HPC Software Engineering

Erik Lindahl

XSEDE/PRACE/RIKEN/Compute Canada HPC Summer School Boulder, CO 2017

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.”

How can you improve software quality?

How do you get a successful career in HPC?

How do you engage in a community?

Open Source & Free Software Development Models

Tools & Recommendations for HPC software engineering

Software maintenance

Handling cultural differences

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

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]

The Picture until early 2011

Source code repository:

CVS

Build Chain:

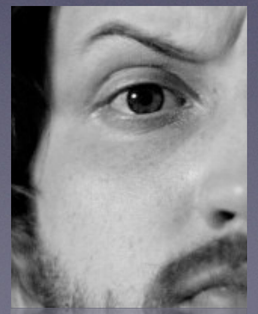
Automake/Autoconf/libtool

Bug Tracking:

Bugzilla

Testing:



Professional modern development tools

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

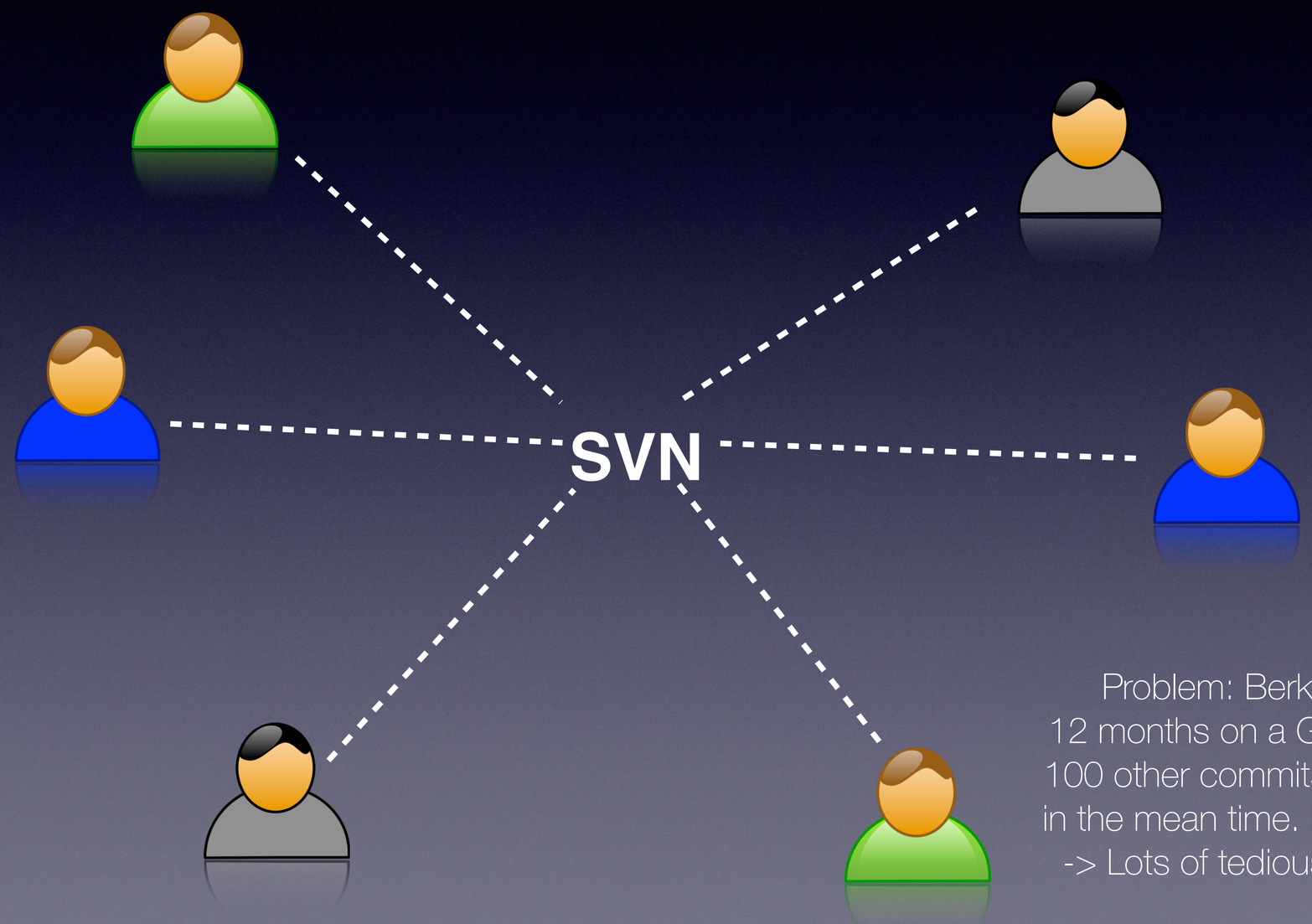
How would you start debugging if the new version crashes?

