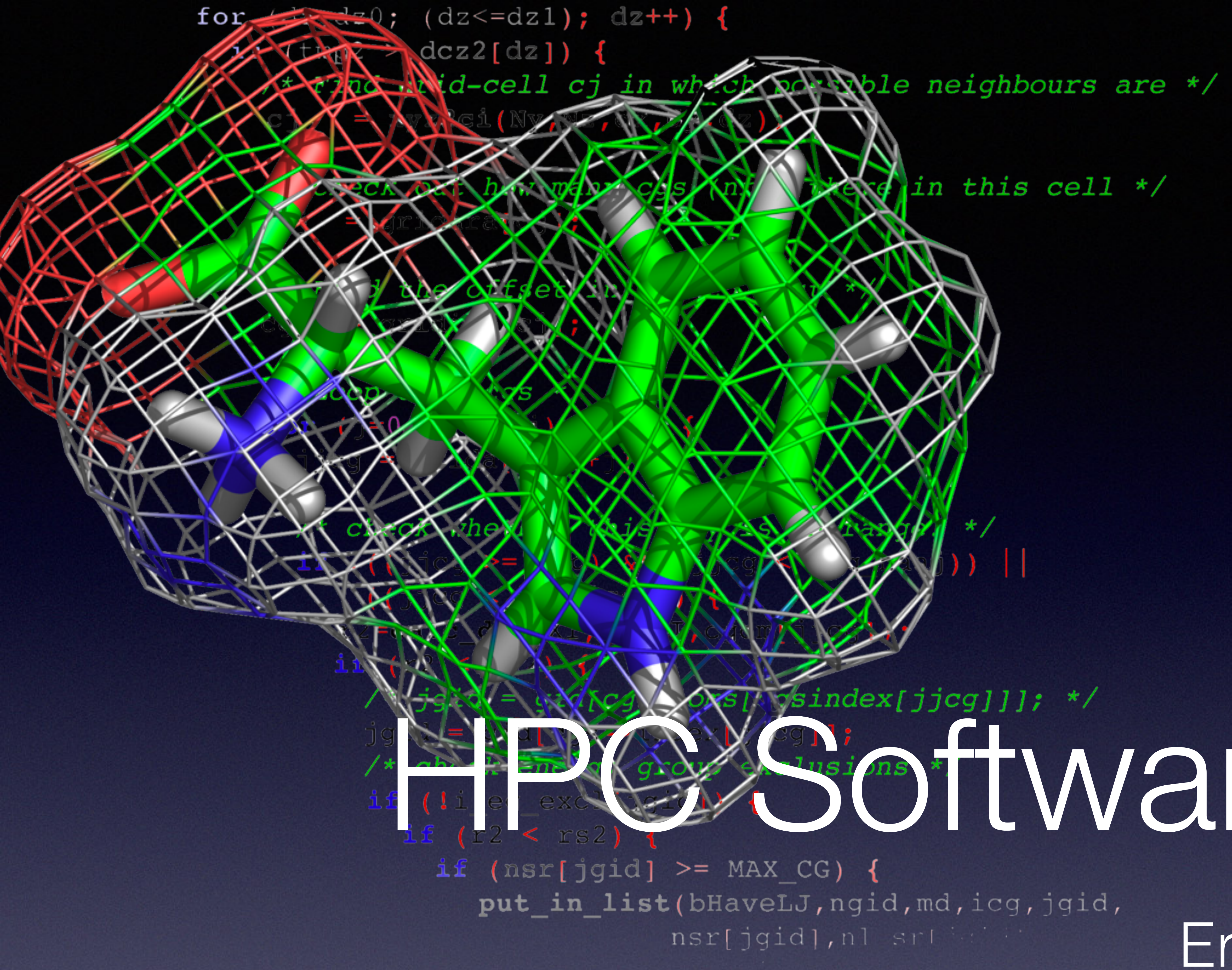

for (dz=dz0; (dz<=dz1); dz++) {
if (tmp2 > dcz2[dz]) {
/* Find grid-cell cj in which possible neighbours are */
/* Check out how many cgs (nrj) there in this cell */
/* check whether ... range */ ||
/* jgid = ...sindex[jjcg]]]; */
/* ... group exclusions */
if (r2 < rs2) {
if (nsr[jgid] >= MAX_CG) {
put_in_list(bHaveLJ,ngid,md,icg,jgid,
nsr[jgid],nl_sr

HPC Software Engineering

Erik Lindahl

XSEDE/PRACE/RIKEN/SciNet HPC Summer School   Ostrava, CZ, 2018

Extra Q & A session: Here (or outside at the tables) 12.45 today!

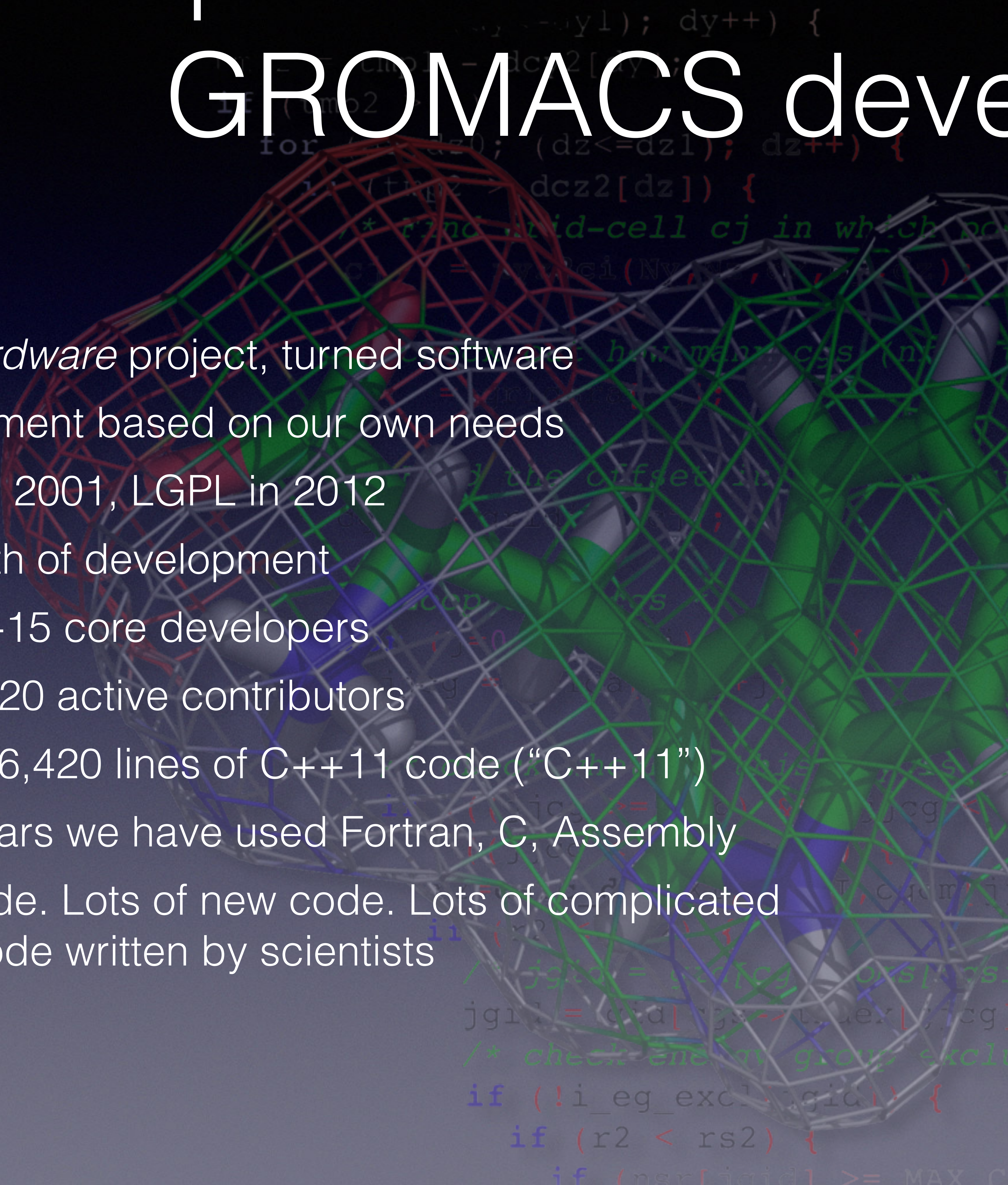
SciLifeLab STOCKHOLM

UNIVERSITET STOCKHOLMS

KTH VETENSKAP OCH KONST

“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.”

Experiences from 20 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

The GROMACS picture until early 2011

Source code repository:

CVS

Build Chain:

Automake/Autoconf/libtool

Bug Tracking:

Bugzilla

Testing:

Scientist

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

Software engineer

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

Without proper software engineering, we are building a technical debt that sooner or later will have to be paid.

“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]

Professional modern development tools

… with illustrations from GROMACS, and a new example based on the IHPCSS Laplace example code!

What changed in our code between IHPCSS 2015 & 2016?
(Basically the difference between GROMACS-5.1 and GROMACS 2016)

957 commits

4163 files changed

393,488 line insertions

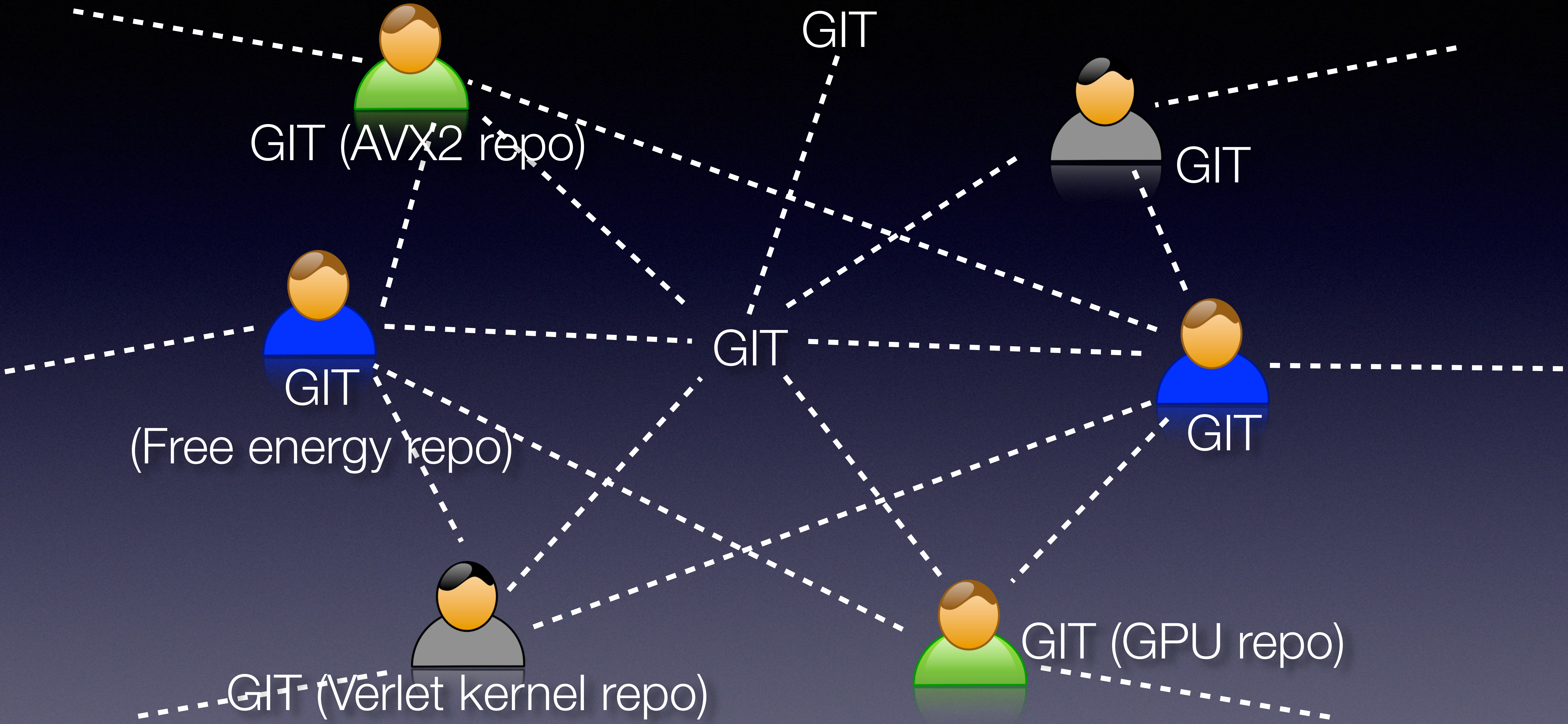
373,227 line deletions

How would you start debugging if the new version crashes?

You have probably all seen this: Your program worked an hour ago, but with the latest edits there is something wrong

