

# HPC Software Engineering

Erik Lindahl

XSEDE/PRACE/RIKEN/SciNet HPC Summer School Kobe, Japan 2019



# Experiences 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 of development
  - Roughly 10-15 core developers
  - Another 15-20 active contributors
- Currently 3,076,420 lines of C++11 code (“C++11”)
- Over the years we have 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 repository: **CVS**

Build Chain:

**Automake/Autoconf/libtool**

Bug Tracking:

**Bugzilla**

Testing:



```
ii (r2 < rs2) {
/* check one on group exclusions */
if (!i_eg_excl[jgid]) {
if (r2 < rs2) {
if (ngrp[jgid] >= MAX_CG) {
```

*“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.”*

# Scientist mentality

- Trained in physics, chemistry, etc.
- Cares about their problem
- Cares about short-term deadlines
- New code = asset
- Writes more code than she reads

# Software engineer mentality

- Trained in CS/software
- Cares about their code
- Cares about long-term maintenance
- New code = liability
- Reads much more code than she writes

*Without proper software engineering, you are taking on a 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]*

# Why Open Source & Software Engineering Matter

The main point of open software is not cost, but that colleagues can check each other's code & assumptions and advance science by correcting flaws.

22 Aug 2018 in [Research & Technology](#)

DOI:10.1063/PT.6.1.20180822a

## 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

8  
COMMENTS

4.6K  
SHARES



< PREV NEXT >



*“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.”*

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



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

Please DO steal this and use it as a template for your own project!

When that is not advanced enough:

<http://www.gromacs.org>

<git://git.gromacs.org>

<http://gerrit.gromacs.org>

<http://redmine.gromacs.org>

<http://jenkins.gromacs.org>

... or browse GitHub for a huge number of other projects.

Check that your open source license matches, then copy-refine-improve.

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 Jan 1**?

671 commits

5,752 files changed

157,177 line insertions

1,622,410 line deletions\*

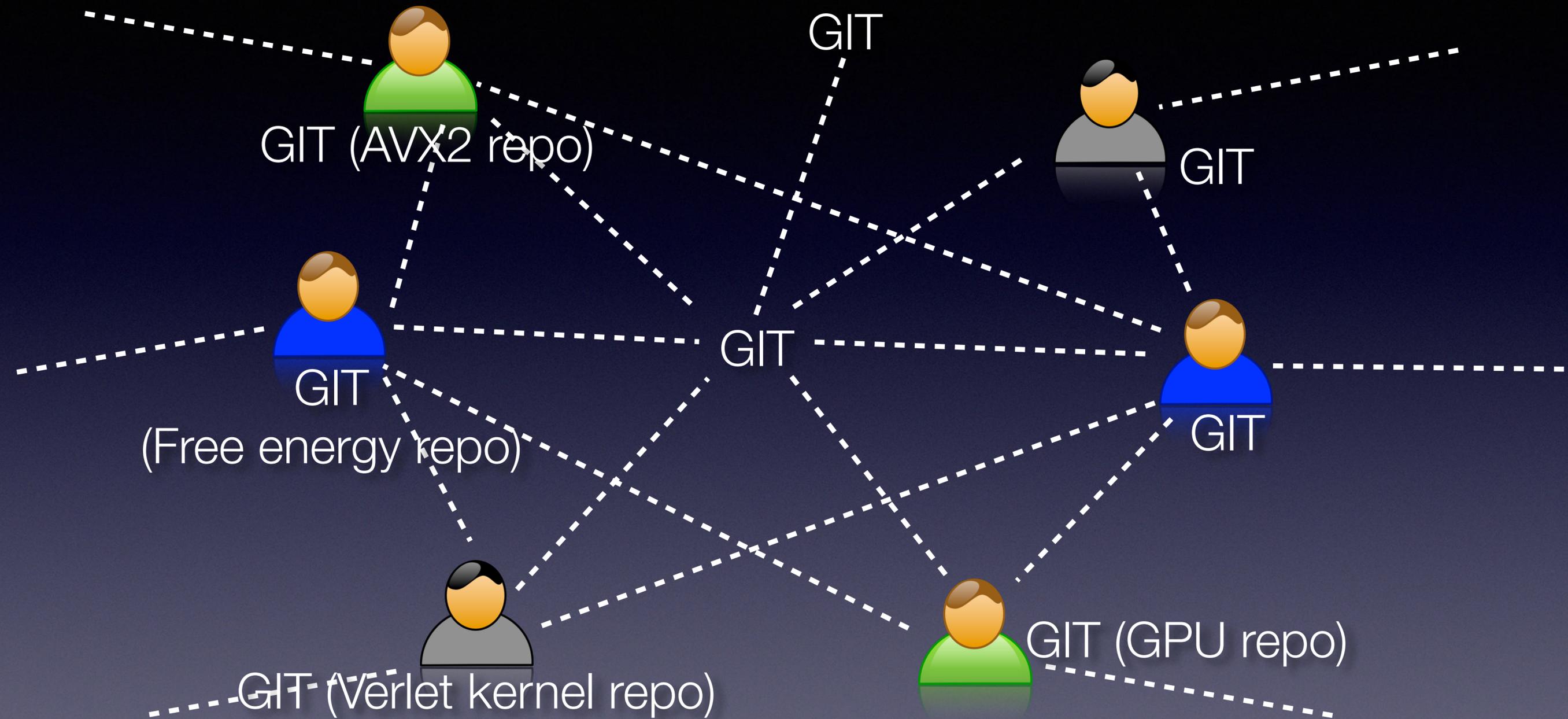
*\*Temporarily removed a bunch of kernels*

How would you start debugging if the new version crashes?

You have probably all seen this: Your program worked last week, but now there is something wrong

What if it crashes with “-O3”, but when you try to debug it works fine?

# Better source control: GIT



Use [gitlab.com](https://gitlab.com), or [github.com](https://github.com)!

New example software engineering project for you to play around with:  
<https://github.com/IHPCSS/software-engineering>

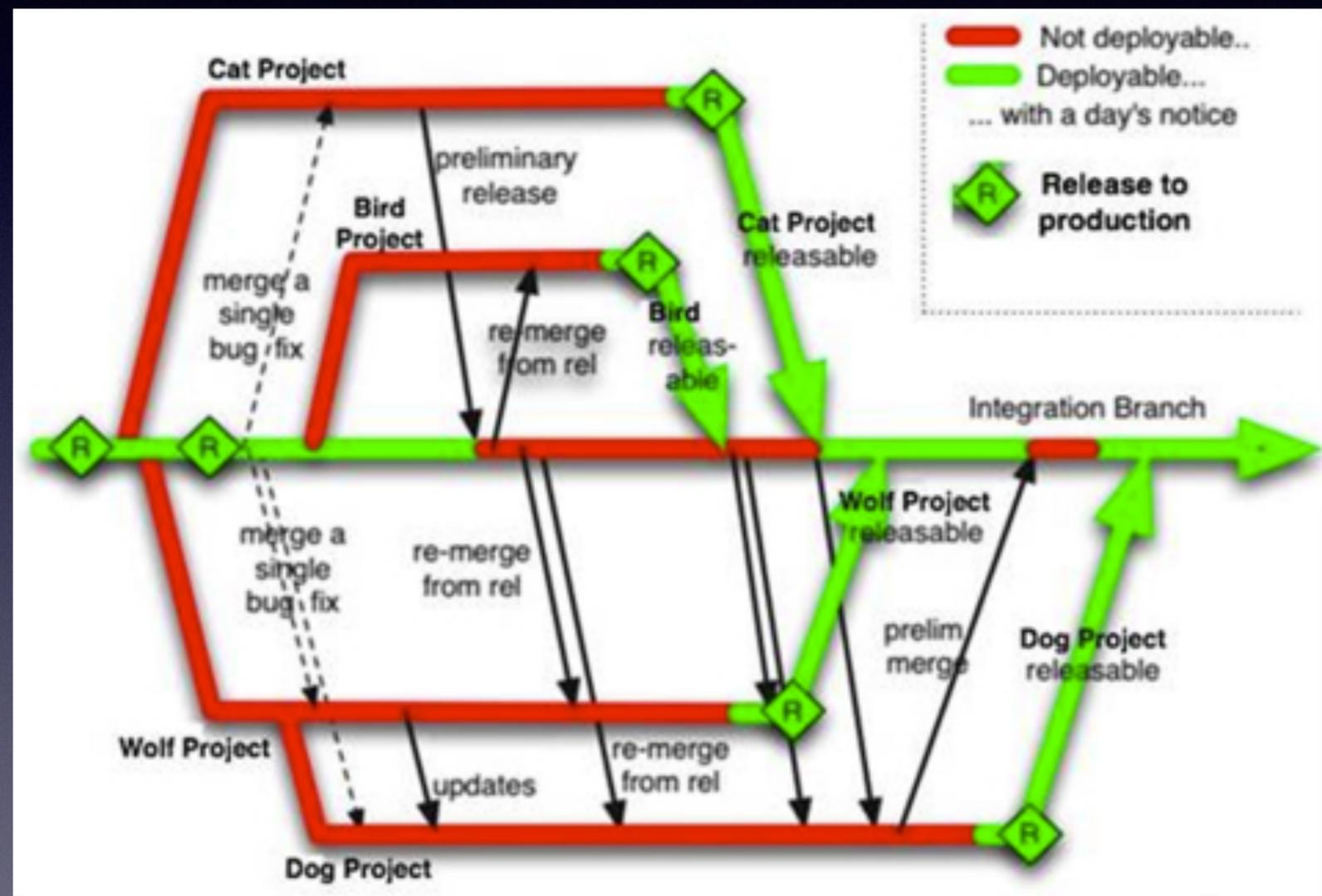
# What git will give you

- 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: Git is a VERY powerful tool, but the advanced features can be difficult to understand***

