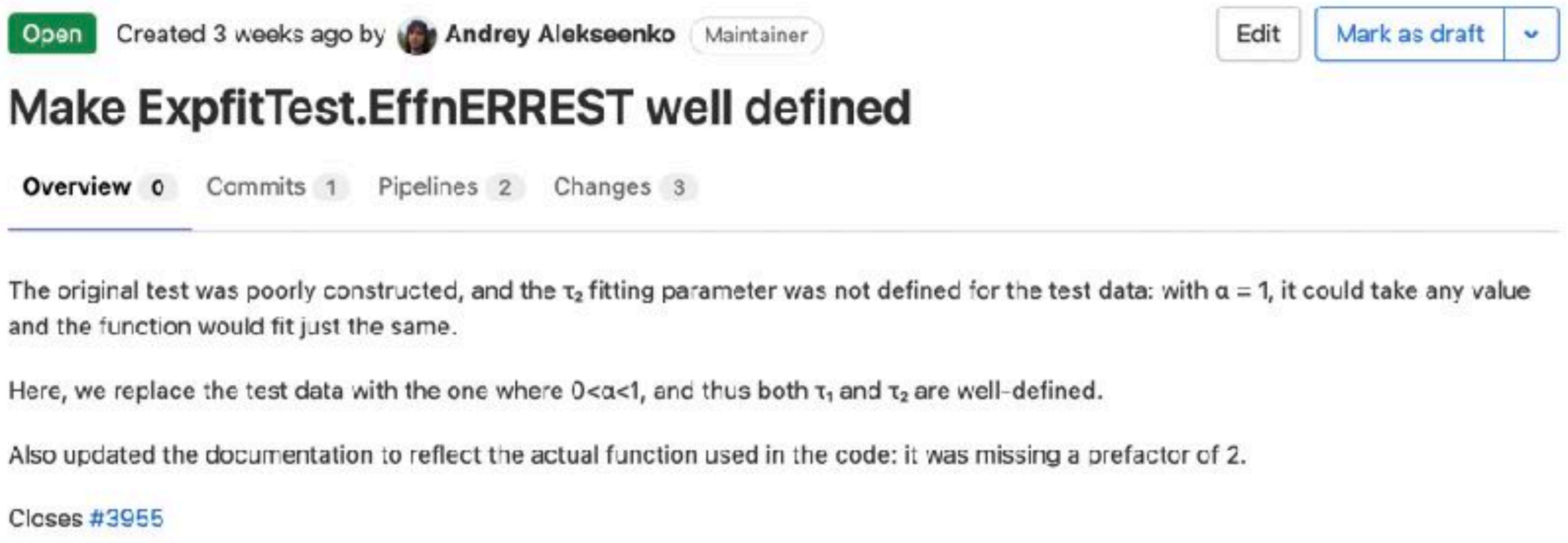
Add the time t as a variable in the transformation coordinate expression
#4088 · created 3 weeks ago by Oliver Fleetwood · 0 · updated 3 weeks ago

Several algorithms do not reflect changes in reference temperature due to simulated annealing
#4087 · created 3 weeks ago by Pascal Merz · 0 · updated 3 weeks ago

Update plugin::RestraintModule template per CPP core guidelines. 0 of 1 task completed
#4086 · created 1 month ago by M. Eric Irrgang · 2022-infrastructure-stable · documentation · gmxapi · gmxapi C++ · 1 · 0 · updated 1 month ago

Problem about 'gmx distance', realloc(): invalid next size
#4085 · created 1 month ago by 李代文 · 0 · updated 1 month ago

Improve declaration/documentation of NBLIB interaction types 0 of 1 task completed
#4084 · created 1 month ago by Mark Abraham · 2022 beta targets · nblib · nblib listed-forces · 0 · updated 1 month ago



Make ExpfitTest.EffnERREST well defined

The original test was poorly constructed, and the τ_2 fitting parameter was not defined for the test data: with $\alpha = 1$, it could take any value and the function would fit just the same.

Here, we replace the test data with the one where $0<\alpha<1$, and thus both τ_1 and τ_2 are well-defined.

Also updated the documentation to reflect the actual function used in the code: it was missing a prefactor of 2.