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

Better source control: GIT



Free hosting available e.g. at github.com

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

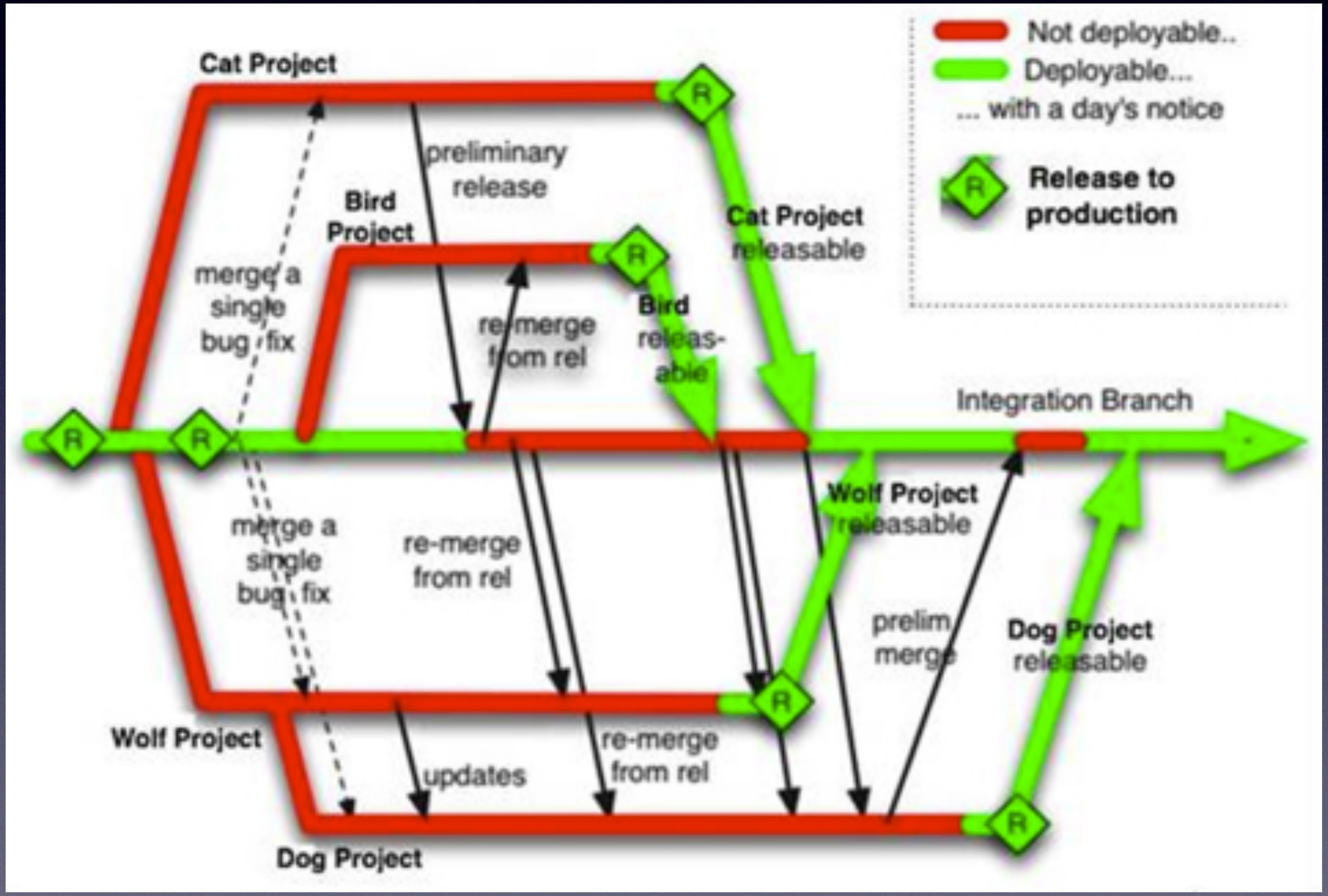
What git will give you

- 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

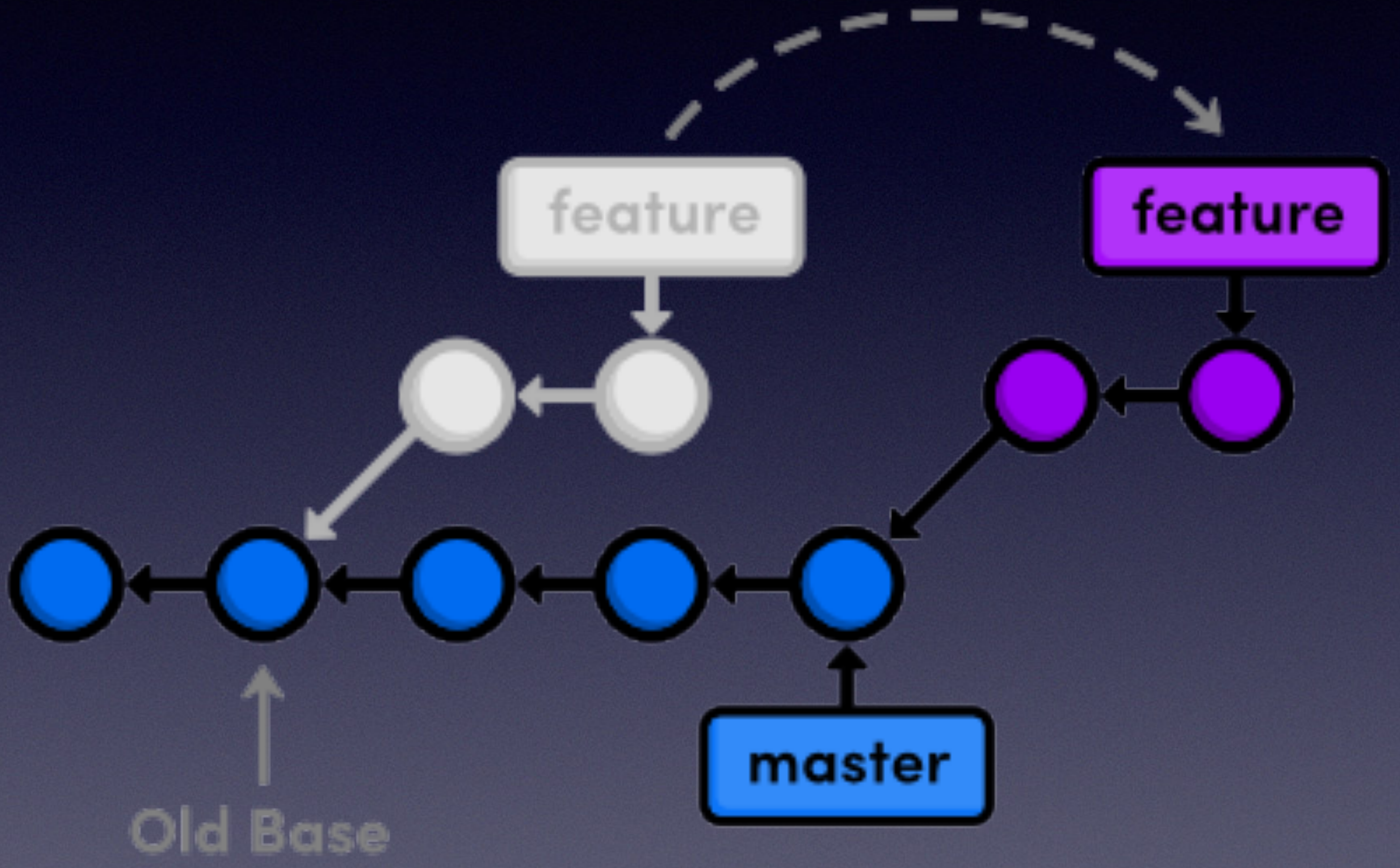
THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

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

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

Have a look at some of the commits in the IHPCSS-laplace repo!

Is your code portable?

Does your code compile on
windows (MSVC)?
PGI Compilers? Pathscale?
Blue Gene?
K computer (Fujitsu compilers)?
ARM? AArch64?
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

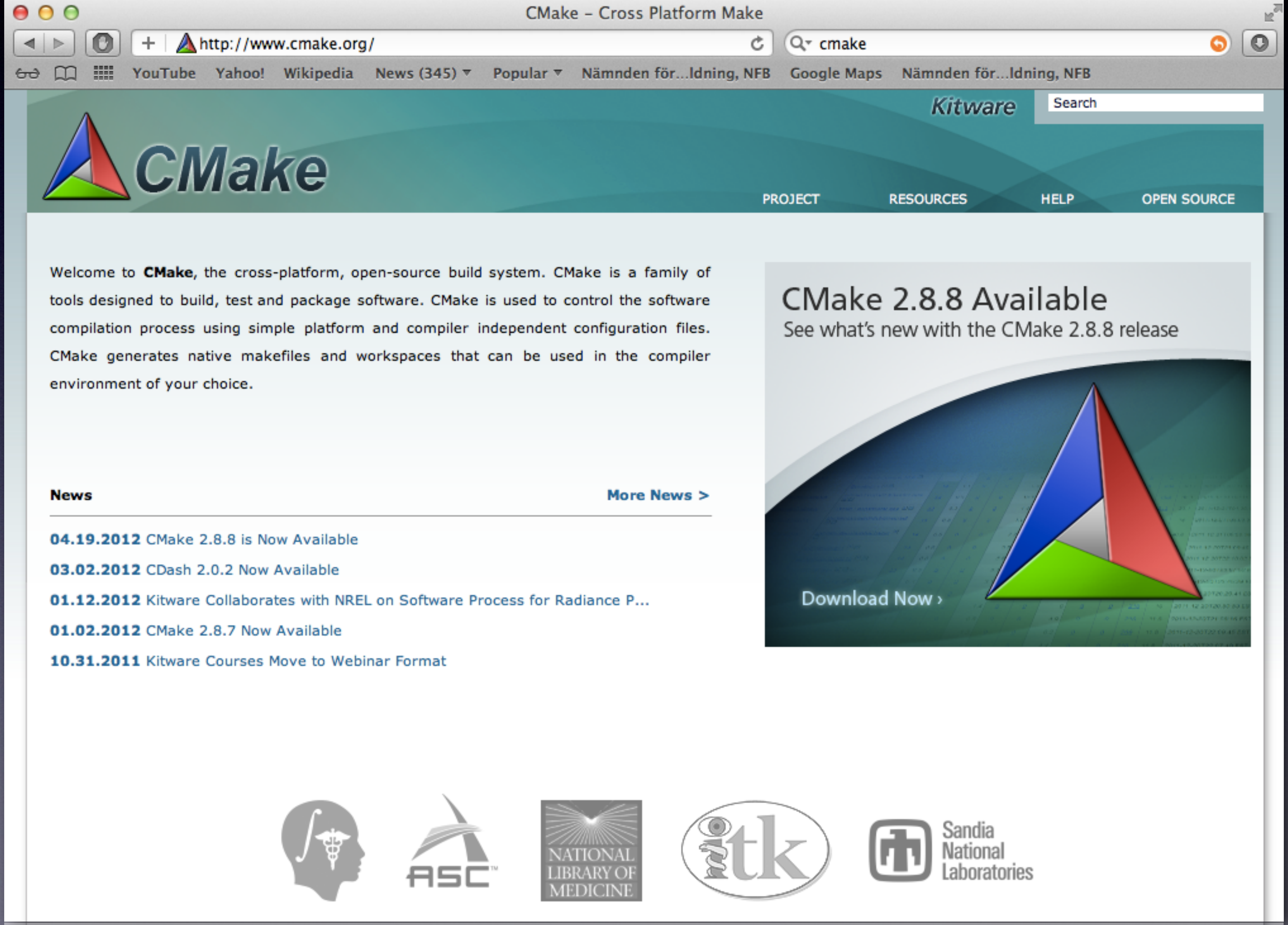
Friends don’t let friends use GNU autotools…

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)