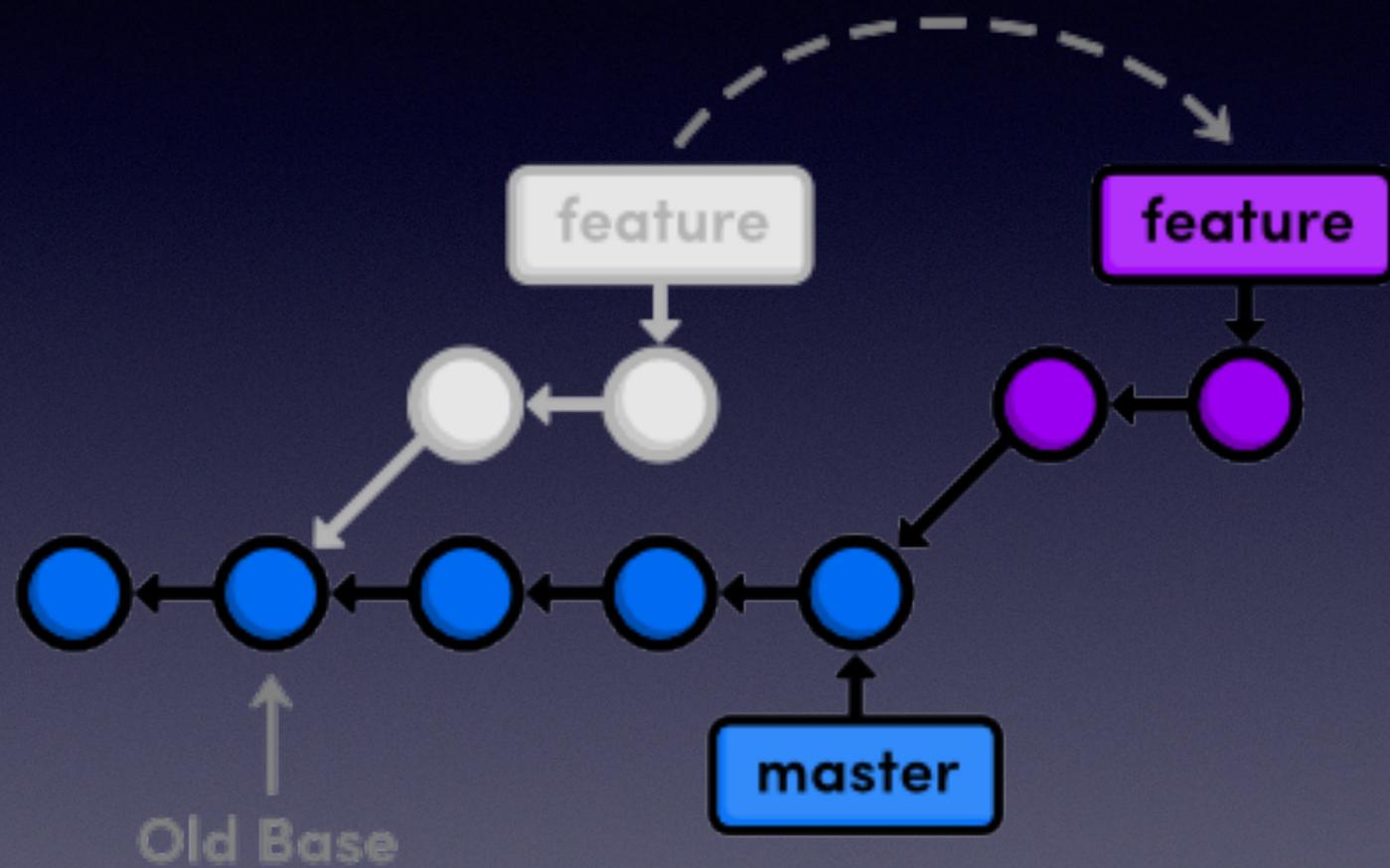


# One Possible Git Workflow: Multiple branches & merging



- Each feature is a new branch
- Think of the hybrid challenge:
  - Common base is the scalar version
  - Feature 1: MPI
  - Feature 2: OpenMP
  - Feature 3: OpenACC
- 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**

# Better approach: (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

# Good git commits are

- Small (think 10-100 lines, not 1000)
- Decomposed as far as possible
- Limited to address a single issue
- Well documented
- Tested to work

Have a look at some of the commits in the IHP.CSS-laplace repo!

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

**Initial CMake support, change to use C++ compiler** [Browse files](#)

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.

🔗 master

👤 Erik Lindahl committed on Jul 9, 2018 1 parent [962bdd9](#) commit [45d2c4833b8ccd9477df6581e1131b73e8e16d79](#)

📄 Showing **2 changed files** with 14 additions and 0 deletions. Unified Split

▼ 14 ■■■■■■ CMakeLists.txt

```
...    ...    @@ -0,0 +1,14 @@
1      + #
2      + # Example CMake file for John Urbanic's laplace
3      + # solver, now using a C++ compiler.
4      + #
5      +
6      + # Make sure we don't have a pre-historic CMake version
7      + cmake_minimum_required(VERSION 2.8)
8      +
9      + # We want a name
10     + project(Laplace)
11     +
12     + # Build the binary 'laplace' from the source file (now renamed to be a c++ source)
13     + add_executable(laplace laplace_serial.cpp)
14     +
```

▼ 0 ■■■■■■ laplace\_serial.c → laplace\_serial.cpp

File renamed without changes.

# Is your code portable?

Does your code compile on  
windows (MSVC)?

PGI Compilers? Pathscale?  
Blue Gene?

K computer (Fujitsu compilers)?

ARM? AArch64? With the ARM compiler? Clang? Gcc?

PowerPC (big endian)?

Google NativeClient?

OpenPower (little endian?)