Closes #3955

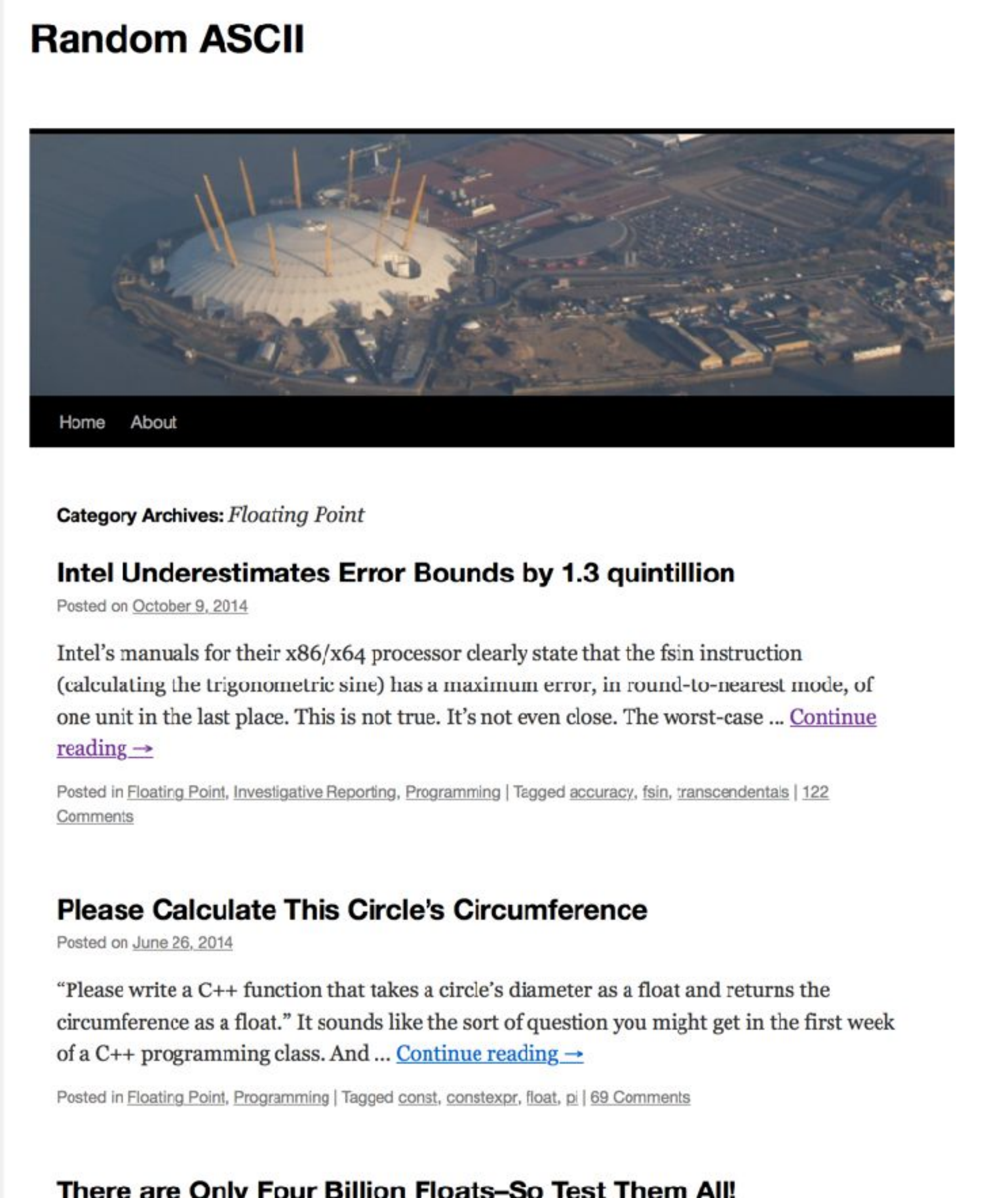
Automatic referencing in commit messages!

For IHPCSS/software-engineering, we use the integrated issue tracker in GitHub, but this too supports automated referencing e.g. for closing bugs.