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 **More News >**

04.19.2012 CMake 2.8.8 is Now Available

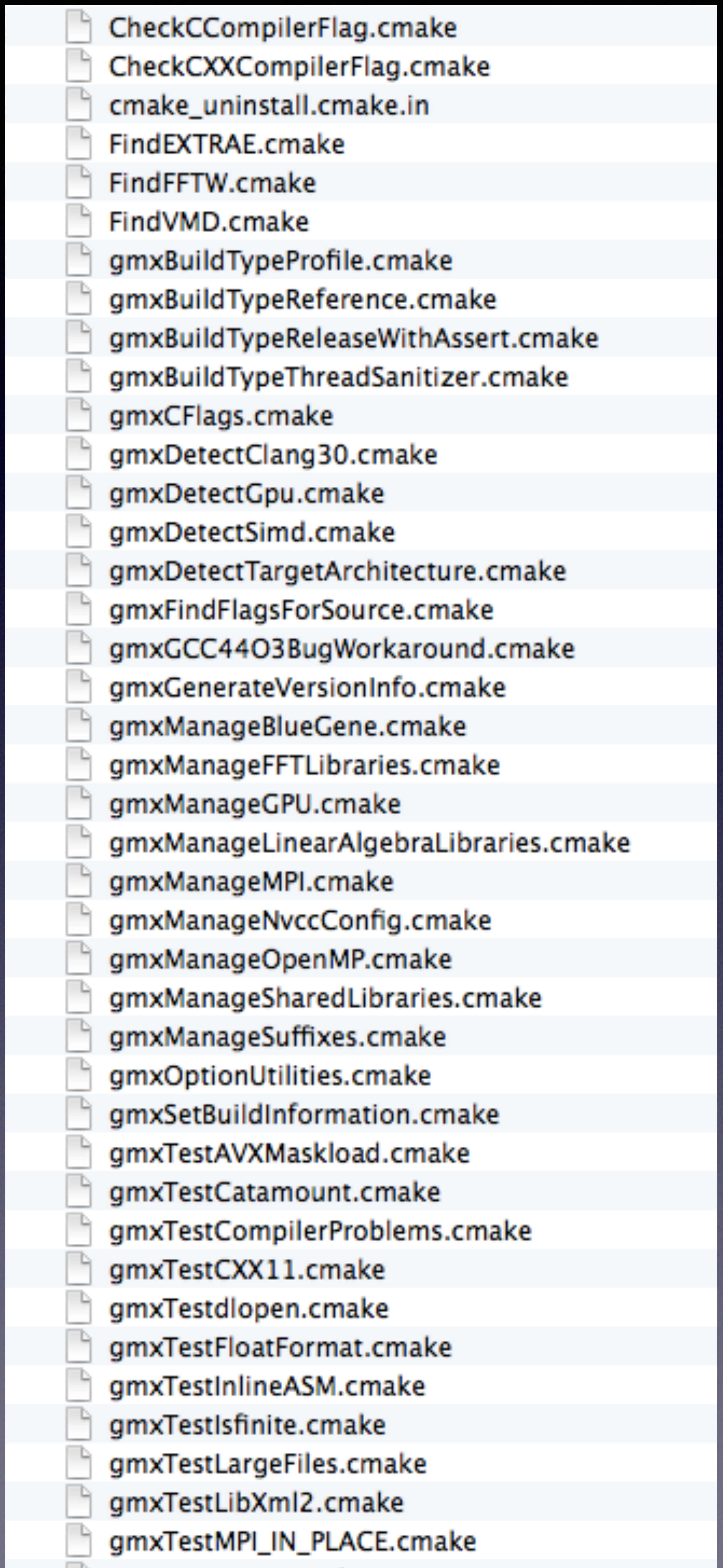
03.02.2012 CDash 2.0.2 Now Available

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

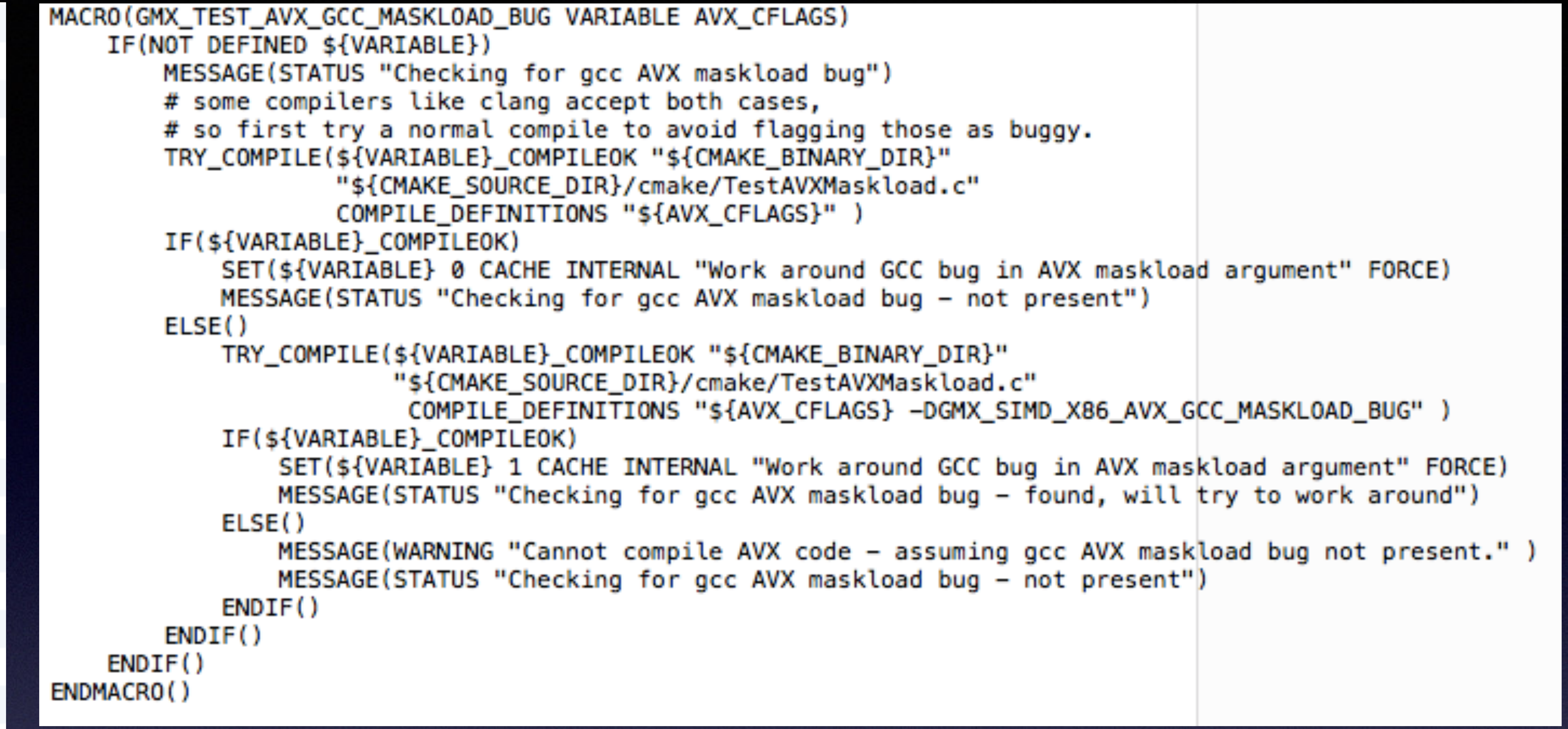
01.02.2012 CMake 2.8.7 Now Available

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()
```

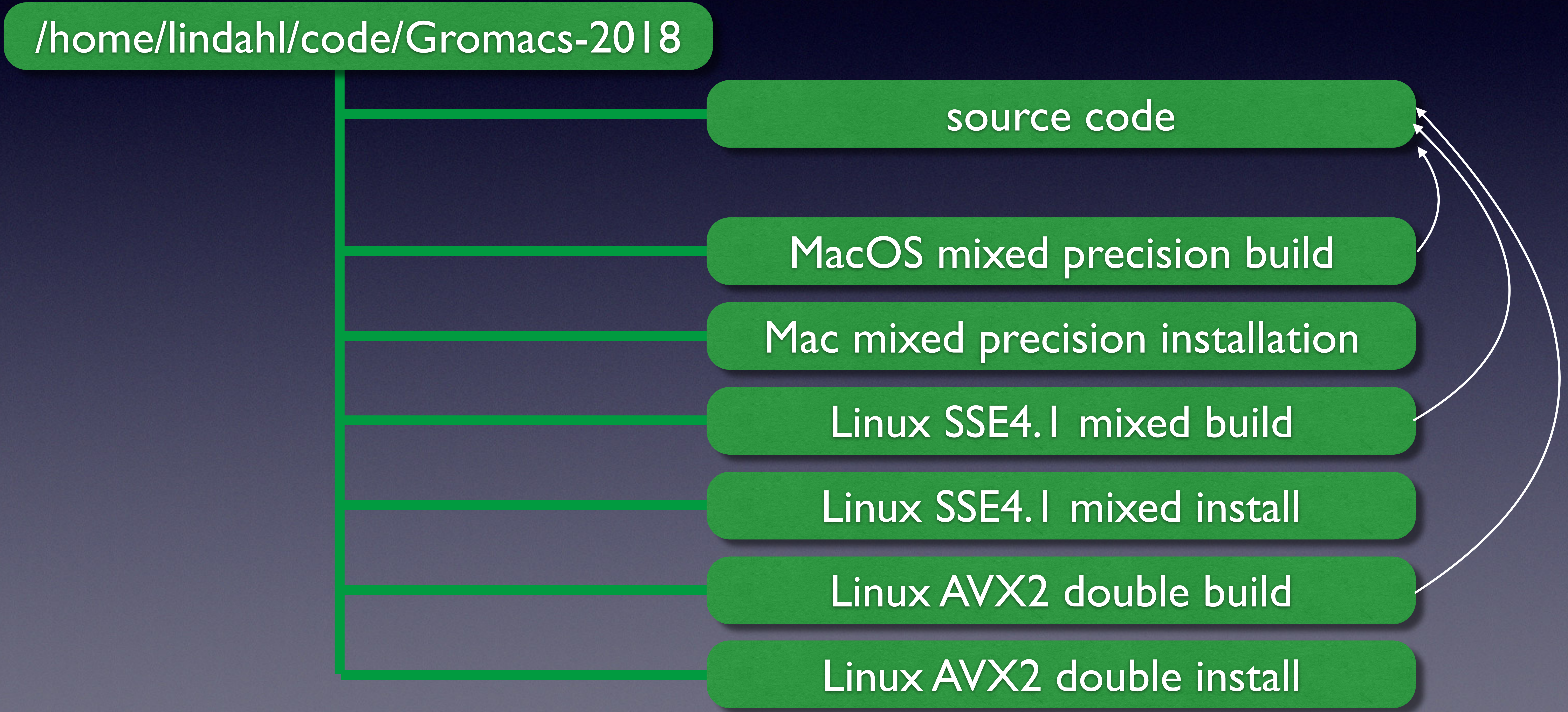
Optional components (FFT libs) and extensive regressiontests can be downloaded automatically

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: ~30 lines, a few source subdirectories, and 3-4 modules in the cmake subdirectory.

Out-of-source builds

Don’t put the build objects inside the source code directory!



/home/lindahl/code/Gromacs-2018

source code

MacOS mixed precision build

Mac mixed precision installation

Linux SSE4.1 mixed build

Linux SSE4.1 mixed install

Linux AVX2 double build

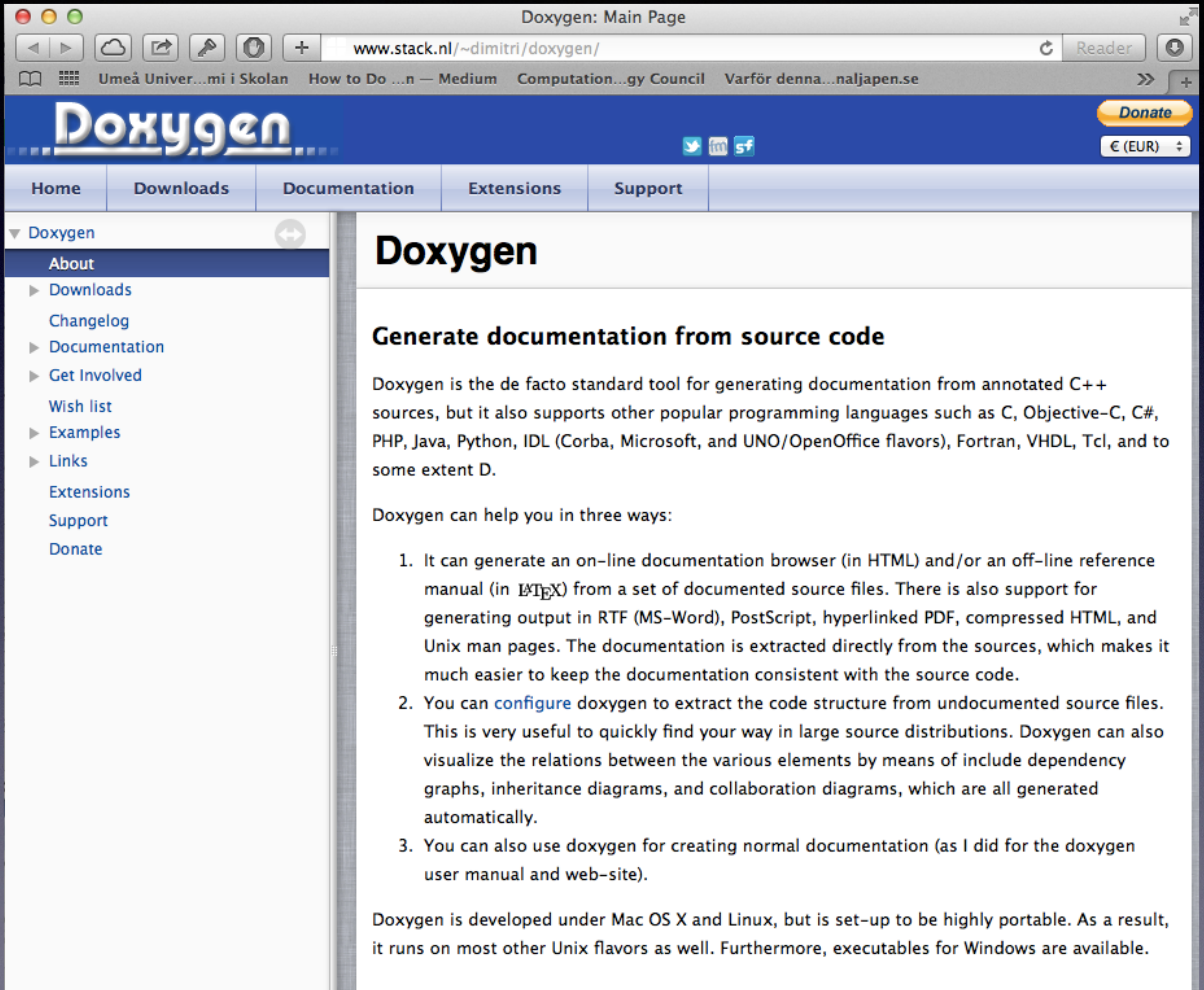
Linux AVX2 double install

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

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

Documentation

Direct source code documentation should stay in the source!



Doxygen

Generate documentation from source code

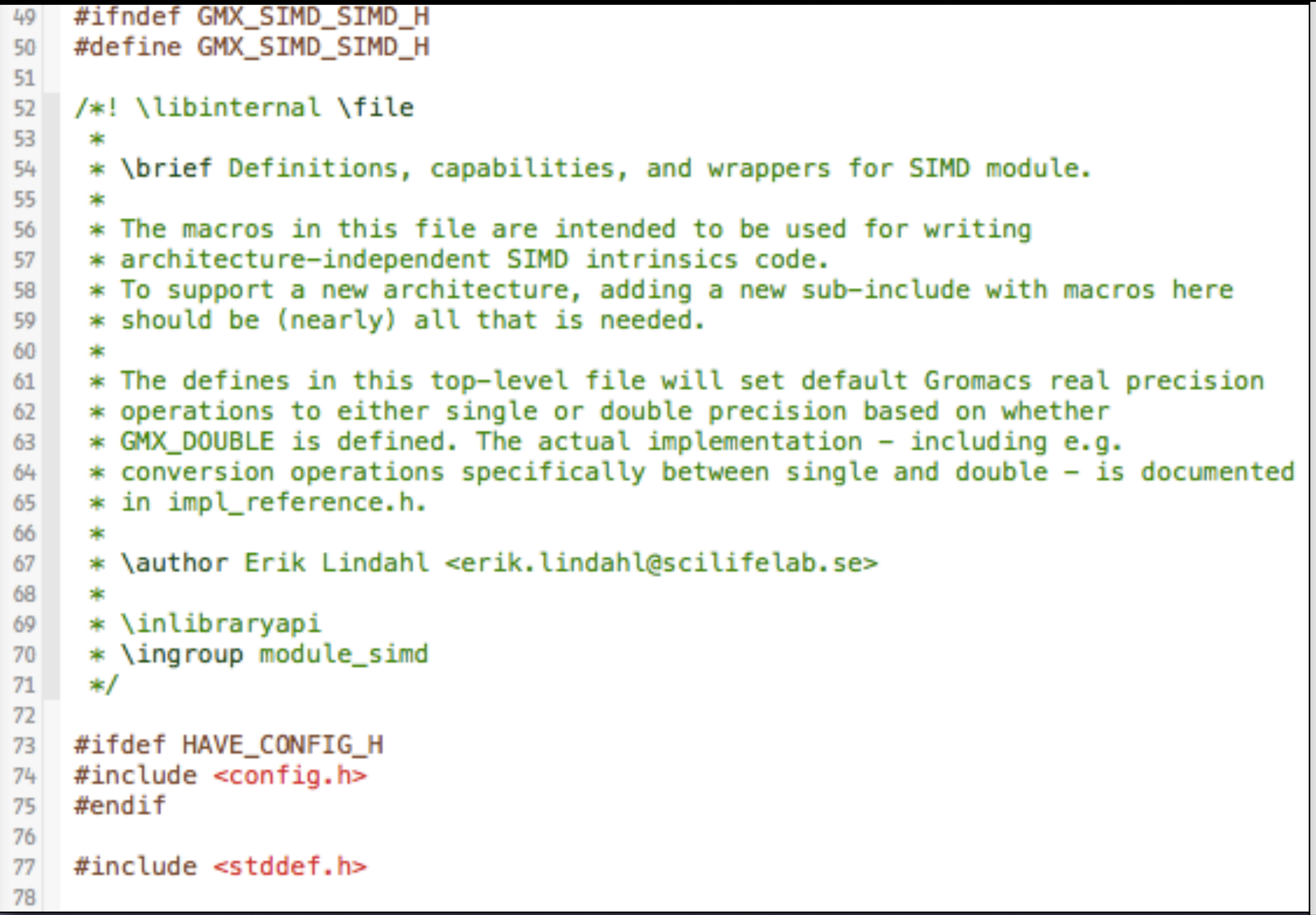
Doxygen is the de facto standard tool for generating documentation from annotated C++ sources, but it also supports other popular programming languages such as C, Objective-C, C#, PHP, Java, Python, IDL (Corba, Microsoft, and UNO/OpenOffice flavors), Fortran, VHDL, Tcl, and to some extent D.

Doxygen can help you in three ways:

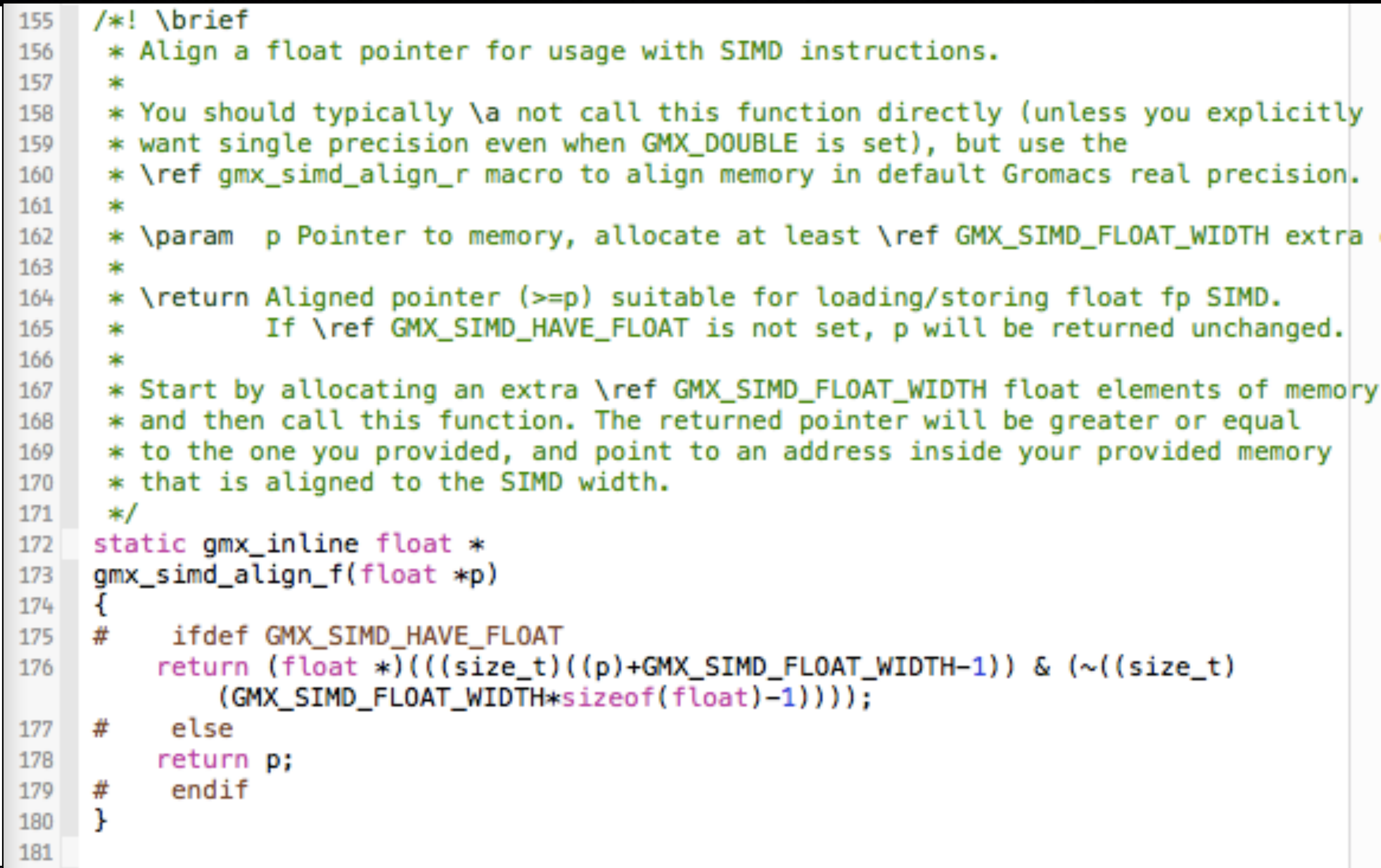
1. It can generate an on-line documentation browser (in HTML) and/or an off-line reference manual (in LaTeX) from a set of documented source files. There is also support for generating output in RTF (MS-Word), PostScript, hyperlinked PDF, compressed HTML, and Unix man pages. The documentation is extracted 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 - our SIMD module:
[gromacs/src/gromacs/simd/]



```
#ifndef GMX_SIMD_SIMD_H
#define GMX_SIMD_SIMD_H

/*! \libinternal \file
 *
 * \brief Definitions, capabilities, and wrappers for SIMD module.
 *
 * The macros in this file are intended to be used for writing
 * architecture-independent SIMD intrinsics code.
 * To support a new architecture, adding a new sub-include with macros here
 * should be (nearly) all that is needed.
 *
 * The defines in this top-level file will set default Gromacs real precision
 * operations to either single or double precision based on whether
 * GMX_DOUBLE is defined. The actual implementation - including e.g.
 * conversion operations specifically between single and double - is documented
 * in impl_reference.h.
 *
 * \author Erik Lindahl <erik.lindahl@scilifelab.se>
 *
 * \inlibraryapi
 * \ingroup module_simd
 */