# What is a build chain?

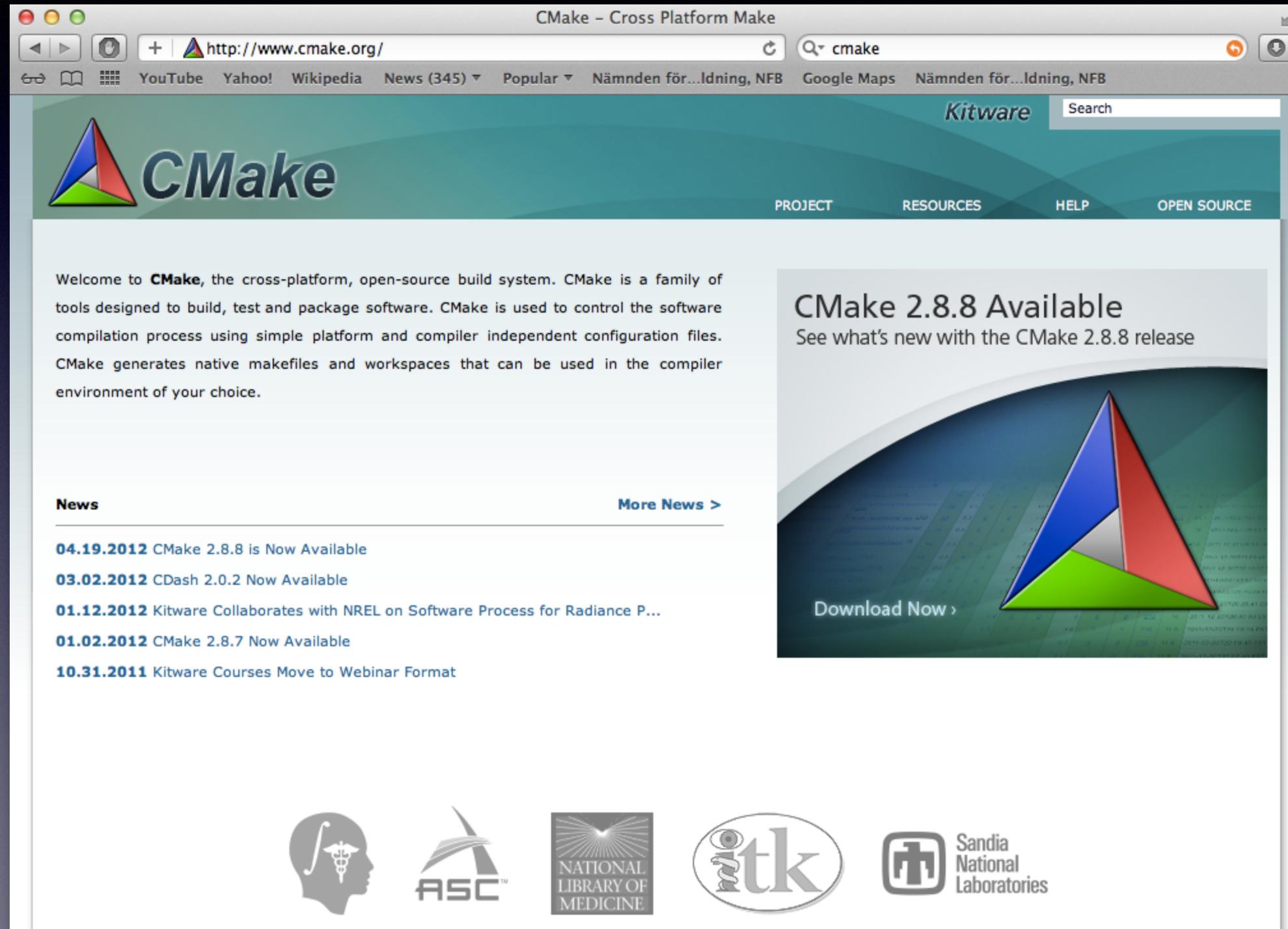
## The typical user progression:

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

# Configuration

- “Where is the X11 library? MKL? LibXML?”
- “What version is 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?”

# CMake: Cross-platform replacement for Autoconf, Automake, Libtool (instead of ./configure; make; make install)



The image shows a screenshot of the CMake website homepage as viewed in a browser window. The browser's address bar shows the URL `http://www.cmake.org/` and the search bar contains the text "cmake". The website header features the CMake logo (a 3D pyramid) and the name "CMake" in a large font. To the right of the logo is the Kitware logo and a search input field. Below the header is a navigation menu with links for "PROJECT", "RESOURCES", "HELP", and "OPEN SOURCE".

The main content area includes a welcome message: "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."

Below the welcome message is a "News" section with a "More News >" link. The news items listed are:

- 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

On the right side of the page, there is a large promotional banner for "CMake 2.8.8 Available". The banner features the CMake logo and the text "See what's new with the CMake 2.8.8 release". At the bottom of the banner is a "Download Now >" link.

At the bottom of the page, there is a row of logos for partner organizations: a head profile with a caduceus, ASC (American Society of Civil Engineers), the National Library of Medicine, itk (Insight Toolkit), and Sandia National Laboratories.

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

- CheckCCompilerFlag.cmake
- CheckCXXCompilerFlag.cmake
- cmake\_uninstall.cmake.in
- FindEXTRA.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
- gmxGCC4403BugWorkaround.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 regression tests 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 IHP.CSS/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 for the IHP.CSS Laplace code

```
#
# Example CMake file for the IHP.CSS 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)
```

# 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
```

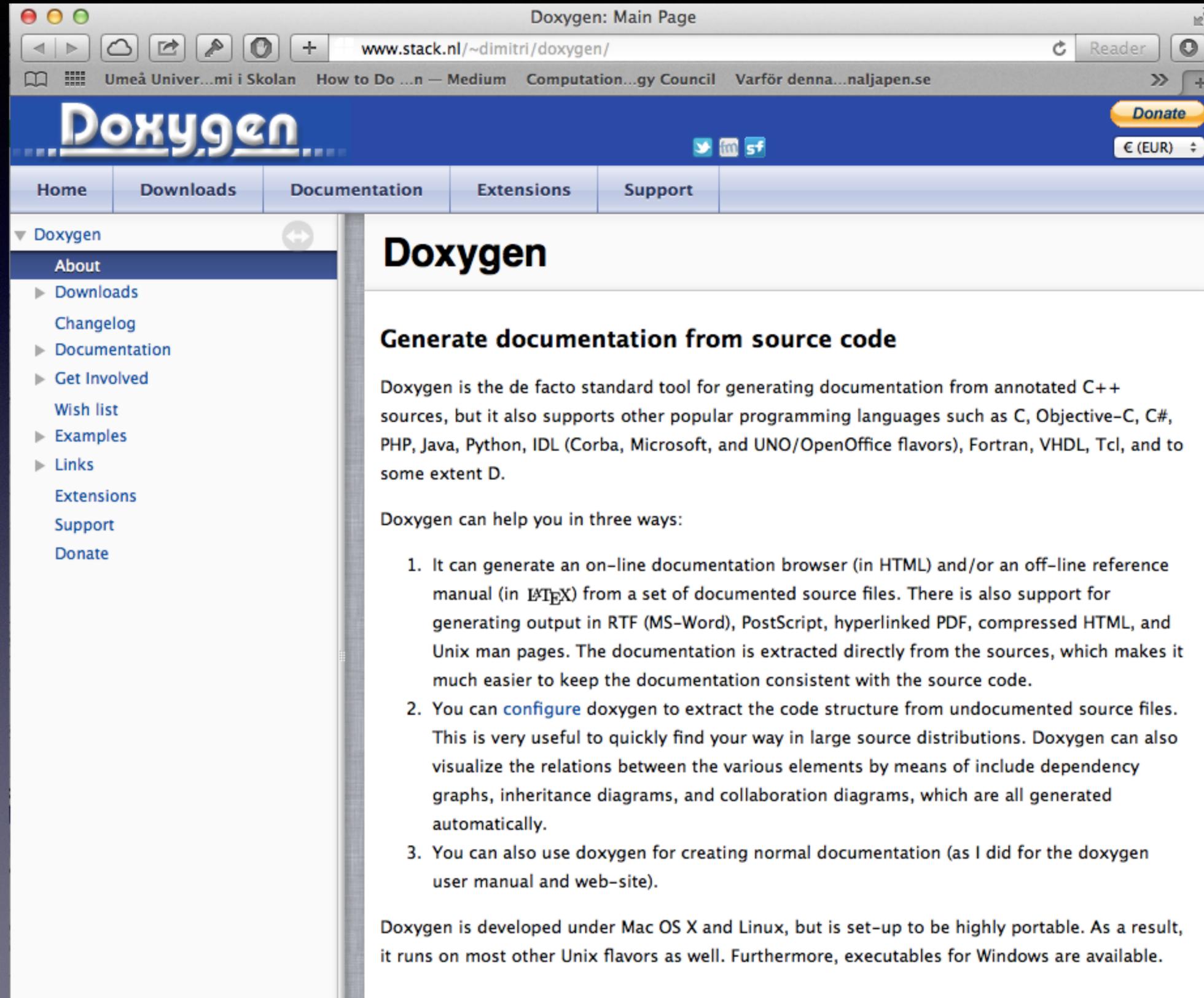
```
$ ~> cd build-openacc
```

```
$ build-openacc> cmake -DOPENACC=0N ../path/to/source/directory
```

Living with your code for ~~years~~ decades:

Documentation

Direct source code documentation should stay in the source!



The image shows a screenshot of a web browser displaying the Doxygen website. The browser's address bar shows the URL `www.stack.nl/~dimitri/doxygen/`. The website has a blue header with the "Doxygen" logo and a navigation menu with links for Home, Downloads, Documentation, Extensions, and Support. A sidebar on the left contains a list of links including About, Downloads, Changelog, Documentation, Get Involved, Wish list, Examples, Links, Extensions, Support, and Donate. The main content area features a large heading "Doxygen" and a sub-heading "Generate documentation from source code". Below this, there is a paragraph describing Doxygen as a tool for generating documentation from annotated C++ sources, followed by a list of three ways it can help users. At the bottom, a paragraph states that Doxygen is developed under Mac OS X and Linux and is highly portable.

Doxygen: Main Page

`www.stack.nl/~dimitri/doxygen/`

Umeå Univer...mi i Skolan How to Do ...n — Medium Computation...gy Council Varför denna...naljapen.se

Doxygen

Donate

€ (EUR)

Home Downloads Documentation Extensions Support

▼ Doxygen

- About
- ▶ Downloads
- Changelog
- ▶ Documentation
- ▶ Get Involved
- Wish list
- ▶ Examples
- ▶ Links
- Extensions
- Support
- Donate

## 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  $\text{\LaTeX}$ ) 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 - our random module:

```
/*! \brief ThreeFry2x64 random engine with 20 interactions.
 *
 * \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, internalCounterBits>(key0, domain) {}

    /*! \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) {}
};
```

The best comments don't explain what your code does, they explain WHY you do it this way!

The IHPCSS/software-engineering example comes with full Doxygen integration - but we have not yet had time to document the code! CMake finds "doxygen" automatically so you can do "make doxygen"

## Gromacs 2019

<a href="#">Main Page</a>	<a href="#">Modules</a>	<a href="#">Other Docs</a>	<a href="#">Namespaces</a>	<b><a href="#">Classes</a></b>	<a href="#">Files</a>	<a href="#">Examples</a>
<a href="#">Class List</a>	<a href="#">Class Index</a>	<a href="#">Class Hierarchy</a>	<a href="#">Class Members</a>			
<a href="#">gmx</a>	<a href="#">ThreeFry2x64</a>					

### 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 interactions.

#### 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 counters.

### 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 >

# High level non-source-code documentation: SPHINX (from Python)

```
.. _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
^^^^^^^^^^^^^^^^^^^^
```

software-engineering  
latest

Search docs

The Laplace Equation  
Software Engineering



Join us for the very first regional African Pycon 🌍 this August in Accra, Ghana

Community Ad

Docs » IHP.CSS Laplace Solver [Edit on GitHub](#)

## IHP.CSS 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](#)

Next ➔

© 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 IHP.CSS-laplace.  
Check out the docs folder, and if you have sphinx/latex installed you can type “make sphinx-html” or “make sphinx-pdf”.

... and we have integrated it with [readthedocs.org](http://readthedocs.org)! Any time a new change is pushed to the github repo, documentation is built automatically at <http://software-engineering.readthedocs.org>

# 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++

# Languages?

- “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>

# The Case for 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

Negative: It is a VERY complex language to master

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 m1(&m);
    ...
}
```

Surprise: C++ can be (much) faster than FORTRAN or C!

## C/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);
```

## C++11

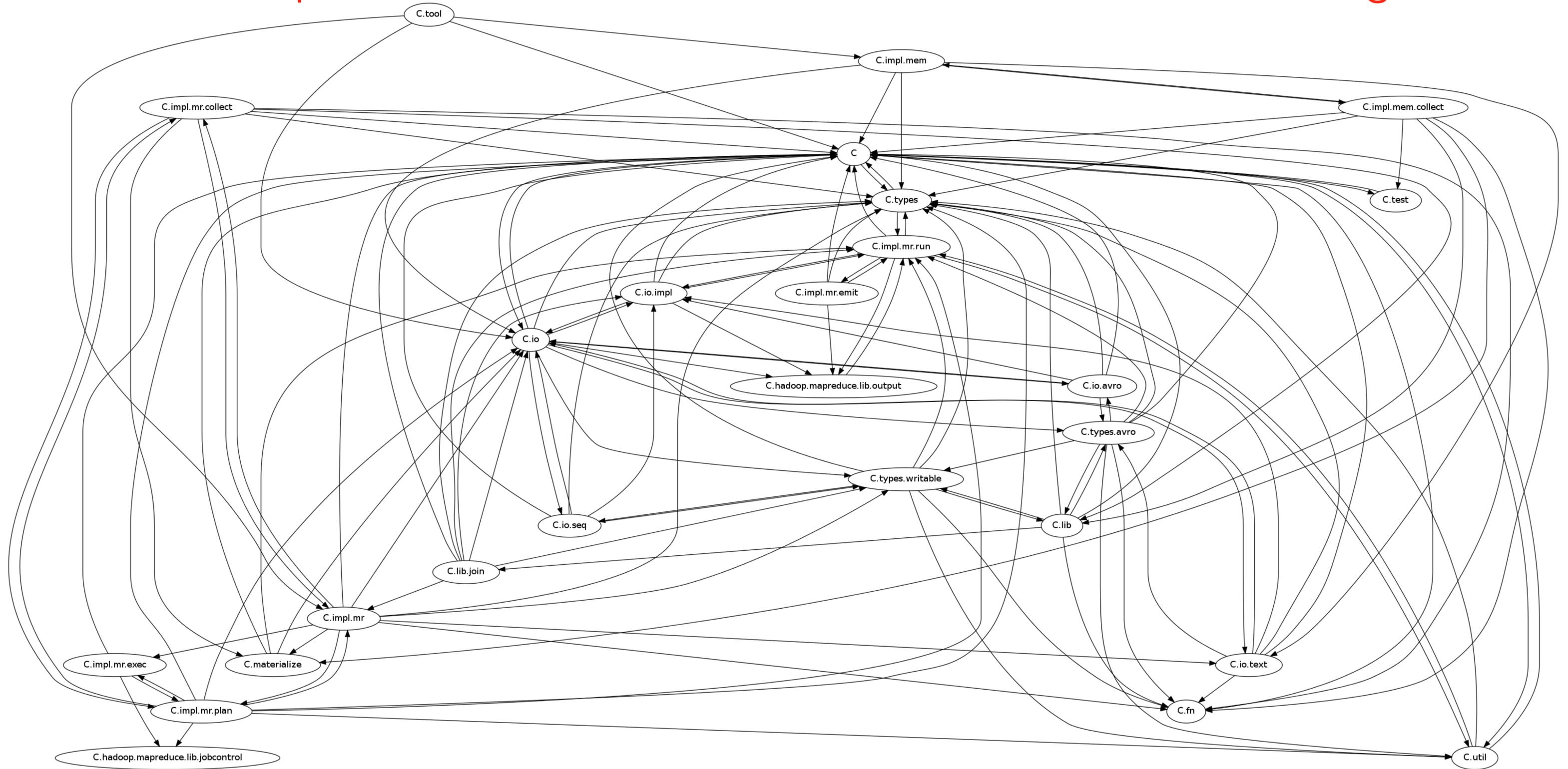
```
template <int choiceA, int choice B>
int
myFunc(obj_t obj)
{
    for(int i=0;i<obj.N;i++)
    {
        if(choiceA==1)
        {
            if(choiceB==1)
            {
                kernelcode1;
            }
            else if(choiceB==2)
            {
                kernelcode2;
        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