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?

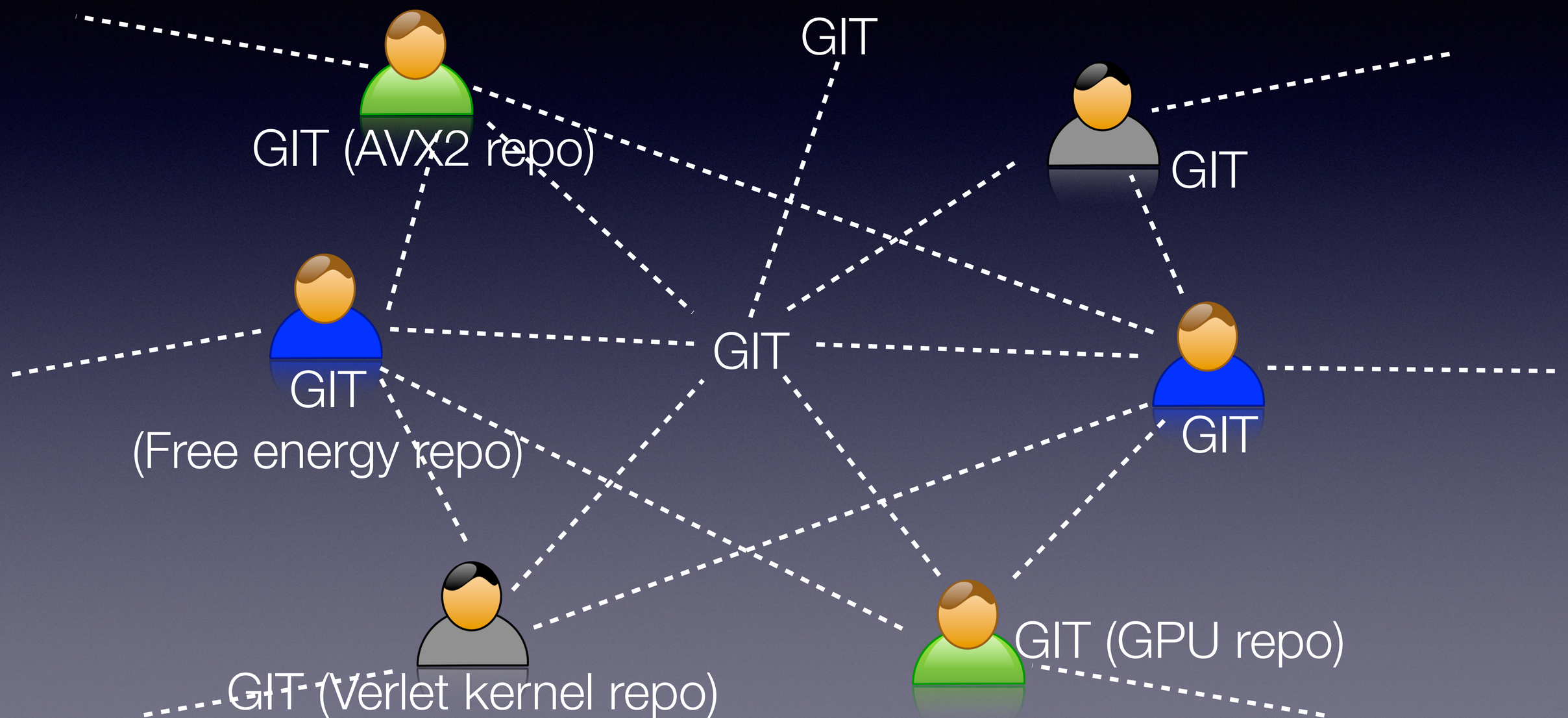
Source code revision control

The CVS/SVN limitation



Problem: Berk has worked
12 months on a GPU branch, but
100 other commits has happened
in the mean time. How to commit?
-> Lots of tedious manual work!