#ifdef HAVE_CONFIG_H
#include <config.h>
#endif

#include <stddef.h>

```



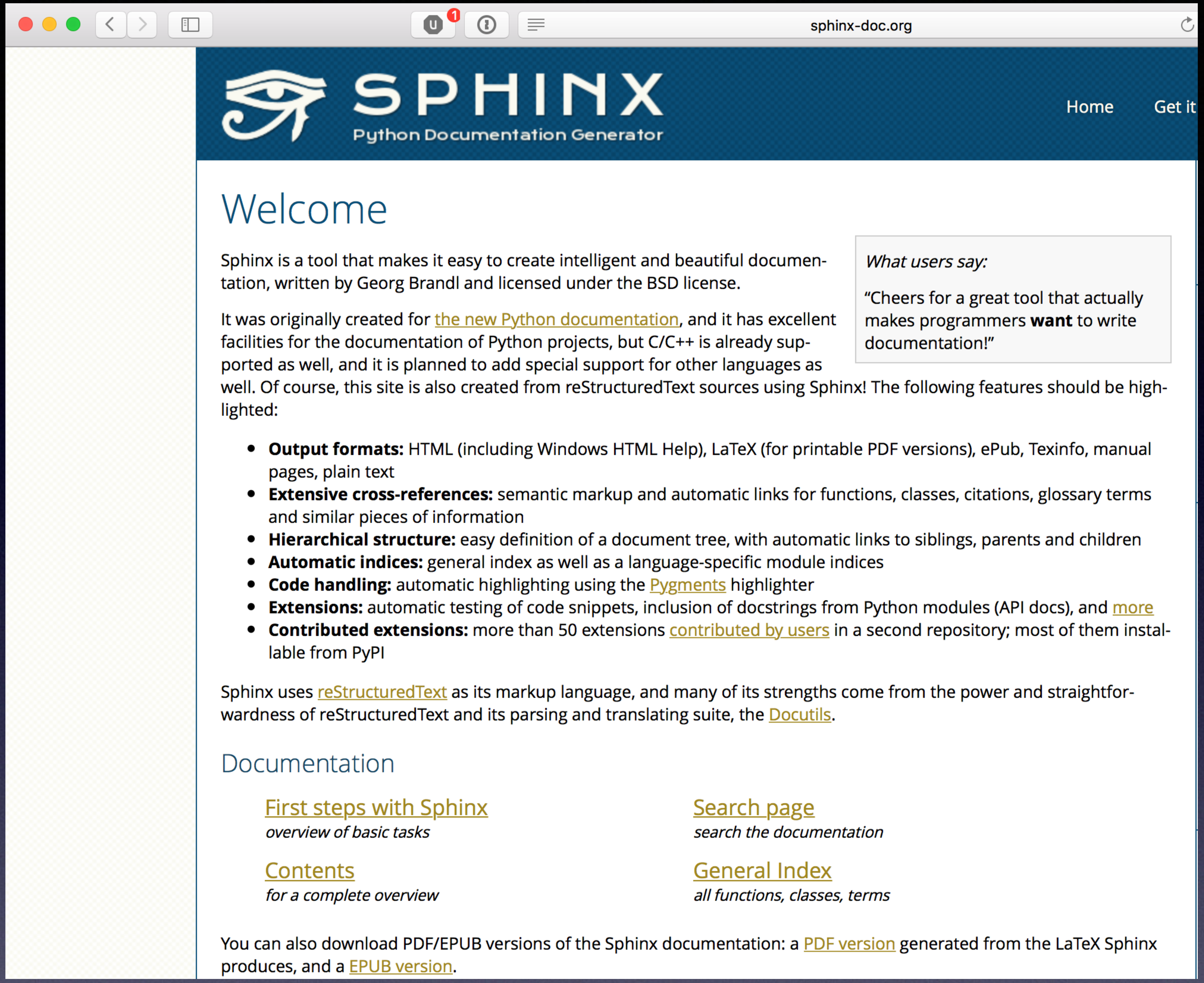
```
/*! \brief
 * Align a float pointer for usage with SIMD instructions.
 *
 * You should typically \a not call this function directly (unless you explicitly
 * want single precision even when GMX_DOUBLE is set), but use the
 * \ref gmx_simd_align_r macro to align memory in default Gromacs real precision.
 *
 * \param  p Pointer to memory, allocate at least \ref GMX_SIMD_FLOAT_WIDTH extra
 *
 * \return Aligned pointer (>=p) suitable for loading/storing float fp SIMD.
 *         If \ref GMX_SIMD_HAVE_FLOAT is not set, p will be returned unchanged.
 *
 * Start by allocating an extra \ref GMX_SIMD_FLOAT_WIDTH float elements of memory
 * and then call this function. The returned pointer will be greater or equal
 * to the one you provided, and point to an address inside your provided memory
 * that is aligned to the SIMD width.
 */
static gmx_inline float *
gmx_simd_align_f(float *p)
{
#    ifdef GMX_SIMD_HAVE_FLOAT
    return (float *)(((size_t)((p)+GMX_SIMD_FLOAT_WIDTH-1)) & (~((size_t)
        (GMX_SIMD_FLOAT_WIDTH*sizeof(float)-1))));
#    else
    return p;
#    endif
}

```

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

For humorous counter-examples, Google "how to write unmaintainable code pdf"

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"

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



Welcome

Sphinx is a tool that makes it easy to create intelligent and beautiful documentation, written by Georg Brandl and licensed under the BSD license.

What users say:

"Cheers for a great tool that actually makes programmers **want** to write documentation!"

It was originally created for the new Python documentation, and it has excellent facilities for the documentation of Python projects, but C/C++ is already supported as well, and it is planned to add special support for other languages as well. Of course, this site is also created from reStructuredText sources using Sphinx! The following features should be highlighted:

- **Output formats:** HTML (including Windows HTML Help), LaTeX (for printable PDF versions), ePub, Texinfo, manual pages, plain text
- **Extensive cross-references:** semantic markup and automatic links for functions, classes, citations, glossary terms and similar pieces of information
- **Hierarchical structure:** easy definition of a document tree, with automatic links to siblings, parents and children
- **Automatic indices:** general index as well as a language-specific module indices
- **Code handling:** automatic highlighting using the Pygments highlighter
- **Extensions:** automatic testing of code snippets, inclusion of docstrings from Python modules (API docs), and more
- **Contributed extensions:** more than 50 extensions contributed by users in a second repository; most of them installable from PyPI

Sphinx uses reStructuredText as its markup language, and many of its strengths come from the power and straightforwardness of reStructuredText and its parsing and translating suite, the Docutils.

Documentation

First steps with Sphinx
overview of basic tasks

Search page
search the documentation

Contents
for a complete overview

General Index
all functions, classes, terms

You can also download PDF/EPUB versions of the Sphinx documentation: a PDF version generated from the LaTeX Sphinx produces, and a EPUB version.

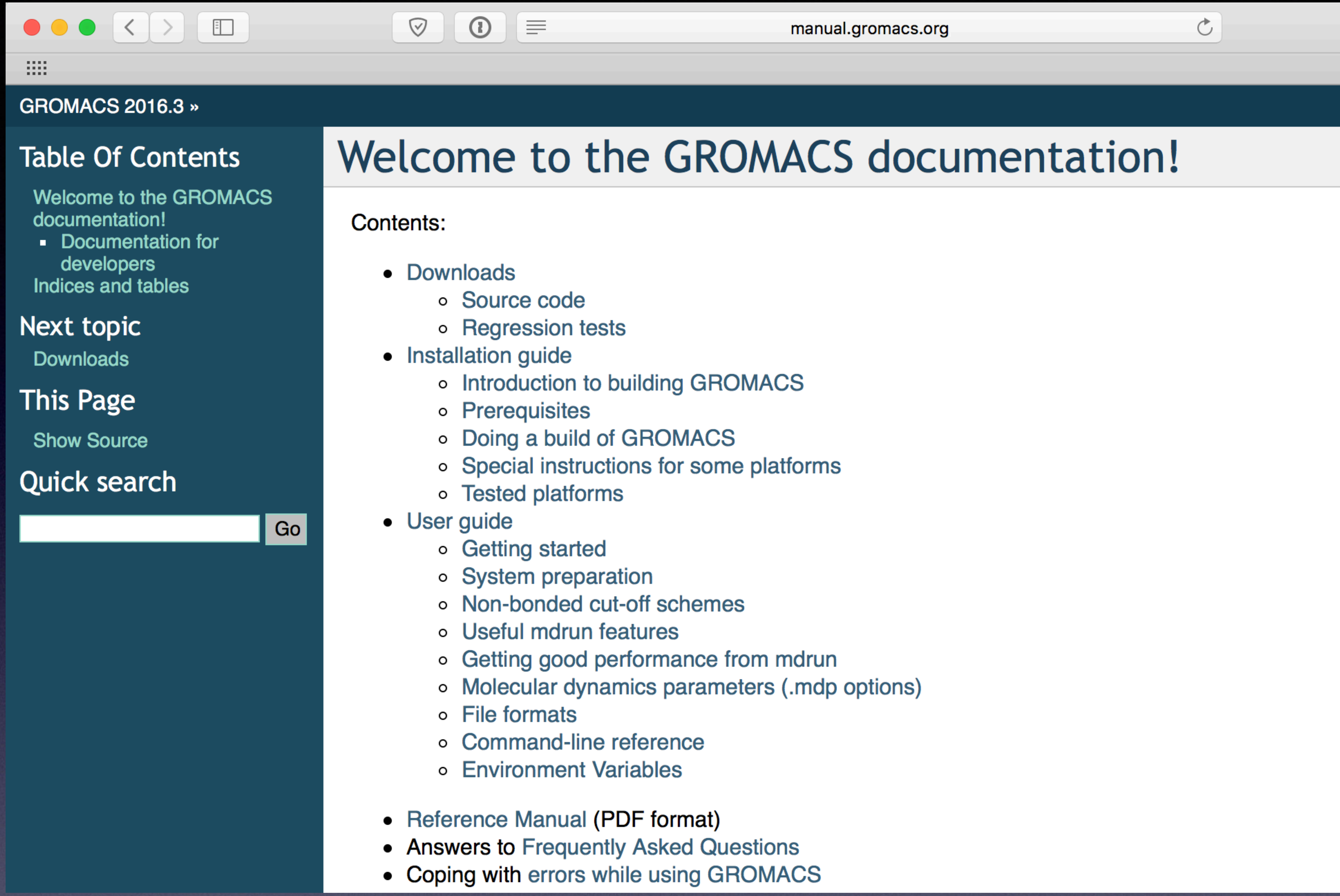


Table Of Contents

- Welcome to the GROMACS documentation!
  - Documentation for developers
- Indices and tables

Next topic

Downloads

This Page

Show Source

Quick search

Welcome to the GROMACS documentation!

Contents:

- Downloads
  - Source code
  - Regression tests
- Installation guide
  - Introduction to building GROMACS
  - Prerequisites
  - Doing a build of GROMACS
  - Special instructions for some platforms
  - Tested platforms
- User guide
  - Getting started
  - System preparation
  - Non-bonded cut-off schemes
  - Useful mdrun features
  - Getting good performance from mdrun
  - Molecular dynamics parameters (.mdp options)
  - File formats
  - Command-line reference
  - Environment Variables

- Reference Manual (PDF format)
- Answers to Frequently Asked Questions
- Coping with errors while using GROMACS

Also 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".

… and we have integrated it with readthedocs.org! Any time a new change is pushed to the gibhub 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
- Make sure all code is thread-safe!
- Have a clear (documented) API for each module
- Write unit tests, not only regression tests
- 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

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 ml(&m);
    …
}
```

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

C/FORTRAN

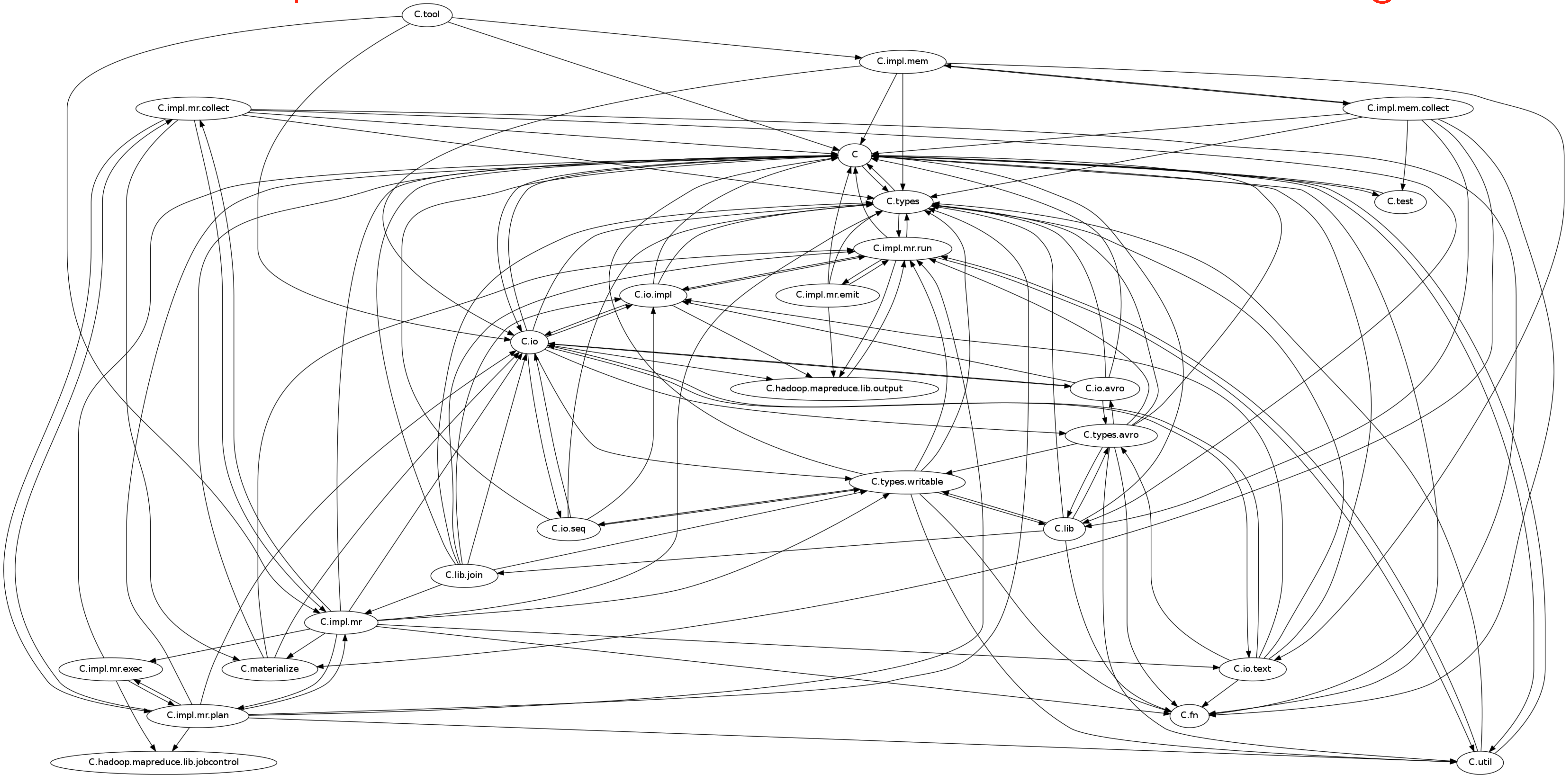
```
int
myFunc(obj_t obj, int choiceA, int choice B)
{
    for(int i=0;i<obj.N;i++)
    {
        if(choiceA==1)
        {
            if(choiceB==1)
            {
                kernelcode1;
            }
            else if(choiceB==2)
            {
                kernelcode2;
            }
    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

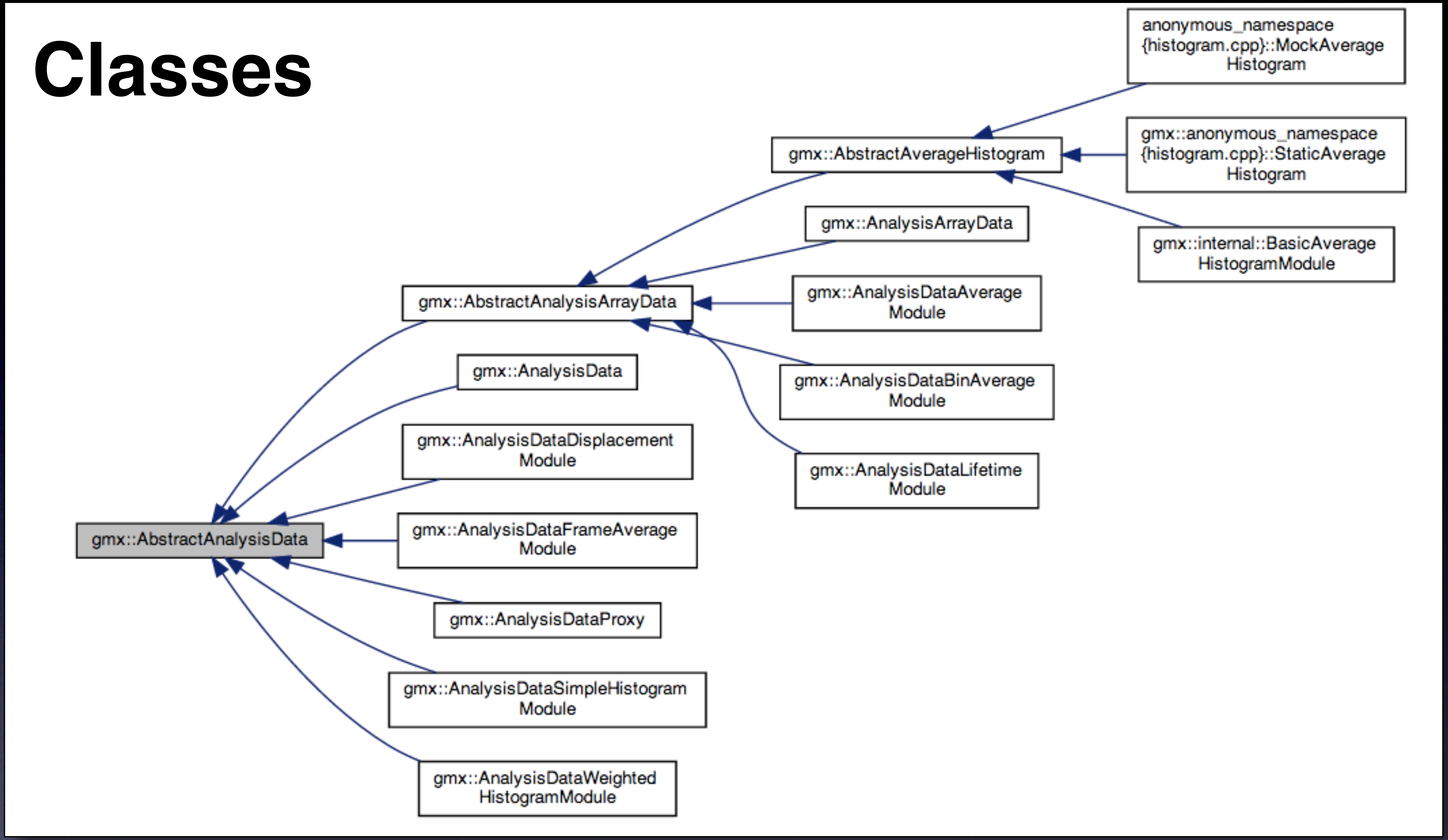
This C++ code will be fully expanded by the compiler. No conditionals present in the generated assembly code.

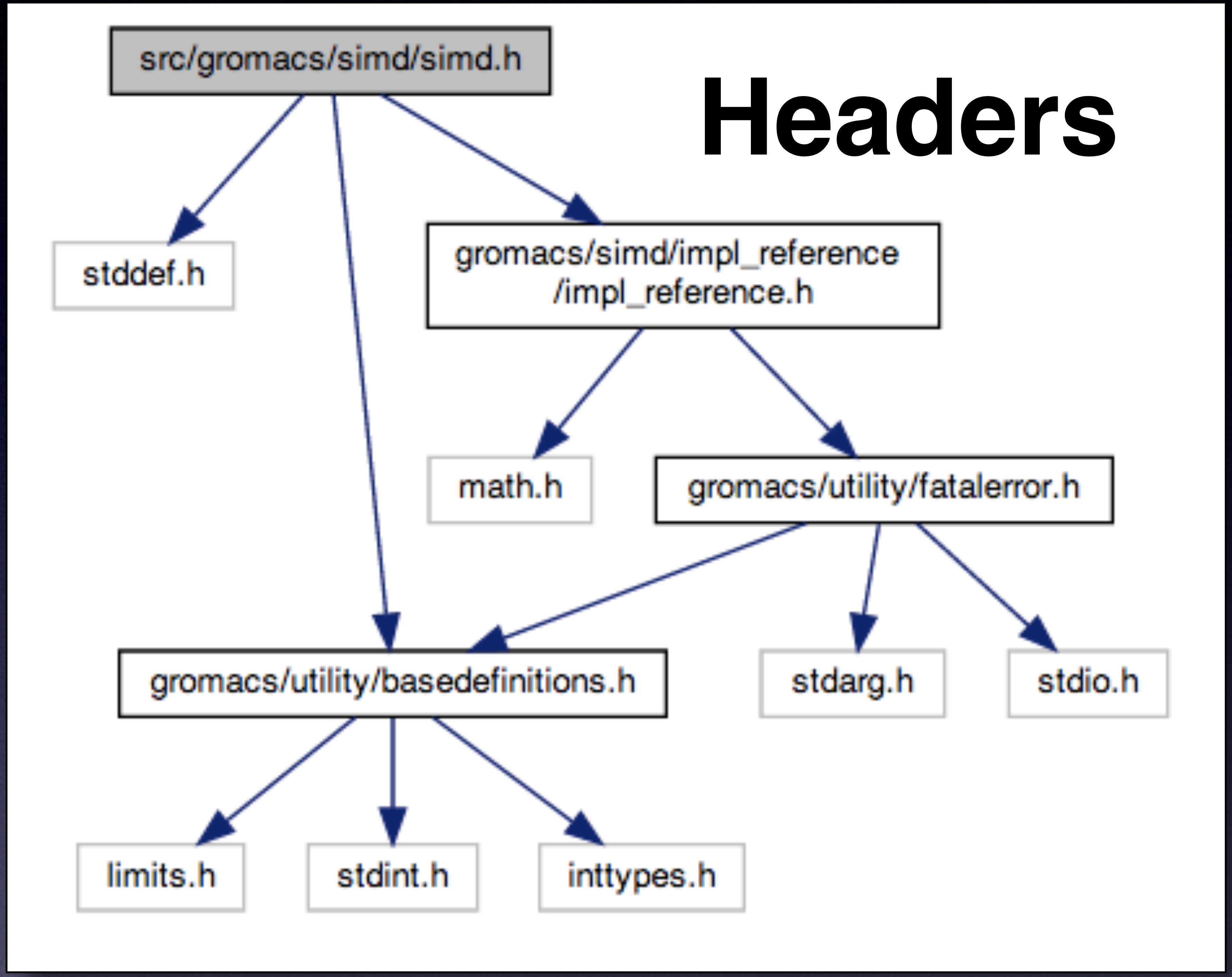
Circular dependencies are bad. If a test fails, where is the bug here?



"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
anonymous_namespace {histogram.cpp}::MockAverage Histogram
gmx::AbstractAverageHistogram
gmx::anonymous_namespace {histogram.cpp}::StaticAverage Histogram
gmx::AnalysisArrayData
gmx::internal::BasicAverage HistogramModule
gmx::AbstractAnalysisArrayData
gmx::AnalysisDataAverage Module
gmx::AnalysisData
gmx::AnalysisDataBinAverage Module
gmx::AnalysisDataDisplacement Module
gmx::AnalysisDataLifetime 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 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”

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
8+1
1k

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

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:

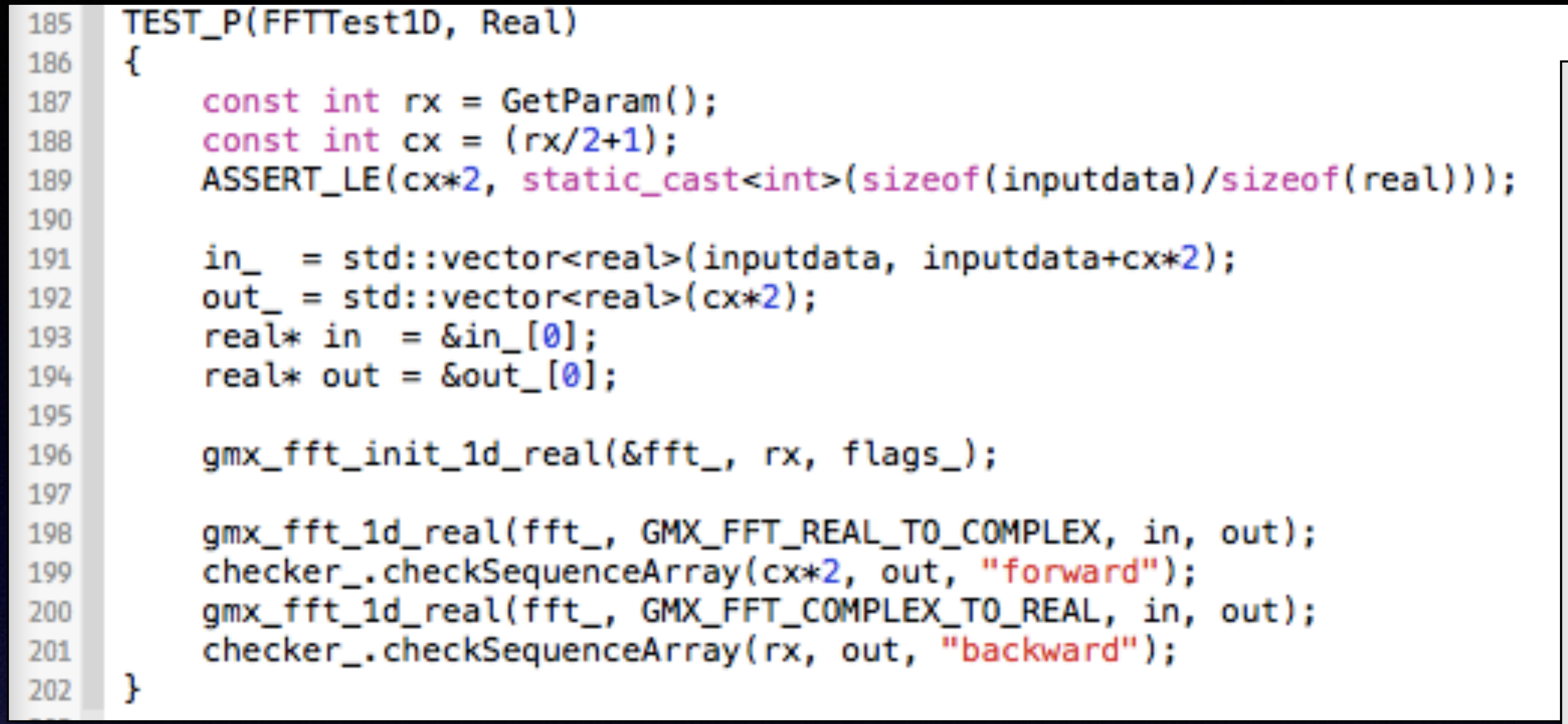
- The Chromium projects (behind the Chrome browser and Chrome OS)
- The LLVM compiler
- Protocol Buffers (Google's data interchange format)

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

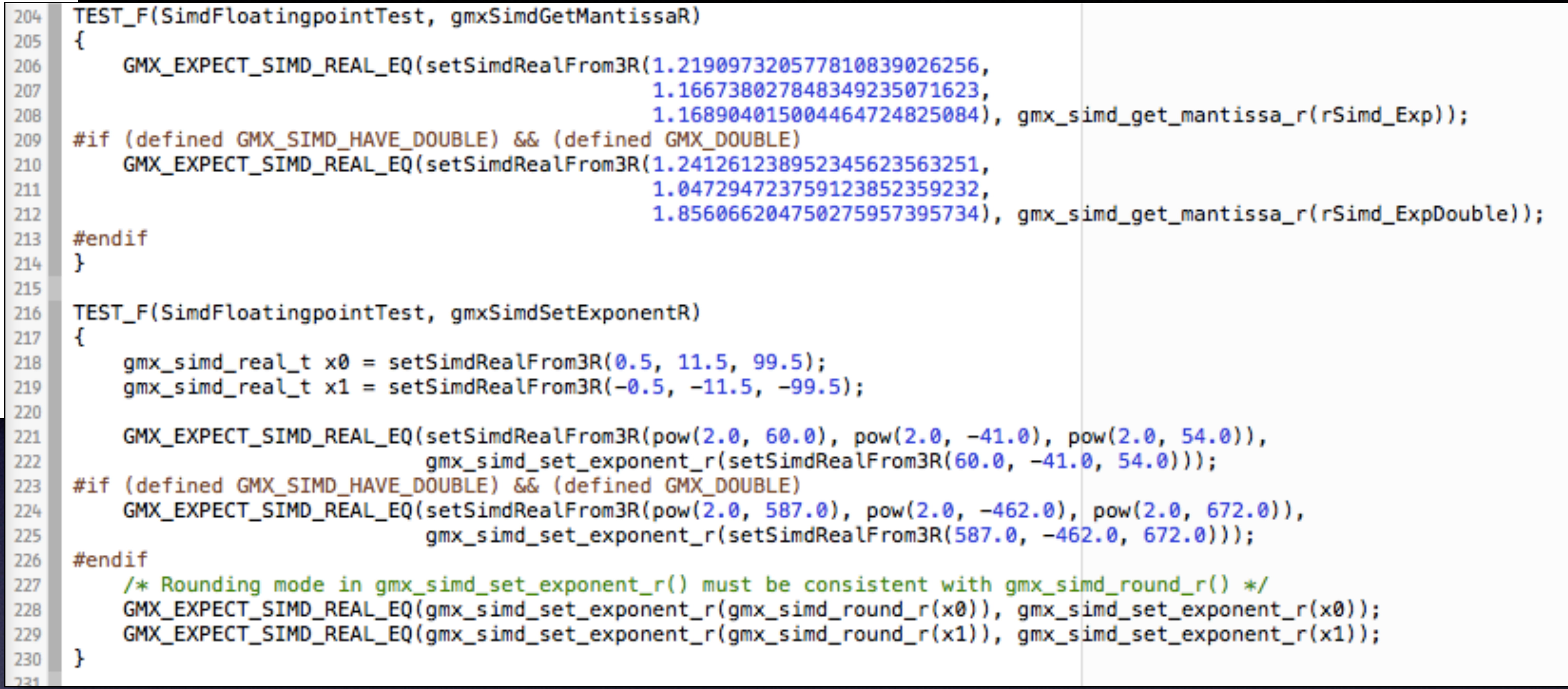
Google Test-related open source projects

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*