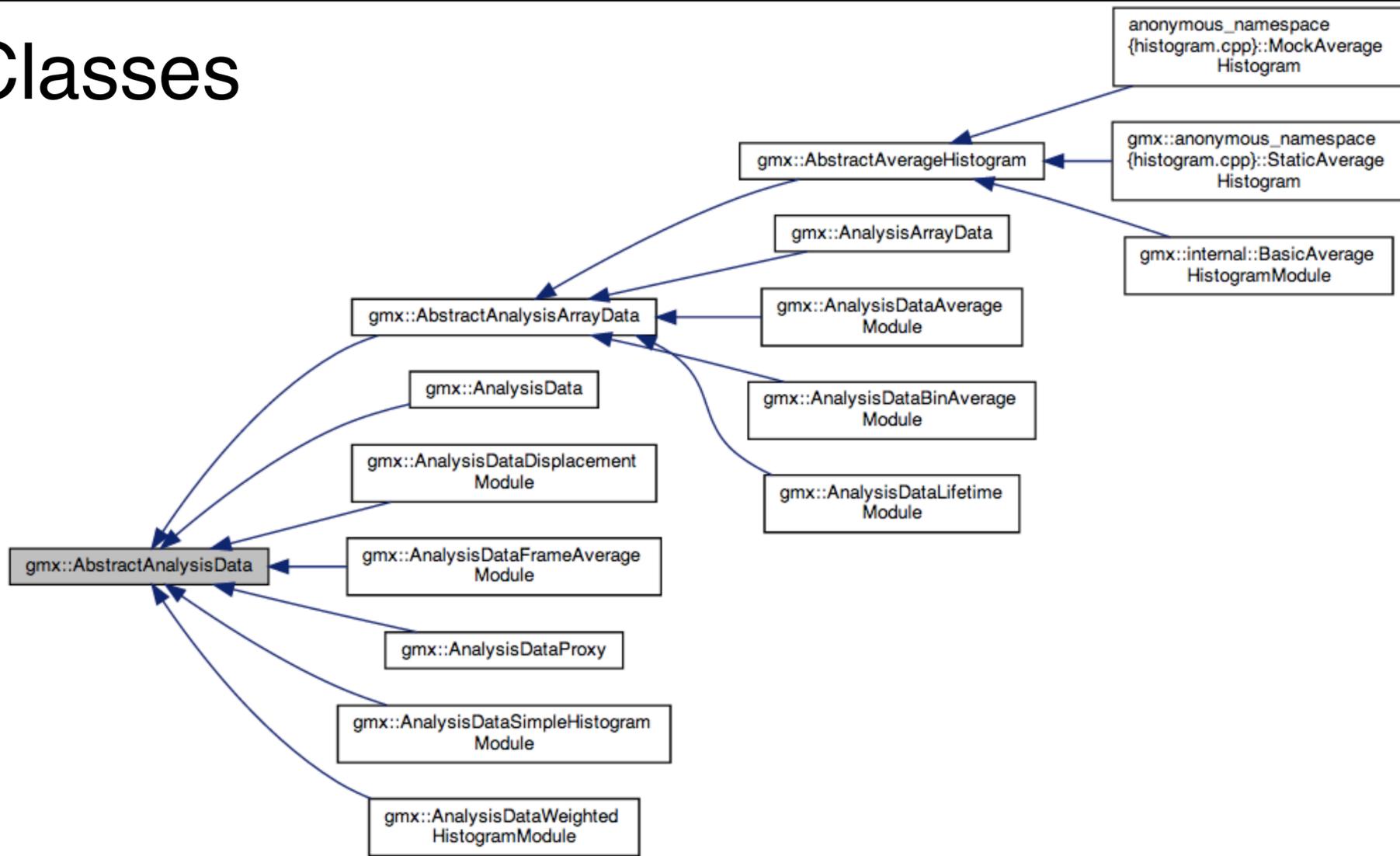
# Circular dependencies are bad. If a test fails, where is the bug here?



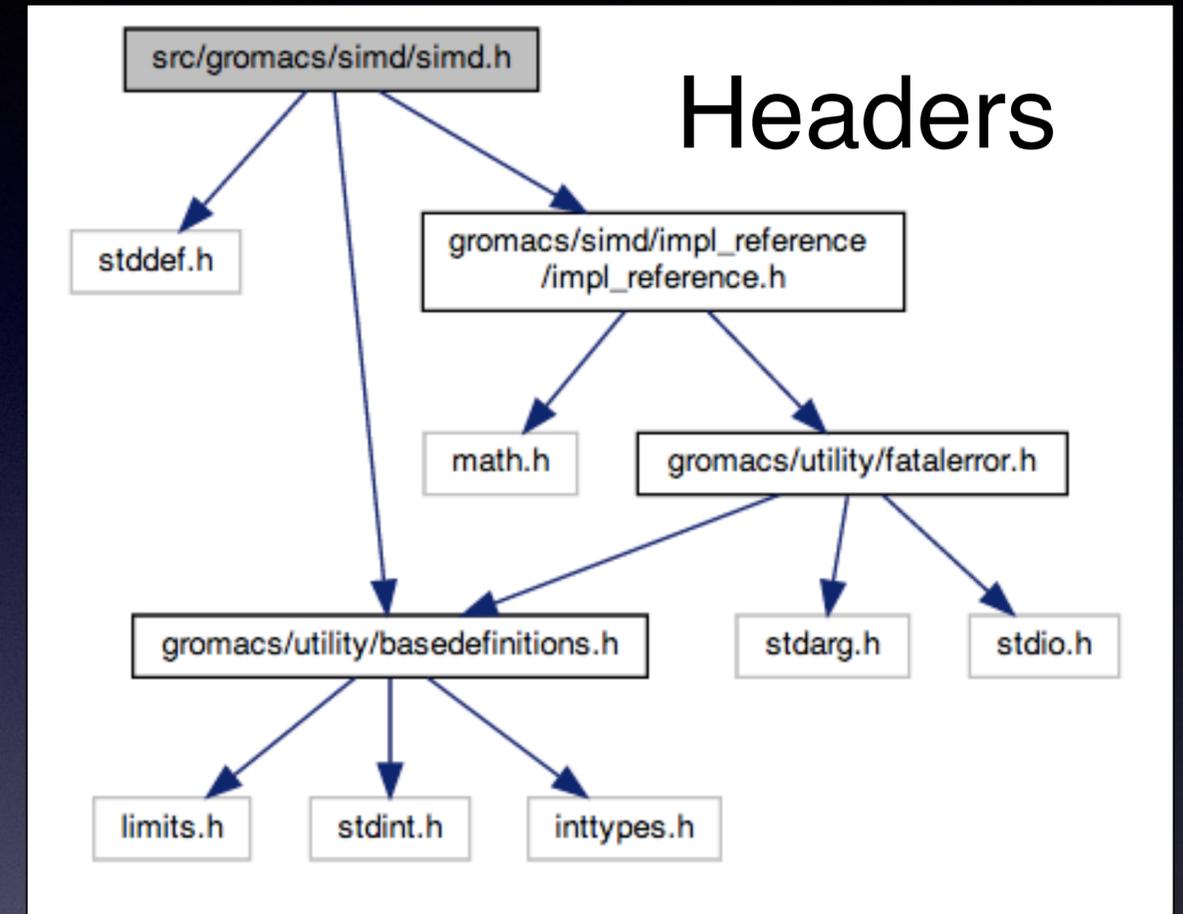
*“It has been discovered that C++ provides a remarkable facility for concealing the trivial details of a program - such as where its bugs are.” (David Keppel)*

# Modularization: Just say 'no' to circular dependencies

## Classes



## Headers



This is hard, but Doxygen helps you detect it!

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



# Aggressive unit testing: "Trust, but verify"

[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](#),  
[sbe...@google.com](#),  
[billydon...@google.com](#)  
[8 committers](#)

## Featured

### Downloads

[gtest-1.7.0.zip](#)

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)

If you know of a project that's using Google Test and want it to be listed here, please let [googletestframework@googlegroups.com](mailto:googletestframework@googlegroups.com) know.

## Google Test-related open source projects

[Google Test UI](#) is test runner that runs your test binary, allows you to track its progress via a progress bar, and displays a list of test failures. Clicking on one shows failure text. Google Test UI is written in C#.

# Example Gromacs unit tests:

The idea is that you should test *everything*

```
185 TEST_P(FFTTest1D, Real)
186 {
187     const int rx = GetParam();
188     const int cx = (rx/2+1);
189     ASSERT_LE(cx*2, static_cast<int>(sizeof(inputdata)/sizeof(real)));
190
191     in_ = std::vector<real>(inputdata, inputdata+cx*2);
192     out_ = std::vector<real>(cx*2);
193     real* in = &in_[0];
194     real* out = &out_[0];
195
196     gmx_fft_init_1d_real(&fft_, rx, flags_);
197
198     gmx_fft_1d_real(fft_, GMX_FFT_REAL_TO_COMPLEX, in, out);
199     checker_.checkSequenceArray(cx*2, out, "forward");
200     gmx_fft_1d_real(fft_, GMX_FFT_COMPLEX_TO_REAL, in, out);
201     checker_.checkSequenceArray(rx, out, "backward");
202 }
```

```
204 TEST_F(SimdFloatingpointTest, gmxSimdGetMantissaR)
205 {
206     GMX_EXPECT_SIMD_REAL_EQ(setSimdRealFrom3R(1.219097320577810839026256,
207                                               1.166738027848349235071623,
208                                               1.168904015004464724825084), gmx_simd_get_mantissa_r(rSimd_Exp));
209     #if (defined GMX_SIMD_HAVE_DOUBLE) && (defined GMX_DOUBLE)
210     GMX_EXPECT_SIMD_REAL_EQ(setSimdRealFrom3R(1.241261238952345623563251,
211                                               1.047294723759123852359232,
212                                               1.856066204750275957395734), gmx_simd_get_mantissa_r(rSimd_ExpDouble));
213     #endif
214 }
215
216 TEST_F(SimdFloatingpointTest, gmxSimdSetExponentR)
217 {
218     gmx_simd_real_t x0 = setSimdRealFrom3R(0.5, 11.5, 99.5);
219     gmx_simd_real_t x1 = setSimdRealFrom3R(-0.5, -11.5, -99.5);
220
221     GMX_EXPECT_SIMD_REAL_EQ(setSimdRealFrom3R(pow(2.0, 60.0), pow(2.0, -41.0), pow(2.0, 54.0)),
222                             gmx_simd_set_exponent_r(setSimdRealFrom3R(60.0, -41.0, 54.0)));
223     #if (defined GMX_SIMD_HAVE_DOUBLE) && (defined GMX_DOUBLE)
224     GMX_EXPECT_SIMD_REAL_EQ(setSimdRealFrom3R(pow(2.0, 587.0), pow(2.0, -462.0), pow(2.0, 672.0)),
225                             gmx_simd_set_exponent_r(setSimdRealFrom3R(587.0, -462.0, 672.0)));
226     #endif
227     /* Rounding mode in gmx_simd_set_exponent_r() must be consistent with gmx_simd_round_r() */
228     GMX_EXPECT_SIMD_REAL_EQ(gmx_simd_set_exponent_r(gmx_simd_round_r(x0)), gmx_simd_set_exponent_r(x0));
229     GMX_EXPECT_SIMD_REAL_EQ(gmx_simd_set_exponent_r(gmx_simd_round_r(x1)), gmx_simd_set_exponent_r(x1));
230 }
231 }
```

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.