OPENING PANDORA'S BOX - WHAT DO YOU KNOW ABOUT FLOATING-POINT?

- In the old days, scientists used double precision and pretended it was “infinite precision”
- GPUs, cache, memory bandwidth issues and half precision means you must understand FP
- http://randomascii.wordpress.com/category/floating-point/
  - Series of blog posts by Bruce Dawson about IEEE754 floating point
  - You should read this if you are working with scientific codes using floating-point!
  - Teaser - this might not always produce 0.0:

$$x = a * b - a * b$$



Random ASCII

Category Archives: *Floating Point*

Intel Underestimates Error Bounds by 1.3 quintillion

Posted on October 9, 2014

Intel’s manuals for their x86/x64 processor clearly state that the fsin instruction (calculating the trigonometric sine) has a maximum error, in round-to-nearest mode, of one unit in the last place. This is not true. It’s not even close. The worst-case … Continue reading →

Posted in Floating Point, Investigative Reporting, Programming | Tagged accuracy, fsin, transcendentals | 122 Comments

Please Calculate This Circle’s Circumference

Posted on June 26, 2014

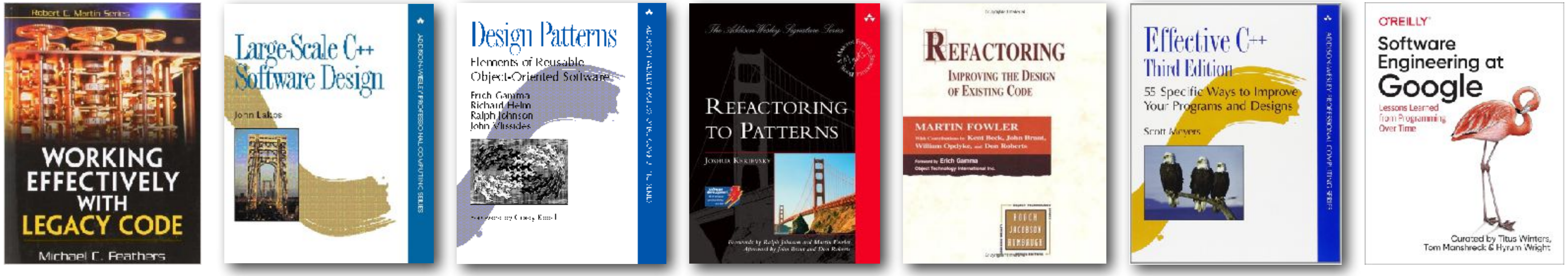
“Please write a C++ function that takes a circle’s diameter as a float and returns the circumference as a float.” It sounds like the sort of question you might get in the first week of a C++ programming class. And … Continue reading →

Posted in Floating Point, Programming | Tagged const, constexpr, float, pi | 69 Comments

There are Only Four Billion Floats–So Test Them All!

SOME RECOMMENDED ADDITIONAL READING

- Working effectively with legacy code [Michael Feathers]
- Large-scale C++ software design [John Lakos]
- Design Patterns - Elements of Reusable Object-oriented software [Gamma, Helm, Johnson, Vlissides] “Gang of four”
- Refactoring to Patterns [Joshua Kerievsky]
- Refactoring - improving the design of existing code [Martin Fowler]
- Effective C++ - 55 specific ways to improve your programs and design [Scott Meyers]
- Patterns for concurrent, parallel, and distributed systems [Douglas Schmidt]
- Clean Code [Robert Martin]
- Software Engineering at Google [Winters, Manshrecj, Wright]
- What everybody should know about floating-point math [David Goldberg]


Robert C. Martin Series
WORKING EFFECTIVELY WITH LEGACY CODE
Michael C. Feathers
Large-Scale C++ Software Design
John Lakos
Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
REFACTORING TO PATTERNS
Joshua Kerievsky
REFACTORING
IMPROVING THE DESIGN OF EXISTING CODE
MARTIN FOWLER
Foreword by Erich Gamma
Effective C++
Third Edition
55 Specific Ways to Improve Your Programs and Designs
Scott Meyers
O'REILLY
Software Engineering at Google
Lessons Learned from Programming Over Time
Curated by Titus Winters, Tom Manshreck & Hyrum Wright