```
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");
}
```

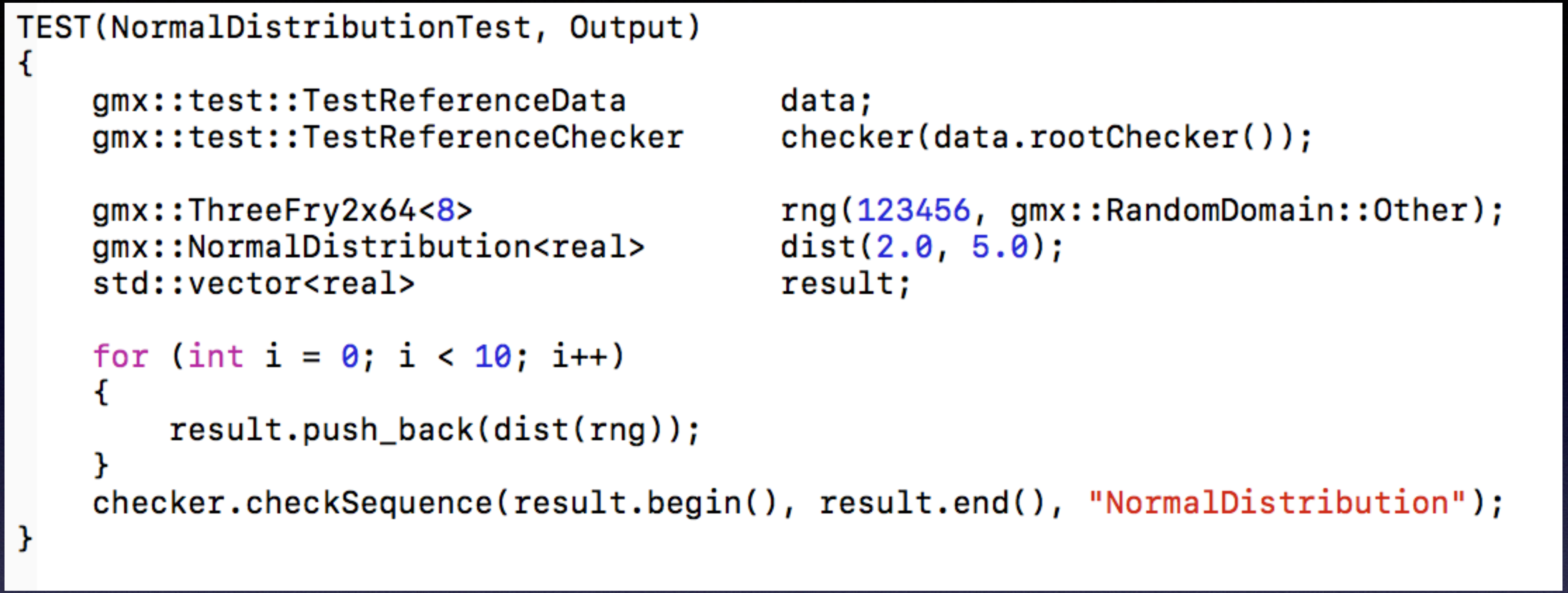


```
TEST_F(SimdFloatingpointTest, gmxSimdGetMantissaR)
{
    GMX_EXPECT_SIMD_REAL_EQ(setSimdRealFrom3R(1.219097320577810839026256,
                                              1.166738027848349235071623,
                                              1.168904015004464724825084), gmx_simd_get_mantissa_r(rSimd_Exp));
#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.

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.

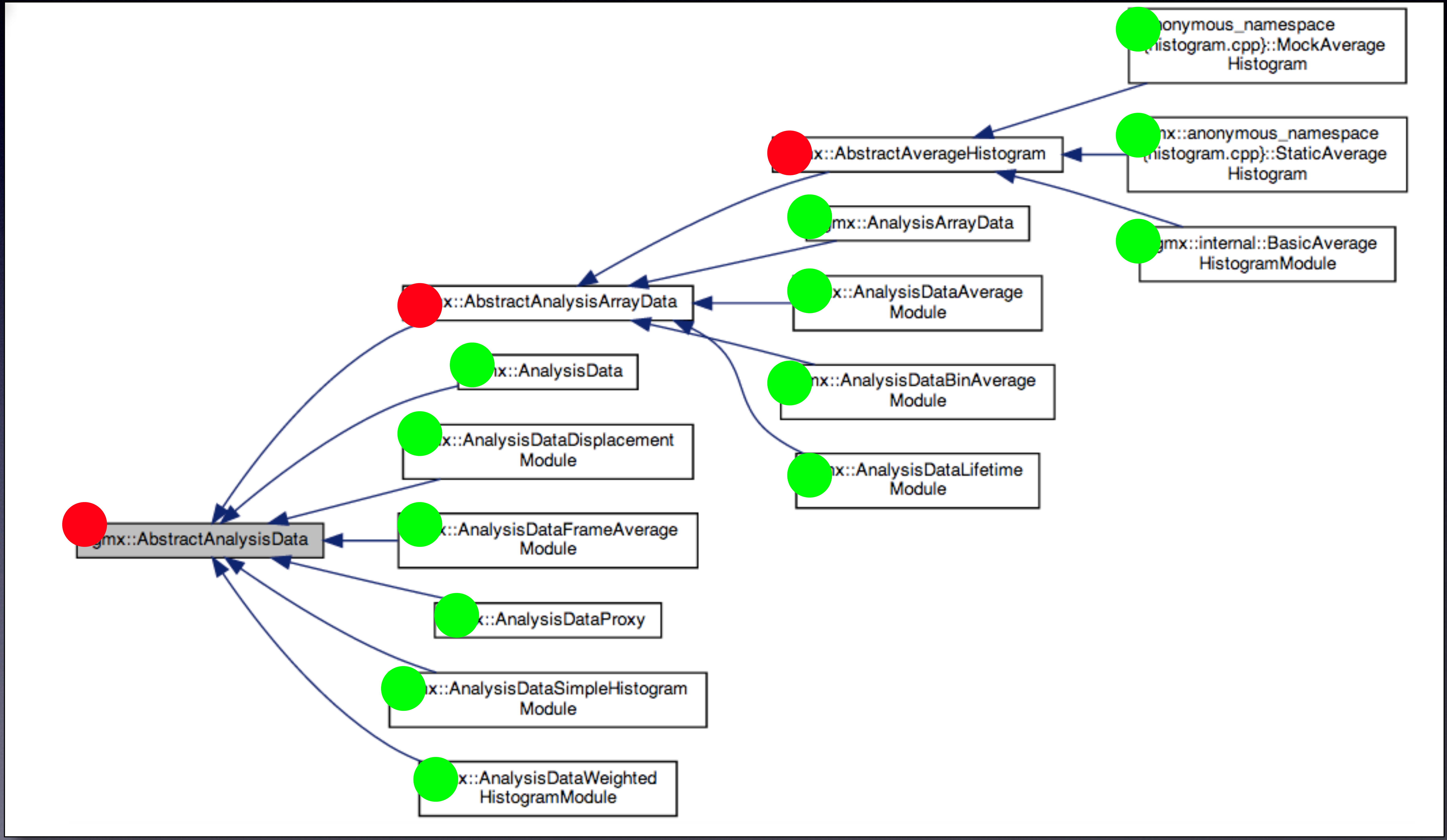
Why? Because we found that libstdc++ and libcxx do not use the same algorithm, so code will not produce the same results. We need to use our own algorithm - make sure it keeps working.

No need to ask: Of course we have integrated GoogleTest support into the IHPCSS/software-engineering repo - but since I only started working on it this Tuesday 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?



onymous_namespace
istogram.cpp}::MockAverage
Histogram
x::AbstractAverageHistogram
nx::anonymous_namespace
istogram.cpp}::StaticAverage
Histogram
gmx::internal::BasicAverage
HistogramModule
mx::AnalysisArrayData
x::AbstractAnalysisArrayData
x::AnalysisDataAverage
Module
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.