Good unit tests should isolate bugs to *tiny* parts of your code  
In C++, each method in a class should ideally have exhaustive unit tests

```
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.

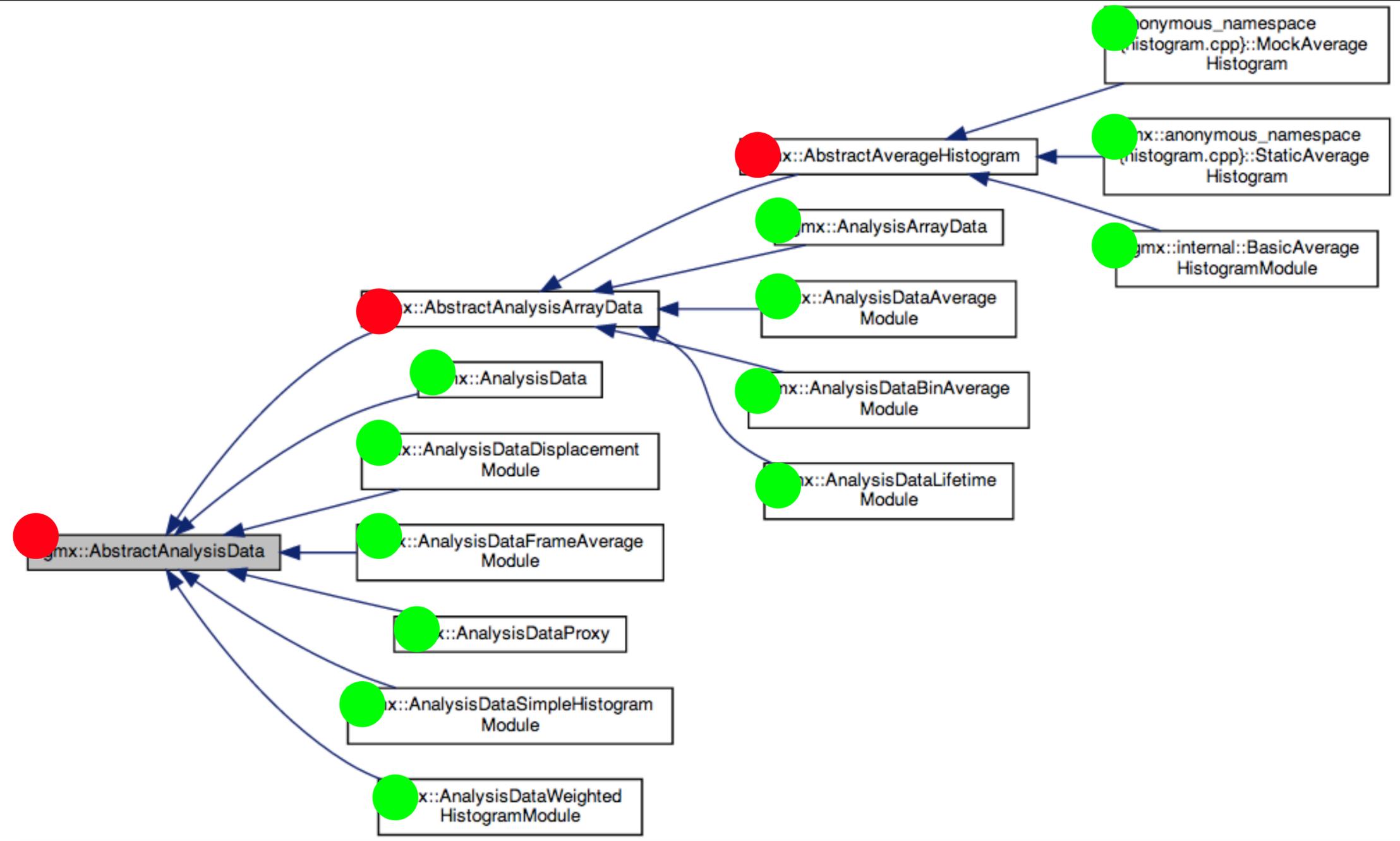
Why? Because we found that libstdc++ and libcxx do not use the same algorithm, so code will not produce the same results. We need to use our own algorithm - make sure it keeps working.

No need to ask: Of course we have integrated GoogleTest support into the IHP.CSS/software-engineering repo - but I have not had time to write the actual tests yet. 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?



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 code makes it into Gromacs

*Who is allowed to write to your code repository?*

Problems if you think some  
less talented developers  
might submit buggy code

Such as this  
one



# Gerrit Code Review

The screenshot shows the Gerrit Code Review website in a browser window. The browser address bar displays "http://code.google.com/p/gerrit/". The page header includes the Gerrit logo and a search bar. The main navigation menu contains "Project Home", "Downloads", "Issues", and "Source". The "Project Home" section is active, showing a "Summary" and "People" tab. The "Project Information" section includes a description: "Web based code review and project management for Git based projects." The "Objective" section states: "Gerrit is a web based code review system, facilitating online code reviews for projects using the Git version control system. Gerrit makes reviews easier by showing changes in a side-by-side display, and allowing inline comments to be added by any reviewer. Gerrit simplifies Git based project maintainership by permitting any authorized user to submit changes to the master Git repository, rather than requiring all approved changes to be merged in by hand by the project maintainer. This functionality enables a more centralized usage of Git." The "News" section lists several updates from 2012, including releases of versions 2.2.2.2, 2.3.1, 2.4.2, 2.4.1, and 2.4 final, as well as announcements about MultiMaster support, scaling tips, and a hackathon report. The left sidebar contains sections for "Project Information" (starring by 1374 users, code license Apache License 2.0, labels git, codereview, Google, jgit, VCS), "Members" (listing several email addresses and 21 contributors), and "Downloads" (gerrit-2.4.2.war).

Nobody can commit directly to our central Git repo anymore  
... which means we can allow anybody to commit in gerrit!

status:open | gerrit.gromacs Code Review

gerrit.gromacs.org/#/q/status:open,n,z

status:open

Search Register Sign

Open Merged Abandoned

Search for status:open

Subject	Status	Owner	Project	Branch	Updated	CR	V
New quote		Mark Abraham	gromacs	release-5-0	3:30 PM	+1	✓
Improve module dependency graph layout		Teemu Murtola	gromacs	master (doxygen)	2:42 PM	✓	✓
Module dependency cycle checker for 'doc-check'		Teemu Murtola	gromacs	master (doxygen)	2:41 PM	✓	✓
Updated reference with fixed potential-shift dispcorr		Erik Lindahl	regressiontests	release-4-6	2:11 PM	✓	✓
Fixed shift and switch modifiers, particularly for free-energy		Berk Hess	gromacs	release-4-6	2:10 PM	+1	✓
Remove mdrun -seppot		Mark Abraham	gromacs	master	2:09 PM	+1	✓
Move atomprop.* to topology/	Submitted, Merge Pending	Teemu Murtola	gromacs	master (legacyheaders)	1:48 PM	✓	✓
Update tests using v-rescale		Mark Abraham	regressiontests	release-5-0	1:25 PM		
Use RNG correctly for v-rescale thermostat		Mark Abraham	gromacs	release-5-0	1:25 PM	✓	
Move some verlet headers to mdlib		Roland Schulz	gromacs	master	11:36 AM		✗
RFC: Used IWYU to partially clean up includes		Roland Schulz	gromacs	master	10:49 AM		✓
Check to ensure not reading past end of file.		Magnus Lundborg	tng	master	9:47 AM		
Fixed wrong journal reference in manual		David van der Spoel	gromacs	master	9:44 AM	✓	✓
Add StringFormatter and formatAndJoin to stringutil		Mark Abraham	gromacs	master (g-tune-pme-reform)	5:57 AM	✓	✓
RFC: Make all include paths same format		Roland Schulz	gromacs	master	5:33 AM		
Replace all command line parsing with Options		Teemu Murtola	gromacs	master (cmdline)	Jun 1	✓	
Move mtop_util.* and topsort.* to topology/		Teemu Murtola	gromacs	master (legacyheaders)	Jun 1	✓	
Remove more uses of typedefs.h		Teemu Murtola	gromacs	master (legacyheaders)	Jun 1	✓	
Enable 4-letter residue names in PDB output		Erik Lindahl	gromacs	release-5-0	Jun 1	-1	