Better source control: GIT



Start your free repo on
github.com or atlassian.com

Local branches

Several repositories, but public & private

Easy to have separate branches for patches

Enable both push and pull patches

No real “master” repository

http://git-scm.com

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

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

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 C++11 flags should be used 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?”
- “What library should I like with for gettimeofday()?”
- “What C backend compiler is used with CUDA-8.0?”
- “What underscore naming standard does this Fortran compiler use?”

CMake: Cross-platform replacement for Autoconf, Automake, Libtool

(instead of ./configure; make; make install)

~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

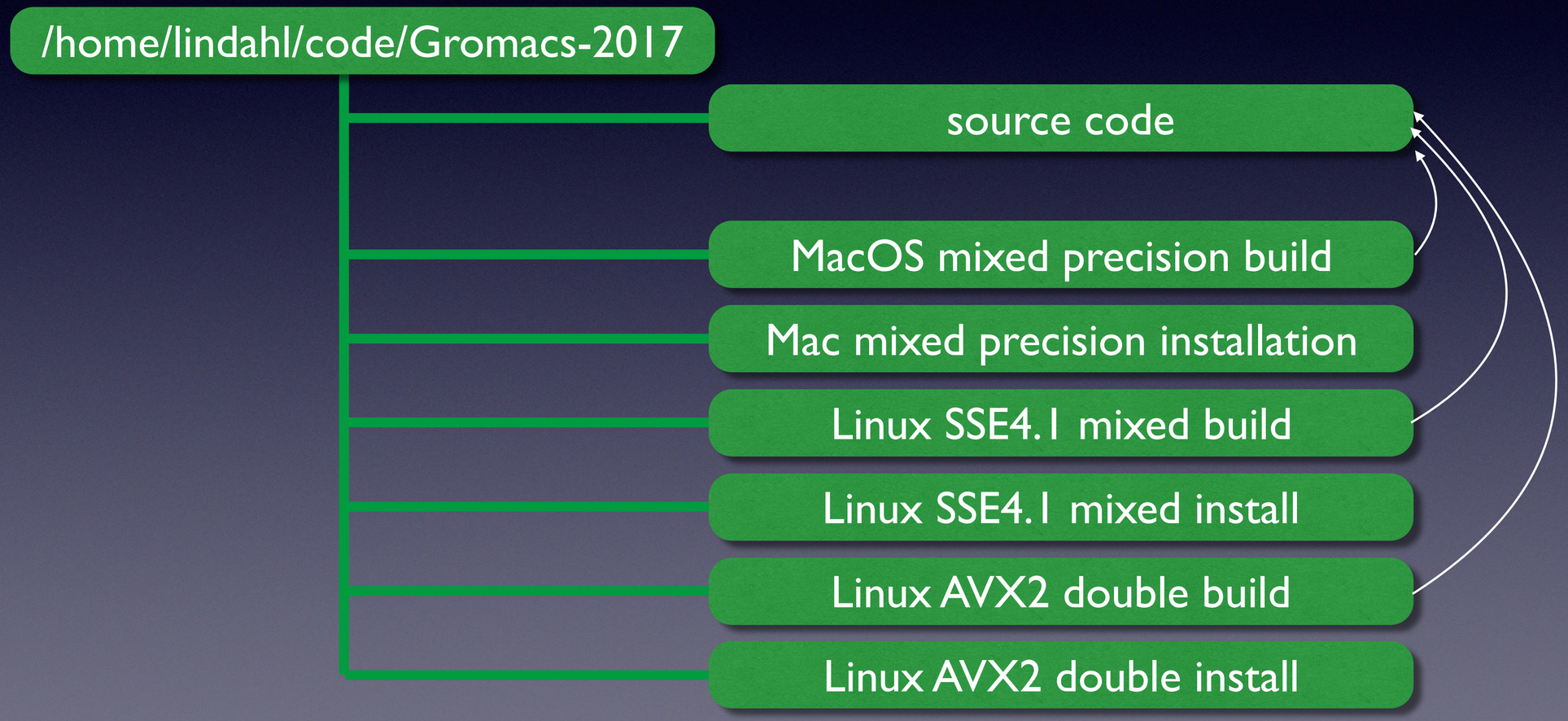
```
MACRO(GMX_TEST_AVX_GCC_MASKLOAD_BUG VARIABLE AVX_CFLAGS)
    IF(NOT DEFINED ${VARIABLE})
        MESSAGE(STATUS "Checking for gcc AVX maskload bug")
        # some compilers like clang accept both cases,
        # so first try a normal compile to avoid flagging those as buggy.
        TRY_COMPILE(${VARIABLE}_COMPILEOK "${CMAKE_BINARY_DIR}"
                    "${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.

Out-of-source builds

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



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

Living with your code for years:

Documentation

Direct source code documentation should stay in the source!

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 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 a humorous set of counter-examples, Google for "how to write unmaintainable code pdf"

Non-source-code documentation: SPHINX (from Python)

sphinx-doc.org

SPHINX
Python Documentation Generator

Home Get it

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.

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:

> *What users say:*
>
> "Cheers for a great tool that actually makes programmers **want** to write documentation!"

- **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.

Examples

Links to documentation generated with Sphinx can be found on the Projects using Sphinx page.

For examples of how Sphinx source files look, use the "Show source" links on all pages of the documentation apart from this welcome page.

You may also be interested in the very nice tutorial on how to create a customized documentation using Sphinx written by the matplotlib developers.

There is a Japanese translation of this documentation, thanks to the Japanese Sphinx user group.

manual.gromacs.org

GROMACS 2016.3 »

Table Of Contents

Next topic

Downloads

This Page

Show Source

Quick search

Go

Welcome to the GROMACS document

Contents:

Documentation for developers

The developer documentation currently consists of two parts:

- A developer guide that provides an overview of the GROMACS c
lines and information on tools used during development.

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
- 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."*

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

Surprise: C++ can be 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);
```