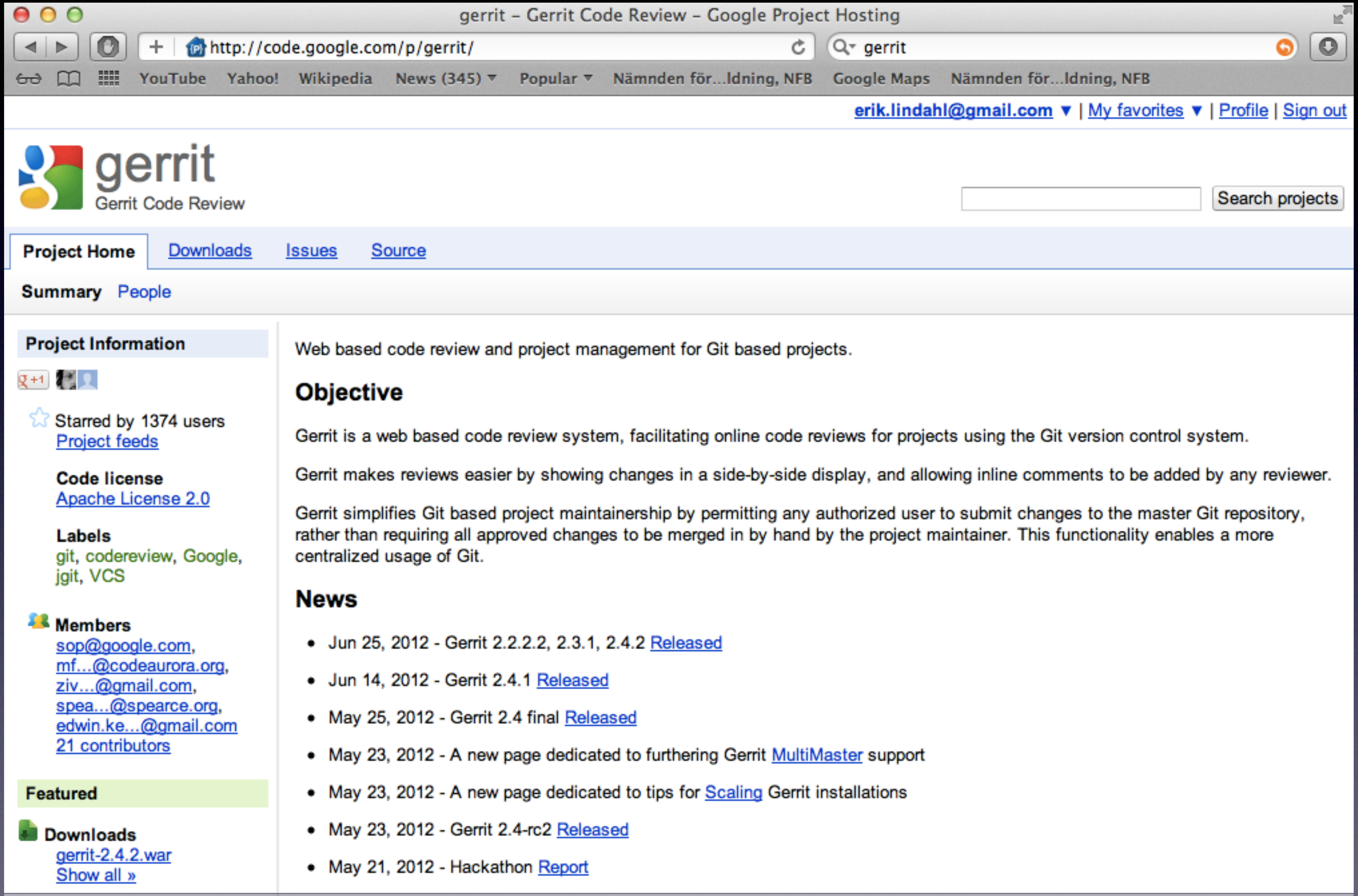
Commits - how code makes it into Gromacs

Who is allowed to write to your code repository?

Problems with developers who submit bad code

Such as this one →

Gerrit Code Review


gerrit – Gerrit Code Review – Google Project Hosting
http://code.google.com/p/gerrit/
gerrit
YouTube
Yahoo!
Wikipedia
News (345)
Popular
Nämnden för...ldning, NFB
Google Maps
Nämnden för...ldning, NFB
erik.lindahl@gmail.com | My favorites | Profile | Sign out
gerrit
Gerrit Code Review
Search projects
Project Home
Downloads
Issues
Source
Summary
People

Project Information

Starred by 1374 users
Project feeds

Code license
Apache License 2.0

Labels
git, codereview, Google, jgit, VCS

Members
sop@google.com, mf...@codeaurora.org, ziv...@gmail.com, spea...@spearce.org, edwin.ke...@gmail.com
21 contributors

Featured

Downloads
gerrit-2.4.2.war
Show all »

Web based code review and project management for Git based projects.

Objective

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.

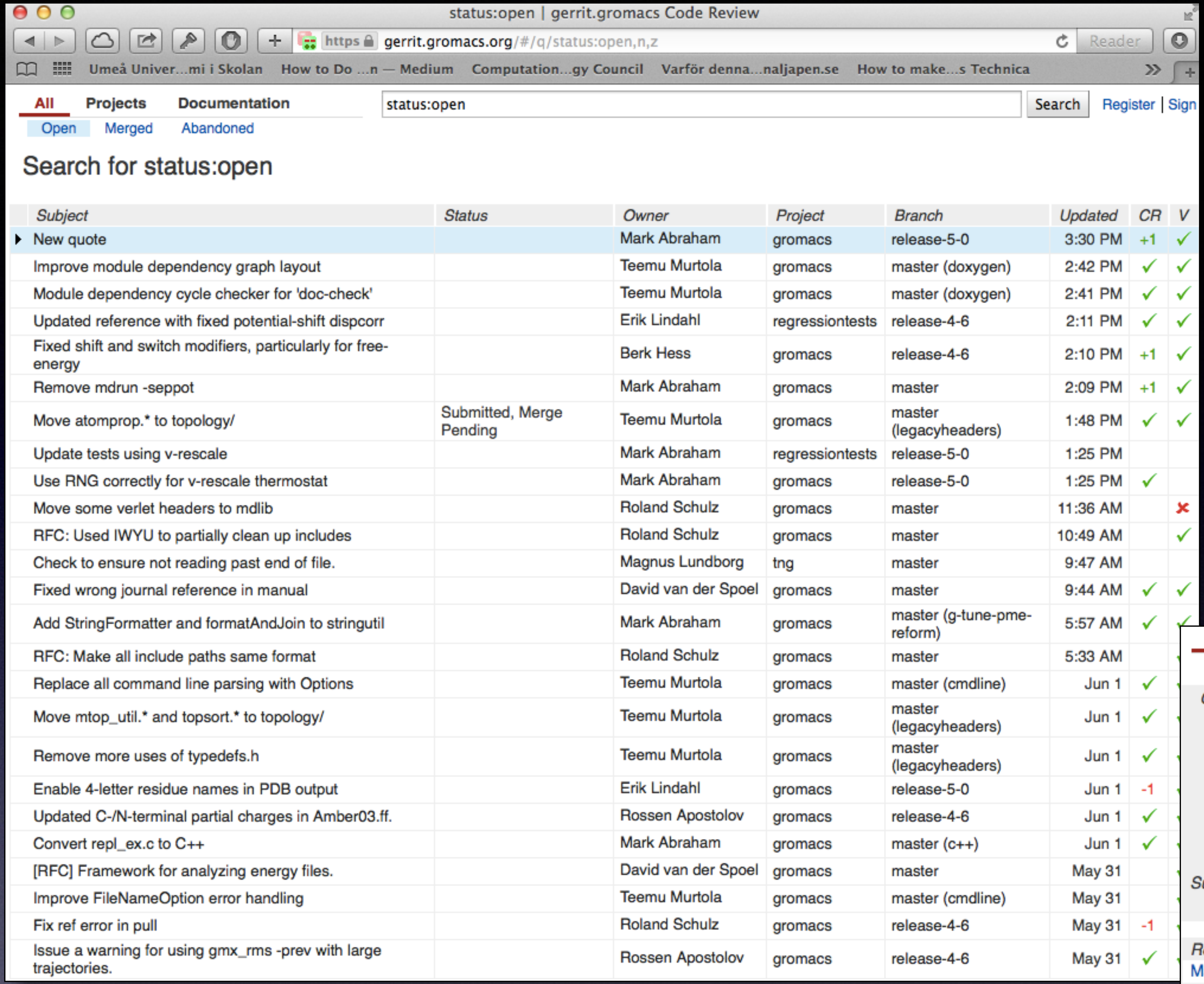
Gerrit simplifies Git based project maintainership by permitting any authorized user to submit changes to the master Git repository, rather than requiring all approved changes to be merged in by hand by the project maintainer. This functionality enables a more centralized usage of Git.

News

- Jun 25, 2012 - Gerrit 2.2.2.2, 2.3.1, 2.4.2 Released
- Jun 14, 2012 - Gerrit 2.4.1 Released
- May 25, 2012 - Gerrit 2.4 final Released
- May 23, 2012 - A new page dedicated to furthering Gerrit MultiMaster support
- May 23, 2012 - A new page dedicated to tips for Scaling Gerrit installations
- May 23, 2012 - Gerrit 2.4-rc2 Released
- May 21, 2012 - Hackathon Report

Nobody can commit directly to our central Git repo anymore

... which means we can allow anybody to commit in gerrit!



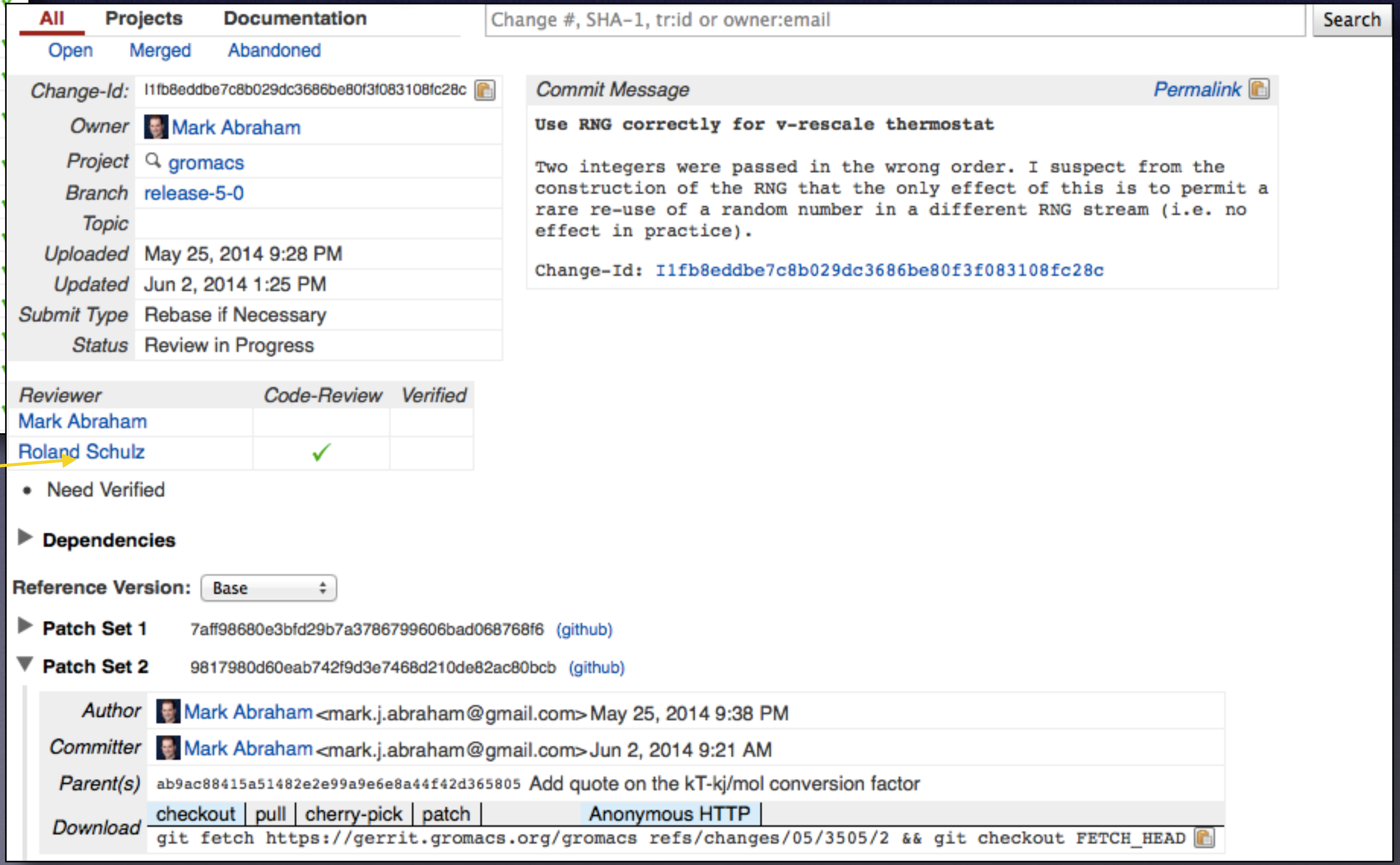
status:open

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



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

Change-Id:	I1fb8eddbe7c8b029dc3686be80f3f083108fc28c
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

```
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: I1fb8eddbe7c8b029dc3686be80f3f083108fc28c
```

Reviewer	Code-Review	Verified
Mark Abraham		
Roland Schulz	✓	

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	git fetch https://gerrit.gromacs.org/gromacs refs/changes/05/3505/2 && git checkout FETCH_HEAD

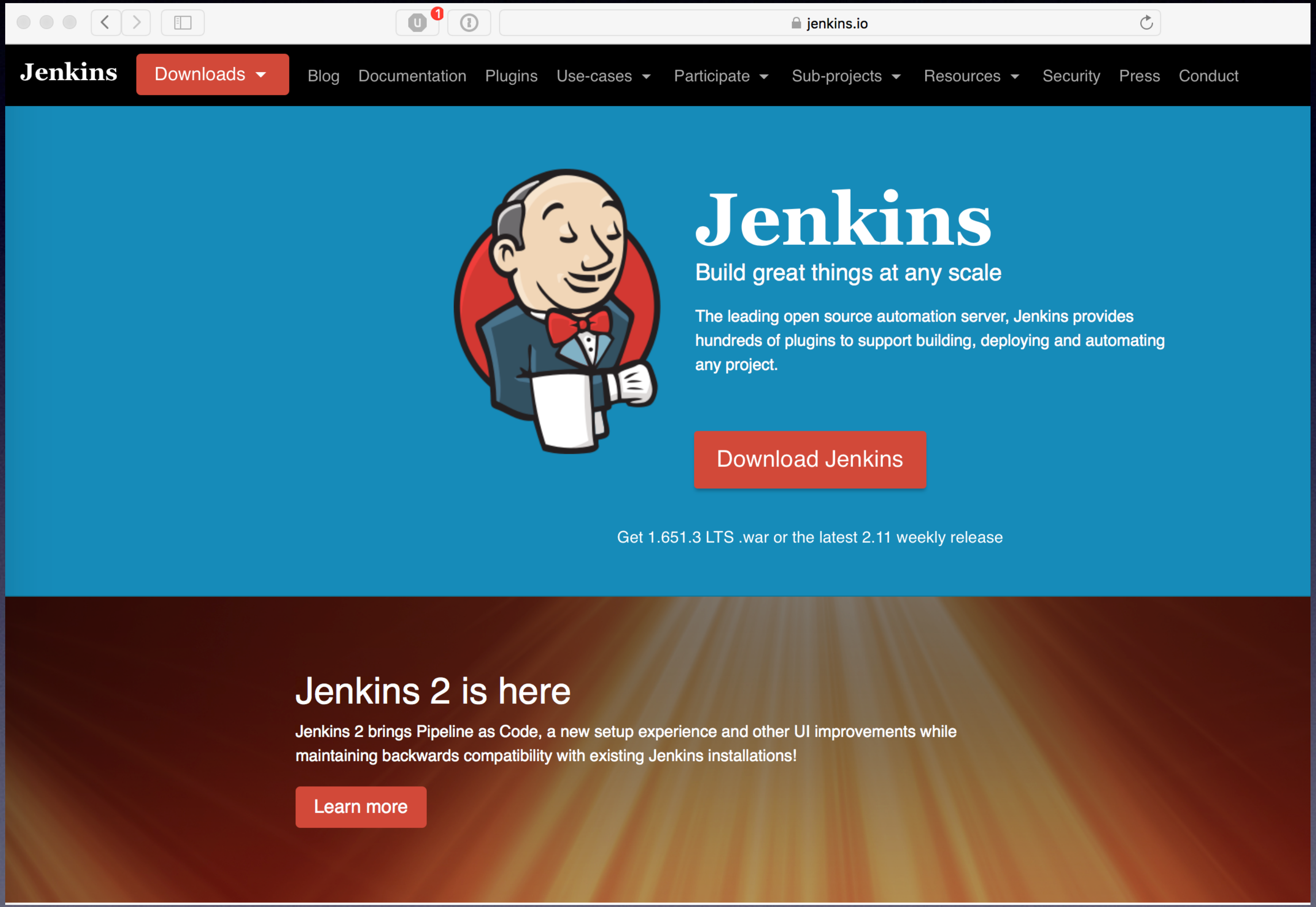
Roland has approved Mark's patch. Anybody can add comments. When two trusted developers say OK, the patch is committed.

Maintaining quality & avoiding breaking stuff

How do I make sure that *I* don’t make mistakes?

Jenkins Continuous Integration

https://jenkins.io



Jenkins

Build great things at any scale

The leading open source automation server, Jenkins provides hundreds of plugins to support building, deploying and automating any project.

Get 1.651.3 LTS .war or the latest 2.11 weekly release

Jenkins 2 is here

Jenkins 2 brings Pipeline as Code, a new setup experience and other UI improvements while maintaining backwards compatibility with existing Jenkins installations!