Updated C-/N-terminal partial charges in Amber03.ff.		Rossen Apostolov	gromacs	release-4-6	Jun 1	✓	
Convert repl_ex.c to C++		Mark Abraham	gromacs	master (c++)	Jun 1	✓	
[RFC] Framework for analyzing energy files.		David van der Spoel	gromacs	master	May 31		
Improve FileNameOption error handling		Teemu Murtola	gromacs	master (cmdline)	May 31		
Fix ref error in pull		Roland Schulz	gromacs	release-4-6	May 31	-1	
Issue a warning for using gmx_rms -prev with large trajectories.		Rossen Apostolov	gromacs	release-4-6	May 31	✓	

Multiple patches in-flight  
 Gerrit/git do dependency tracking, patches can be rebased onto others by hitting a rebase button, or even edited on-the-fly in the window

Extensive comments on code during review

Roland has approved Mark's patch. Anybody can add comments. When two trusted developers say OK, the patch is committed.

Change #, SHA-1, tr:id or owner:email

Open Merged Abandoned

Change-Id: I1fb8eddb7c8b029dc3686be80f3f083108fc28c

Owner: Mark Abraham

Project: gromacs

Branch: release-5-0

Topic:

Uploaded: May 25, 2014 9:28 PM

Updated: Jun 2, 2014 1:25 PM

Submit Type: Rebase if Necessary

Status: Review in Progress

Commit Message: [Permalink](#)

**Use RNG correctly for v-rescale thermostat**

Two integers were passed in the wrong order. I suspect from the construction of the RNG that the only effect of this is to permit a rare re-use of a random number in a different RNG stream (i.e. no effect in practice).

Change-Id: I1fb8eddb7c8b029dc3686be80f3f083108fc28c

Reviewer	Code-Review	Verified
Mark Abraham		
Roland Schulz	✓	

- Need Verified

Dependencies

Reference Version: Base

Patch Set 1: 7aff98680e3bfd29b7a3786799606bad068768f6 (github)

Patch Set 2: 9817980d60eab742f9d3e7468d210de82ac80bcb (github)

Author: Mark Abraham <mark.j.abraham@gmail.com> May 25, 2014 9:38 PM

Committer: Mark Abraham <mark.j.abraham@gmail.com> Jun 2, 2014 9:21 AM

Parent(s): ab9ac88415a51482e2e99a9e6e8a44f42d365805 Add quote on the kT-kj/mol conversion factor

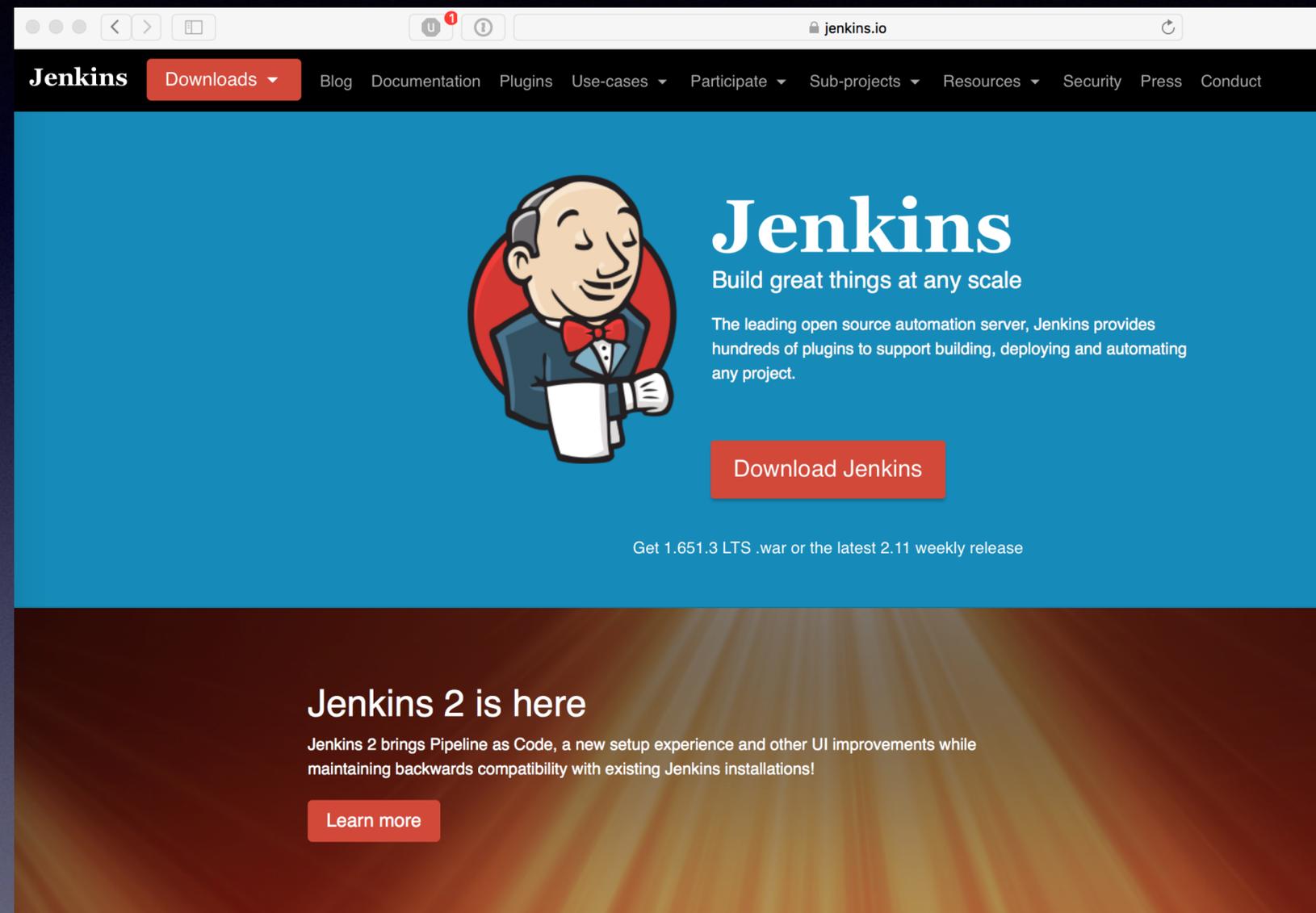
Download: checkout | pull | cherry-pick | patch | Anonymous HTTP | git fetch https://gerrit.gromacs.org/gromacs refs/changes/05/3505/2 && git checkout FETCH\_HEAD

# Maintaining quality & avoiding breaking stuff

How do I make sure that \*I\* don't make mistakes?

# Jenkins Continuous Integration

<https://jenkins.io>



The screenshot shows the Jenkins website homepage. At the top, there is a navigation bar with the Jenkins logo and a 'Downloads' dropdown menu. Below the navigation bar, the main content area features a blue background with the Jenkins logo (a cartoon character in a tuxedo) and the text 'Jenkins Build great things at any scale'. A 'Download Jenkins' button is prominently displayed. Below this, there is a section for 'Jenkins 2 is here' with a 'Learn more' button.

Every single commit is tested automatically on our build farm, including both builds and regression tests.

Results are integrated into the gerrit review

- Catches Cmake build errors
- Catches Google test unit test failures

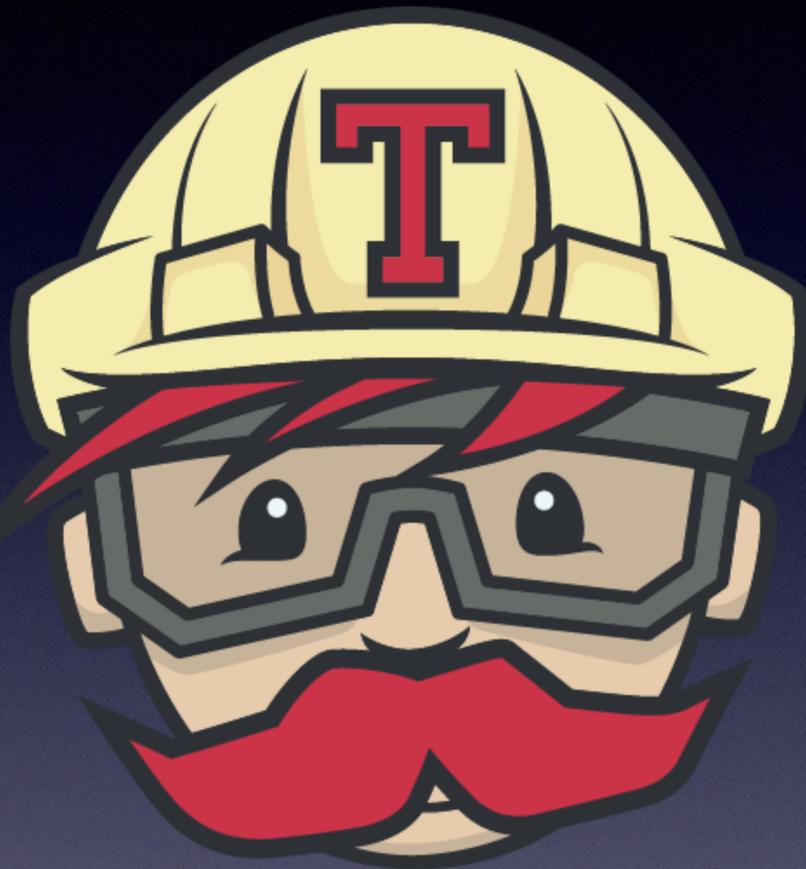
# GROMACS CI tests for *every* commit

- Unit Tests: Do modules reproduce reference values?
- Integration tests: Does a normal full run work?
- Regression tests: Are previous simulation results identical?
- Physical validation tests: Do we reproduce statistical ensemble fluctuations?
- 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
- Uncrustify: Proper code formatting, no tabs, brace standards?
- Doxygen: All classes/methods/arguments/variables documented?
- Coming: Performance regression testing



# Travis CI

<https://travis-ci.org>



- Jenkins 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
- Of course this is enables 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!

# Redmine issue tracking

#	Project	Tracker	Status	Priority	Subject	Assignee	Updated	Target version
962	Gromacs	Bug	New	Normal	segv/hang of EM with foreign lambda's	Berk Hess	06/25/2012 11:30 am	4.5.6
959	Gromacs	Bug	New	Normal	Issue with nonhomogeneous boundaries and domain decomposition		06/20/2012 03:22 am	4.6
958	Gromacs	Bug	New	Normal	MPI on Windows		06/27/2012 07:01 am	
957	Gromacs	Bug	New	Normal	Spurious parameters for Argon in OPLS-AA and Charmm27		06/15/2012 01:18 am	
956	Gromacs	Bug	New	Normal	Unit cell expands in X/Y during semisotropic simulation of an octane slab with 8 threads and -pd but not with 2 threads or when using -dd on 8		06/25/2012 04:19 pm	

- 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?

Automatic referencing in commit messages!