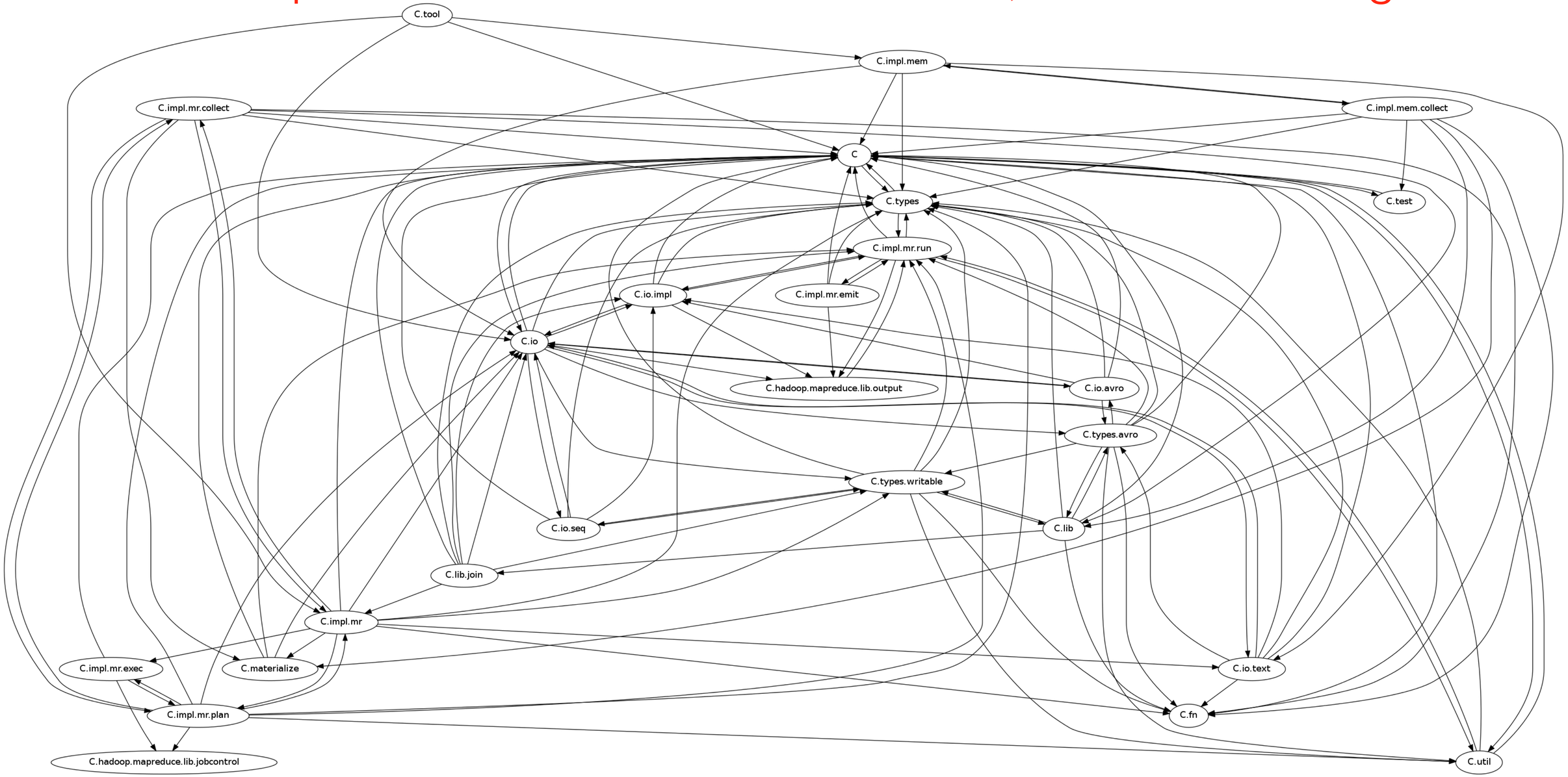
C++11

```
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:

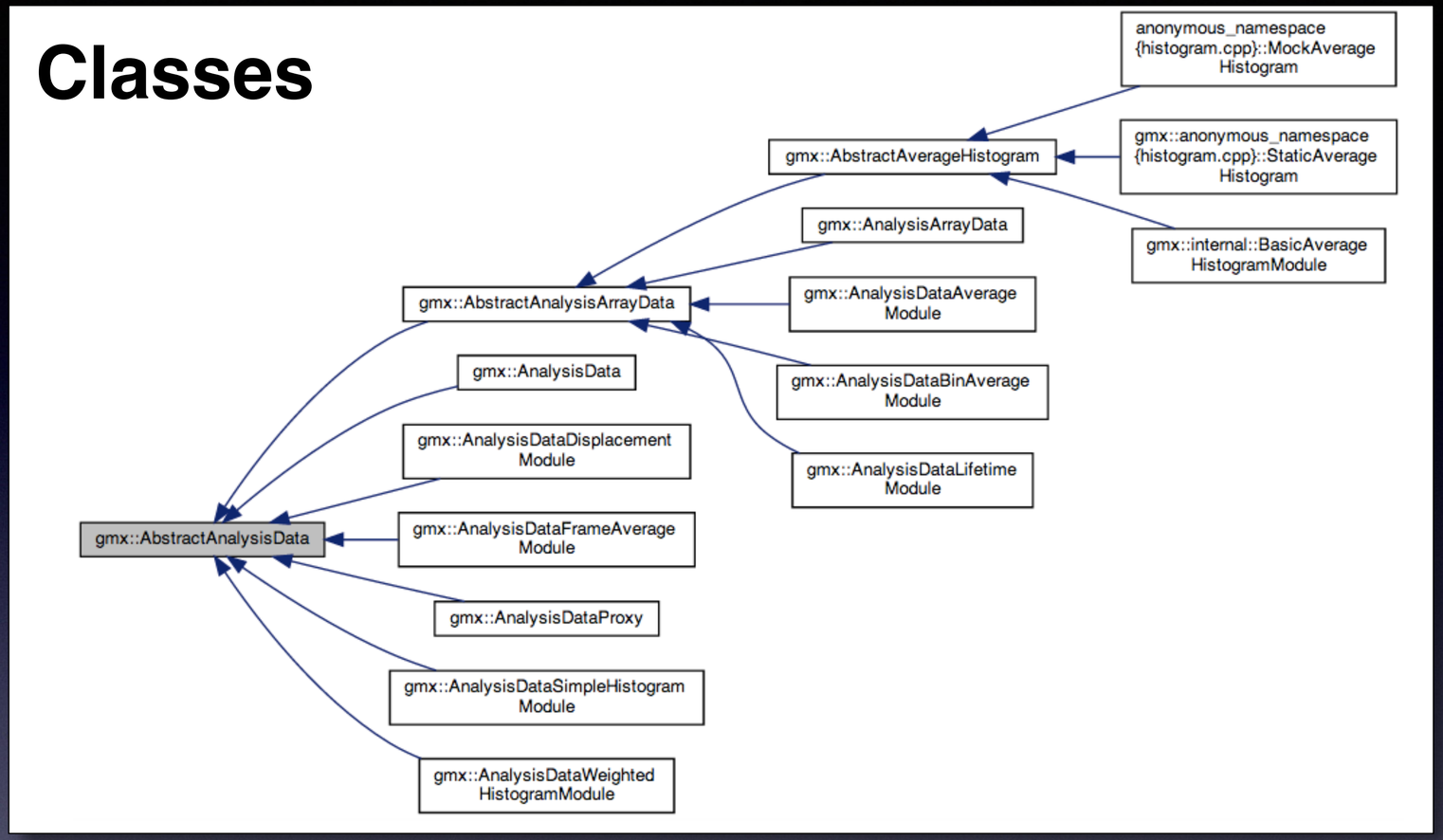
```
extern template int myFunc<2,3>(obj_t obj)
myFunc<2,3>(obj);
```

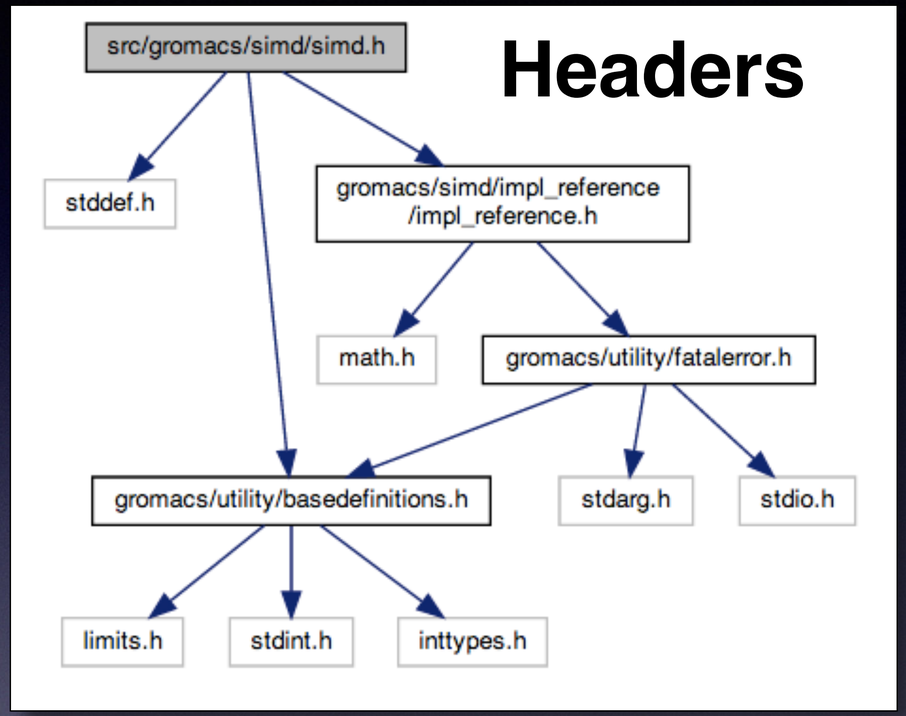
Circular dependencies are bad. If a test fails, where is the bug here?



"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





This is hard, but Doxygen helps you detect it!

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

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

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

g+1 1k

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

Getting Started ¶

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

Who Is Using Google Test?

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

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

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(1.2190973205778108390262 56,
                                              1.1667380278483492350716 23,
                                              1.1689040150044647248250 84), gmx_simd_get_mantissa_r(rSimd_Exp));
#if (defined GMX_SIMD_HAVE_DOUBLE) && (defined GMX_DOUBLE)
    GMX_EXPECT_SIMD_REAL_EQ(setSimdRealFrom3R(1.2412612389523456235632 51,
                                              1.0472947237591238523592 32,
                                              1.8560662047502759573957 34), 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…

Good unit tests should isolate bugs to *tiny* parts of your code
In C++, each method in a class should 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");
}
```