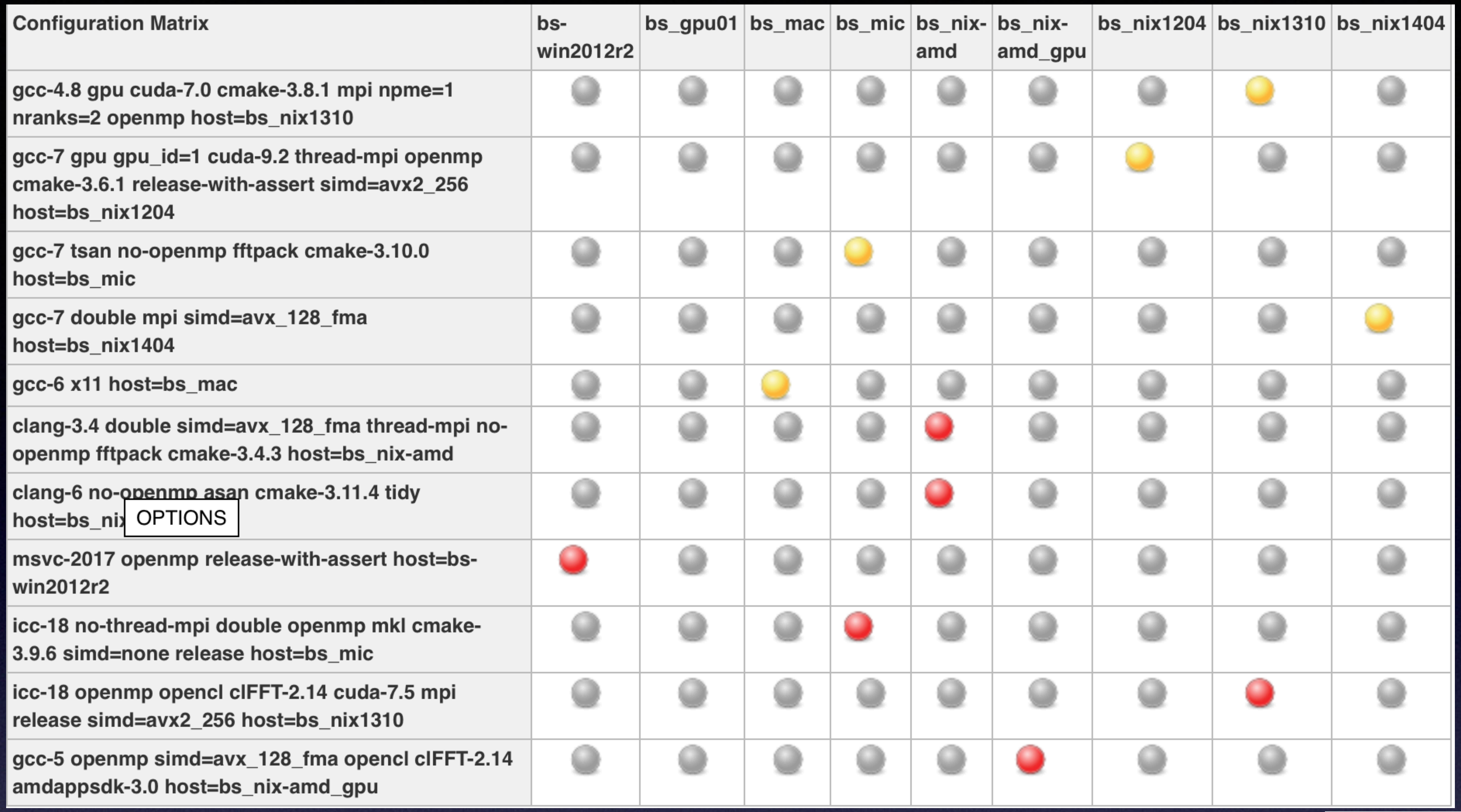
Every single commit is tested automatically on our build farm, including both builds and regression tests.

Results are integrated into the gerrit review

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

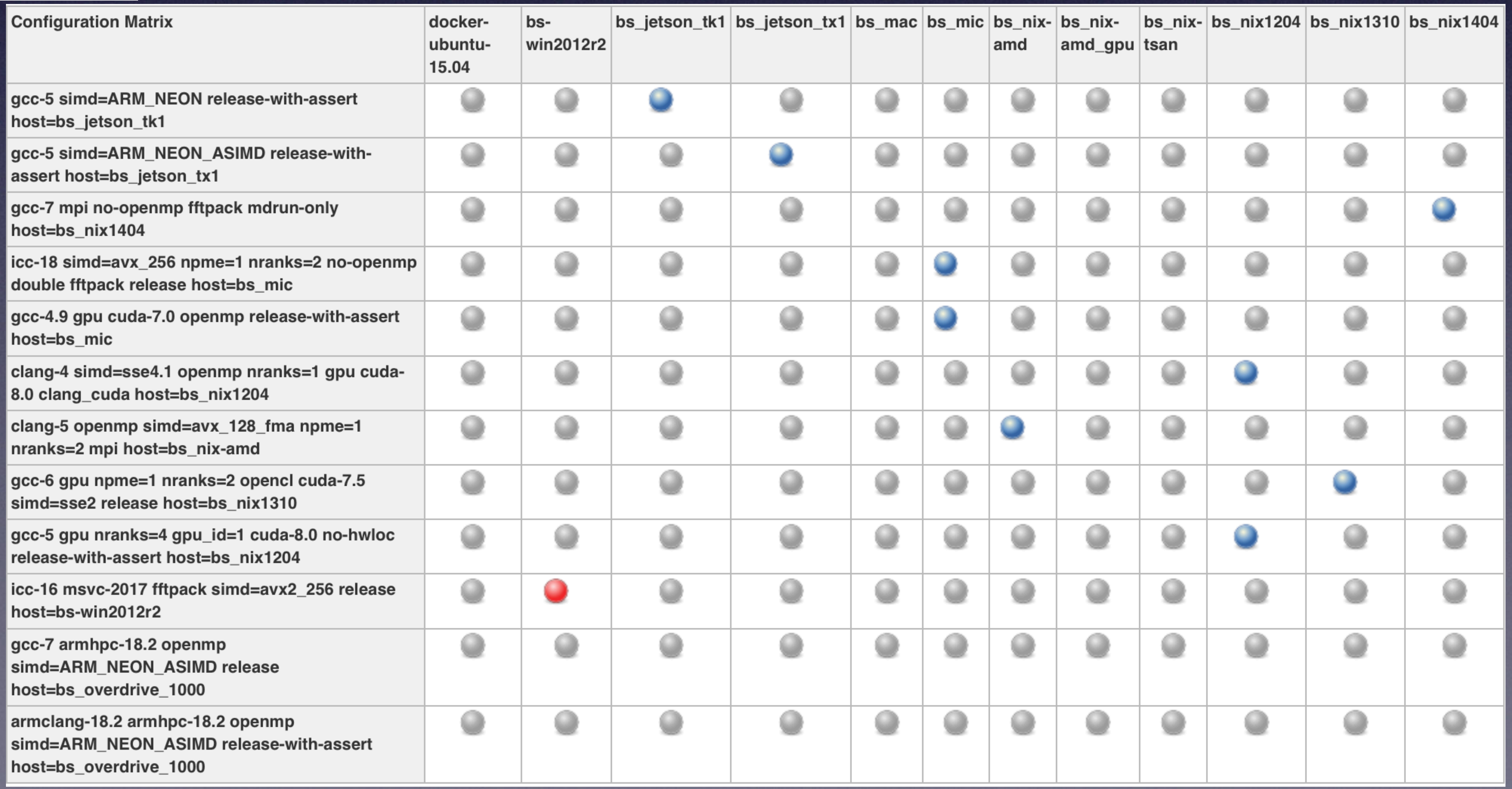
GROMACS CI tests for every commit

- Unit Tests: Do modules reproduce reference values?
- 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
- Uncrustify: Proper code formatting, no tabs, brace standards?
- Doxygen: All classes/methods/arguments/variables documented?
- Coming: Performance regression testing



Configuration Matrix	bs-win2012r2	bs_gpu01	bs_mac	bs_mic	bs_nix-amd	bs_nix-amd_gpu	bs_nix1204	bs_nix1310	bs_nix1404
gcc-4.8 gpu cuda-7.0 cmake-3.8.1 mpi npme=1 nranks=2 openmp host=bs_nix1310	●	●	●	●	●	●	●	🟡	●
gcc-7 gpu gpu_id=1 cuda-9.2 thread-mpi openmp cmake-3.6.1 release-with-assert simd=avx2_256 host=bs_nix1204	●	●	●	●	●	●	🟡	●	●
gcc-7 tsan no-openmp fftpack cmake-3.10.0 host=bs_mic	●	●	●	🟡	●	●	●	●	●
gcc-7 double mpi simd=avx_128_fma host=bs_nix1404	●	●	●	●	●	●	●	●	🟡
gcc-6 x11 host=bs_mac	●	●	🟡	●	●	●	●	●	●
clang-3.4 double simd=avx_128_fma thread-mpi no-openmp fftpack cmake-3.4.3 host=bs_nix-amd	●	●	●	●	🔴	●	●	●	●
clang-6 no-openmp asan cmake-3.11.4 tidy host=bs_ni OPTIONS	●	●	●	●	🔴	●	●	●	●
msvc-2017 openmp release-with-assert host=bs-win2012r2	🔴	●	●	●	●	●	●	●	●
icc-18 no-thread-mpi double openmp mkl cmake-3.9.6 simd=none release host=bs_mic	●	●	●	🔴	●	●	●	●	●
icc-18 openmp opencl clFFT-2.14 cuda-7.5 mpi release simd=avx2_256 host=bs_nix1310	●	●	●	●	●	●	●	🔴	●
gcc-5 openmp simd=avx_128_fma opencl clFFT-2.14 amdappsdk-3.0 host=bs_nix-amd_gpu	●	●	●	●	●	🔴	●	●	●

Pre-submit GROMACS testing: Changes cannot be committed in Gerrit until this matrix is all-green. Tests of a bunch of common compiler/OS/hardware/acceleration options.

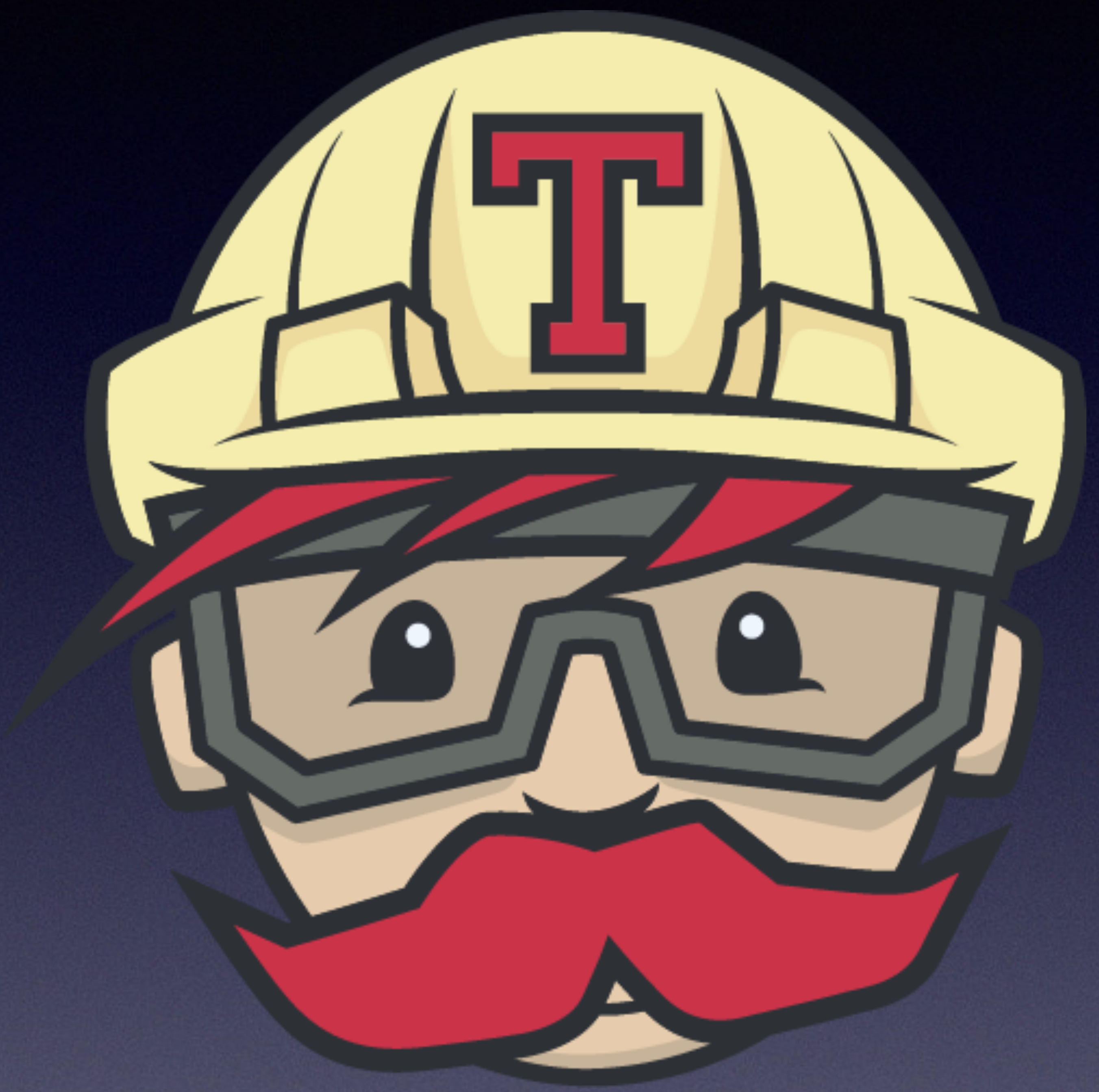


Configuration Matrix	docker-ubuntu-15.04	bs-win2012r2	bs_jetson_tk1	bs_jetson_tx1	bs_mac	bs_mic	bs_nix-amd	bs_nix-amd_gpu	bs_nix-tsan	bs_nix1204	bs_nix1310	bs_nix1404
gcc-5 simd=ARM_NEON release-with-assert host=bs_jetson_tk1	●	●	🔵	●	●	●	●	●	●	●	●	●
gcc-5 simd=ARM_NEON_ASIMD release-with-assert host=bs_jetson_tx1	●	●	●	🔵	●	●	●	●	●	●	●	●
gcc-7 mpi no-openmp fftpack mdrun-only host=bs_nix1404	●	●	●	●	●	●	●	●	●	●	●	🔵
icc-18 simd=avx_256 npme=1 nranks=2 no-openmp double fftpack release host=bs_mic	●	●	●	●	●	🔵	●	●	●	●	●	●
gcc-4.9 gpu cuda-7.0 openmp release-with-assert host=bs_mic	●	●	●	●	●	🔵	●	●	●	●	●	●
clang-4 simd=sse4.1 openmp nranks=1 gpu cuda-8.0 clang_cuda host=bs_nix1204	●	●	●	●	●	●	●	●	●	🔵	●	●
clang-5 openmp simd=avx_128_fma npme=1 nranks=2 mpi host=bs_nix-amd	●	●	●	●	●	●	🔵	●	●	●	●	●
gcc-6 gpu npme=1 nranks=2 opencl cuda-7.5 simd=sse2 release host=bs_nix1310	●	●	●	●	●	●	●	●	●	●	🔵	●
gcc-5 gpu nranks=4 gpu_id=1 cuda-8.0 no-hwloc release-with-assert host=bs_nix1204	●	●	●	●	●	●	●	●	●	🔵	●	●
icc-16 msvc-2017 fftpack simd=avx2_256 release host=bs-win2012r2	●	🔴	●	●	●	●	●	●	●	●	●	●
gcc-7 armhpc-18.2 openmp simd=ARM_NEON_ASIMD release host=bs_overdrive_1000	●	●	●	●	●	●	●	●	●	●	●	●
armclang-18.2 armhpc-18.2 openmp simd=ARM_NEON_ASIMD release-with-assert host=bs_overdrive_1000	●	●	●	●	●	●	●	●	●	●	●	●

Post-submit GROMACS testing: Rare hardware and longer-running performance tests are performed once each patch has been approved.