```
Closes #926 - Raw assembly code has been removed.  
Refs #923 - Old kernels removed, new will be added shortly.  
Fixes #914 - Cmake now does architecture-specific optimization.  
Fixes #912, #913  
Fixes #857 - We detect rdtscp support with CPUID and use it if possible.  
Fixes #750  
Closes #537, #574 - Altivec is now deprecated.  
  
Change-Id: Icfca5a940762f8d82ae67b59c65b2d2ac683256d
```

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

<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 return  $x = 0$ :

$$x = a \cdot b - a \cdot b$$

More worthwhile reading:

“What every computer scientist should  
know about floating-point arithmetic”  
[David Goldberg]

## Random ASCII



[Home](#) [About](#)

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](#)

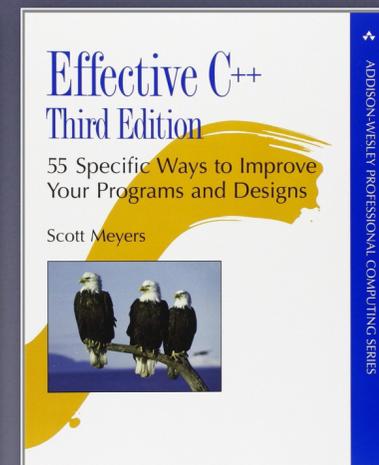
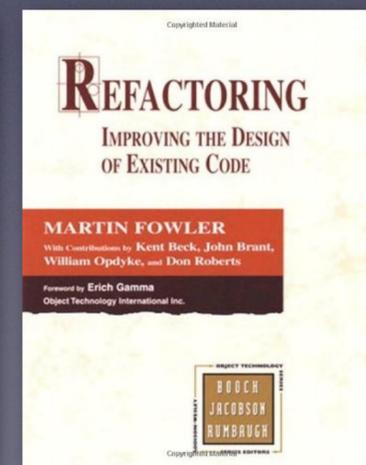
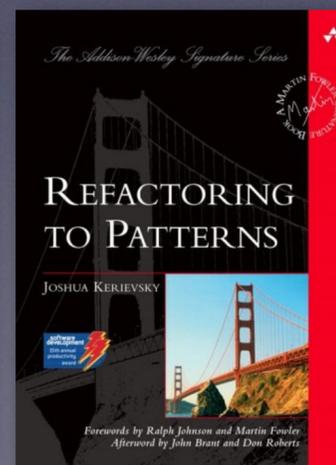
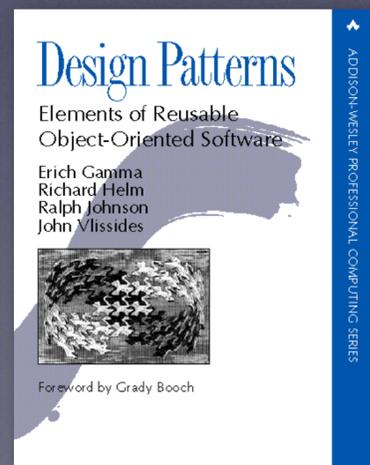
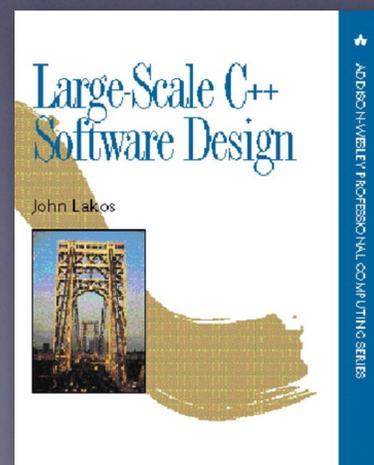
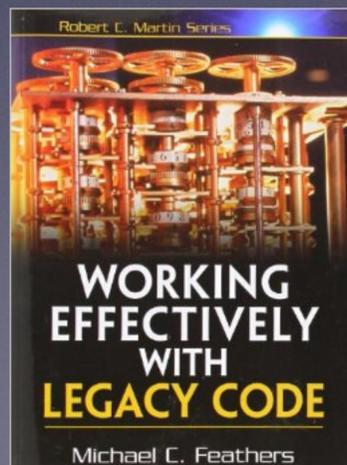
### There are Only Four Billion Floats—So Test Them All!

Posted on [January 27, 2014](#)

A few months ago I saw a blog post touting fancy new SSE3 functions for implementing

# Some good 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:  
<http://www.cs.wustl.edu/~schmidt/patterns-ace.html>
- What everybody should know about floating-point math:  
<http://randomascii.wordpress.com/category/floating-point/>



# Acknowledgments

**GROMACS:** Berk Hess, Szilard Pall, Mark Abraham, Aleksei Liupinov, John Eblen, Roland Shultz, Christian Wennberg, Viveca Lindahl

**RELION:** Dari Kimanius, Björn Forsberg, Sjors Scheres, Alexey Amunts, Marta Carroni, Shintaro Aibara

**NVIDIA:** Mark Berger, Duncan Poole, Julia Levites, Jiri Kraus, Nikolay Markovskiy

**INTEL:** Charles Congdon, Sheng Fu, Kristina Kermanshahche, Yuping Zhao

**CSCS:** Thomas Schulthess, Victor Holanda, Prashant Kanduri **PDC:** Erwin Laure