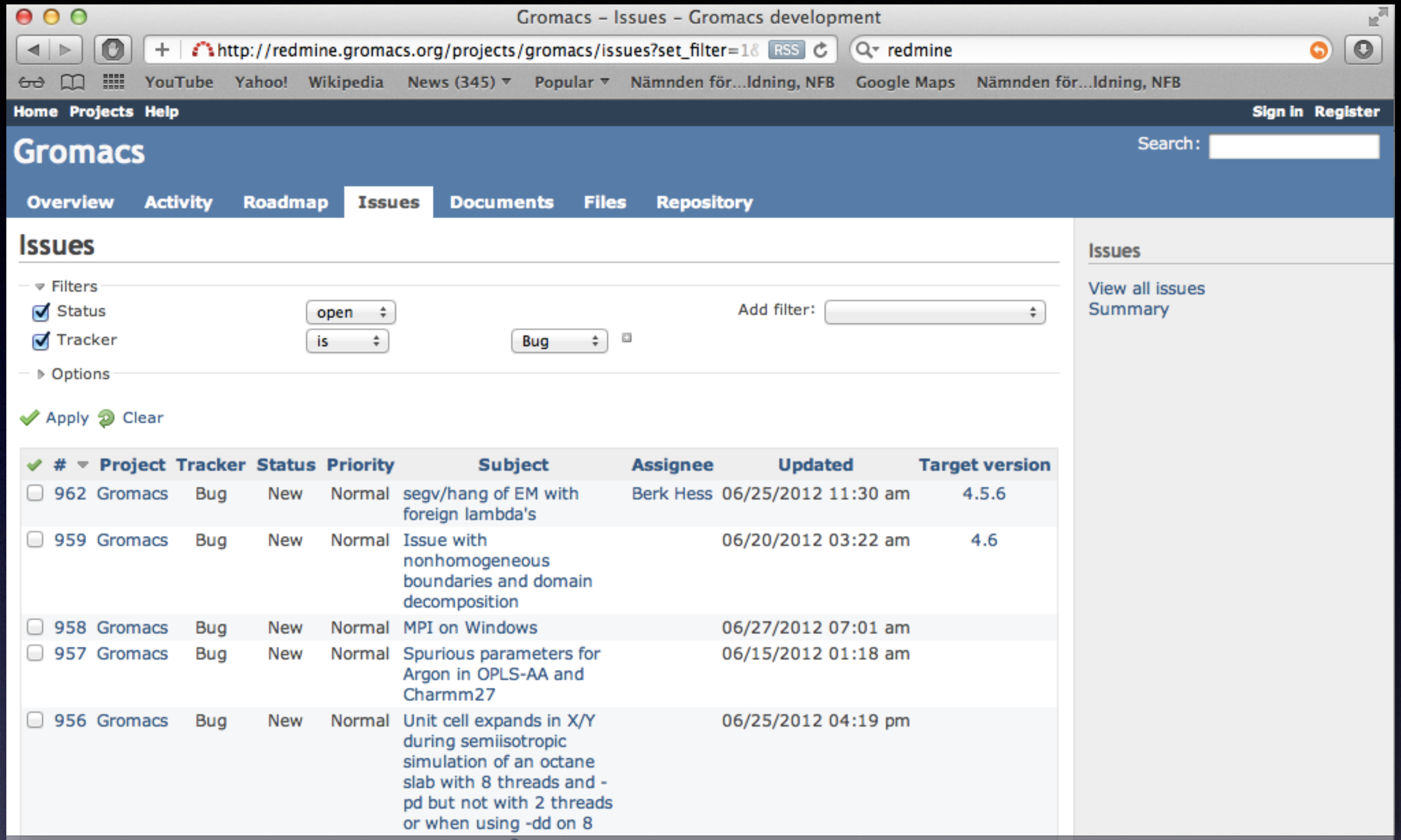
Travis CI

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



Issues

Filters

☑ Status open

☑ Tracker is Bug

Add filter:

Options

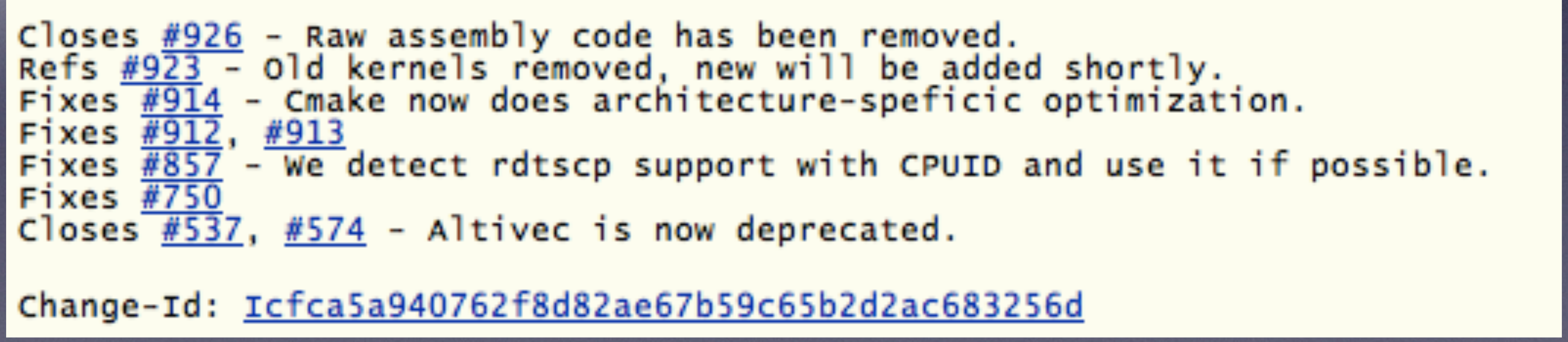
Apply Clear

✓	#	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 semiisotropic 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!

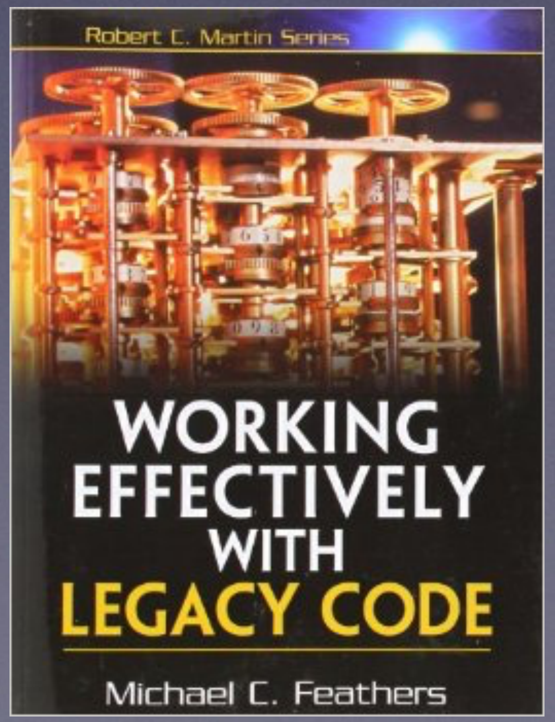


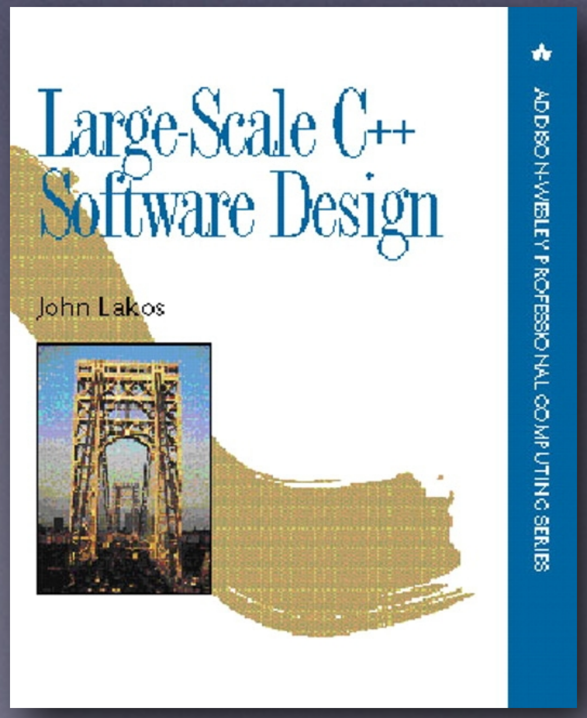
```
Closes #926 - Raw assembly code has been removed.
Refs #923 - Old kernels removed, new will be added shortly.
Fixes #914 - Cmake now does architecture-speficic optimization.
Fixes #912, #913
Fixes #857 - We detect rdtscp support with CPUID and use it if possible.
Fixes #750
Closes #537, #574 - Altivec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

Change-Id: Icfca5a940762f8d82ae67b59c65b2d2ac683256d
```

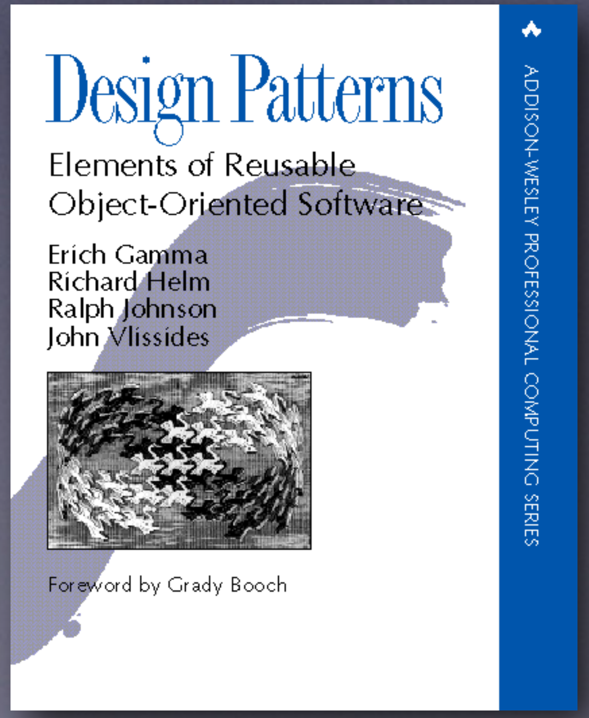
For IHPCSS/software-engineering, we use the simpler integrated issue tracker in GitHub, but this too supports automated referencing e.g. for closing bugs.

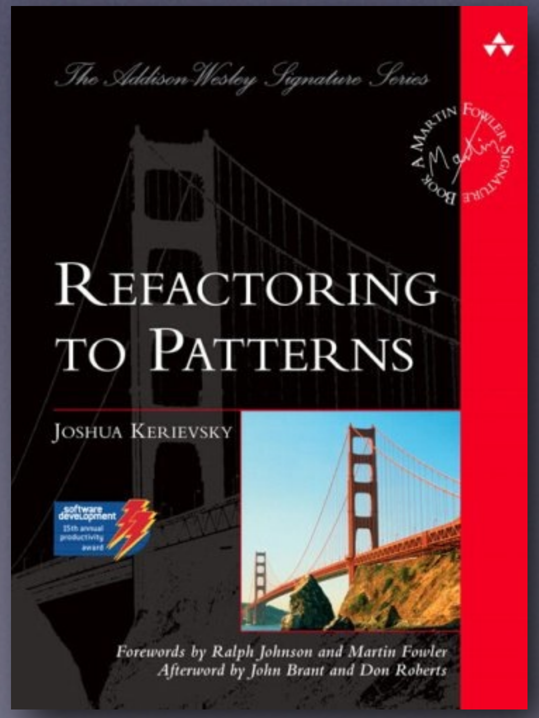
Some good reading

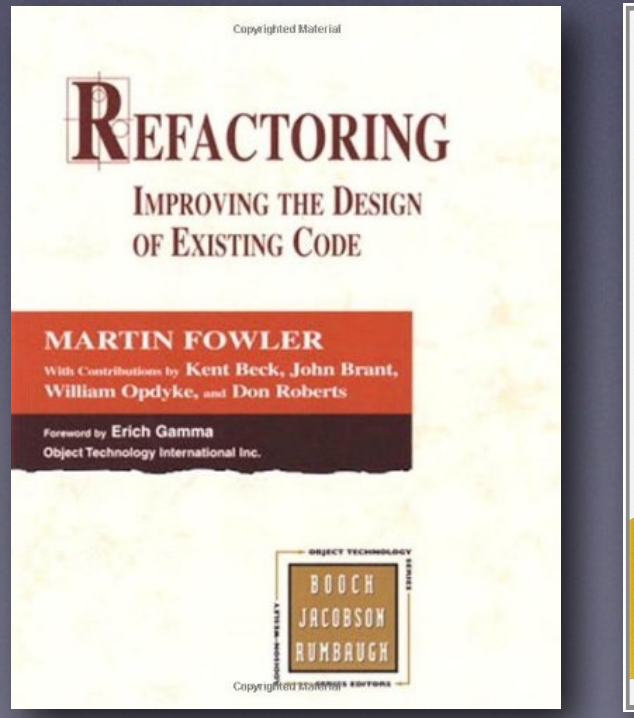
- Working effectively with legacy code [Michael Feathers]
- Large-scale C++ software design [John Lakos]
- Design Patterns - Elements of Reusable Object-oriented software [Gamma, Helm, Johnson, Vlissides] “Gang of four”
- Refactoring to Patterns [Joshua Kerievsky]
- Refactoring - improving the design 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/

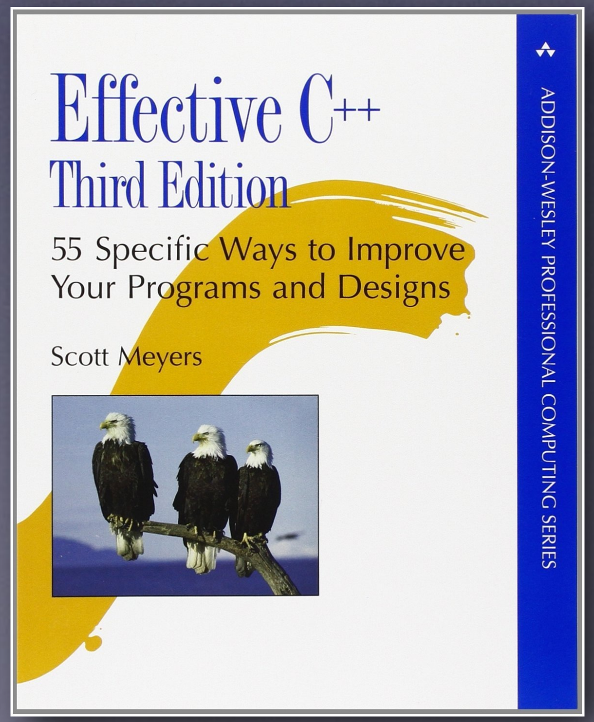



Large-Scale C++ Software Design
John Lakos
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES


Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES


The Addison-Wesley Signature Series
REFACTORING TO PATTERNS
JOSHUA KERIEVSKY
Forewords by Ralph Johnson and Martin Fowler
Afterword by John Brant and Don Roberts


REFACTORING
IMPROVING THE DESIGN OF EXISTING CODE
MARTIN FOWLER
With Contributions by Kent Beck, John Brant, William Opdyke, and Don Roberts
Foreword by Erich Gamma
Object Technology International Inc.
BOOCH JACOBSON RUMBAUGH

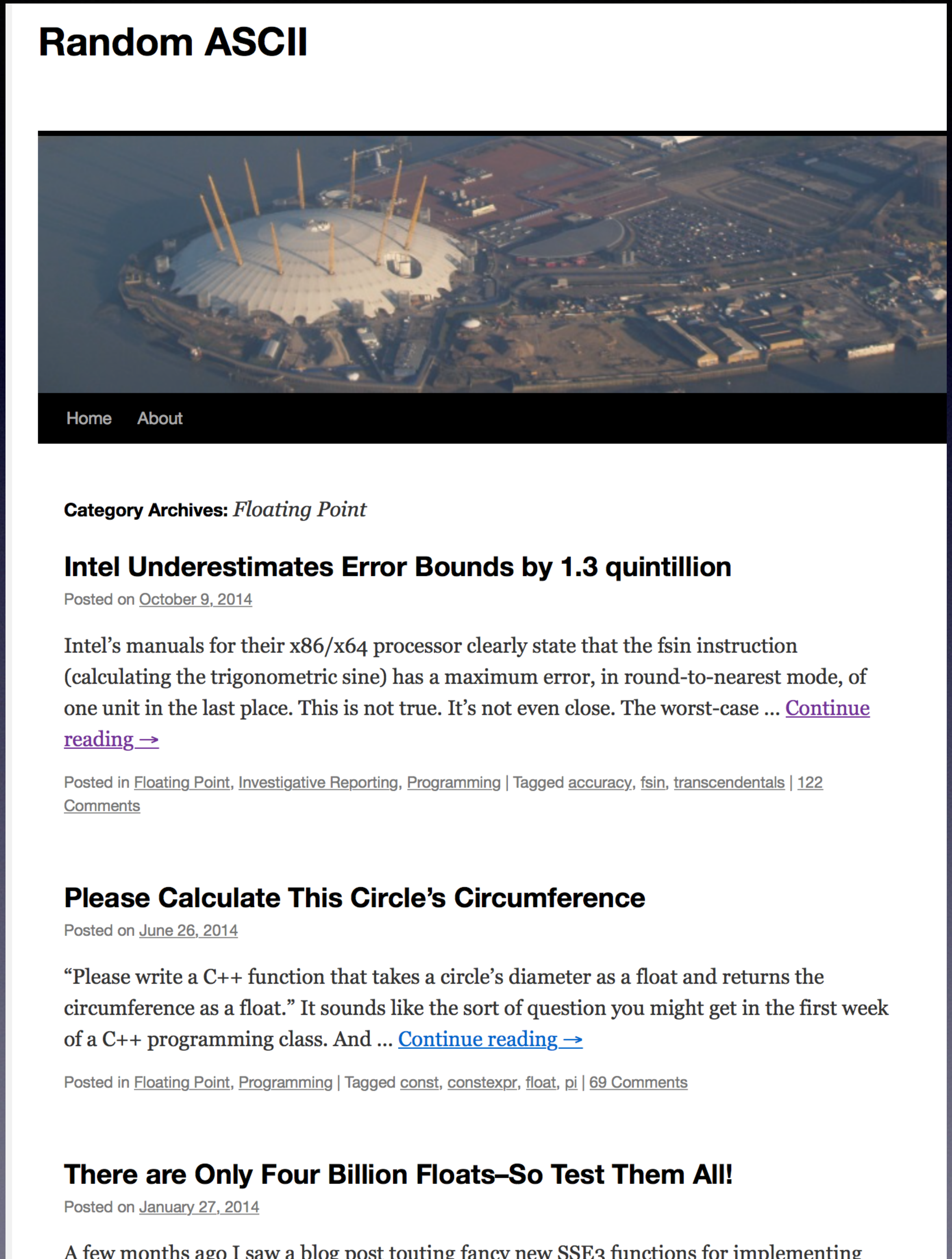

Effective C++
Third Edition
55 Specific Ways to Improve Your Programs and Designs
Scott Meyers
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

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!

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



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!

Posted on January 27, 2014

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

Use the source, Luke

First:

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

There are lots of other open programs out there too.
If you too are free software it is usually OK to reuse
their code in your project (if licenses match)!