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.

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 | gerrit.gromacs Code Review

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

All Projects Documentation 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

All Projects Documentation Change #, SHA-1, tr:id or owner:email Search

Open Merged Abandoned

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 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: I1fb8eddbe7c8b029dc3686be80f3f083108fc28c
```

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
```

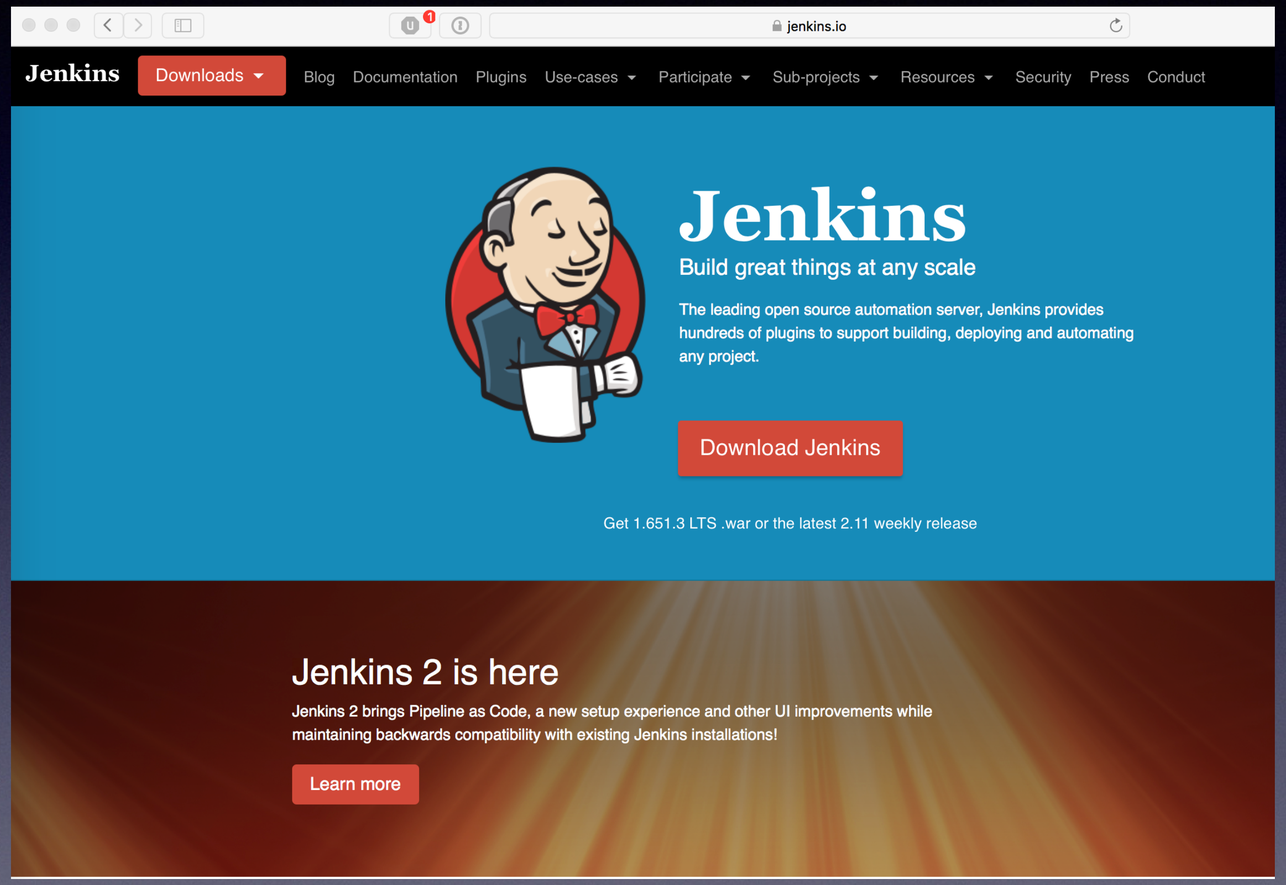
Roland has approved Mark's patch. Anybody can add comments. When two trusted developers say OK, the patch is committed.

Maintaining quality & avoiding breaking stuff

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

Jenkins Continuous Integration

https://jenkins.io



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

Results are integrated into the gerrit review

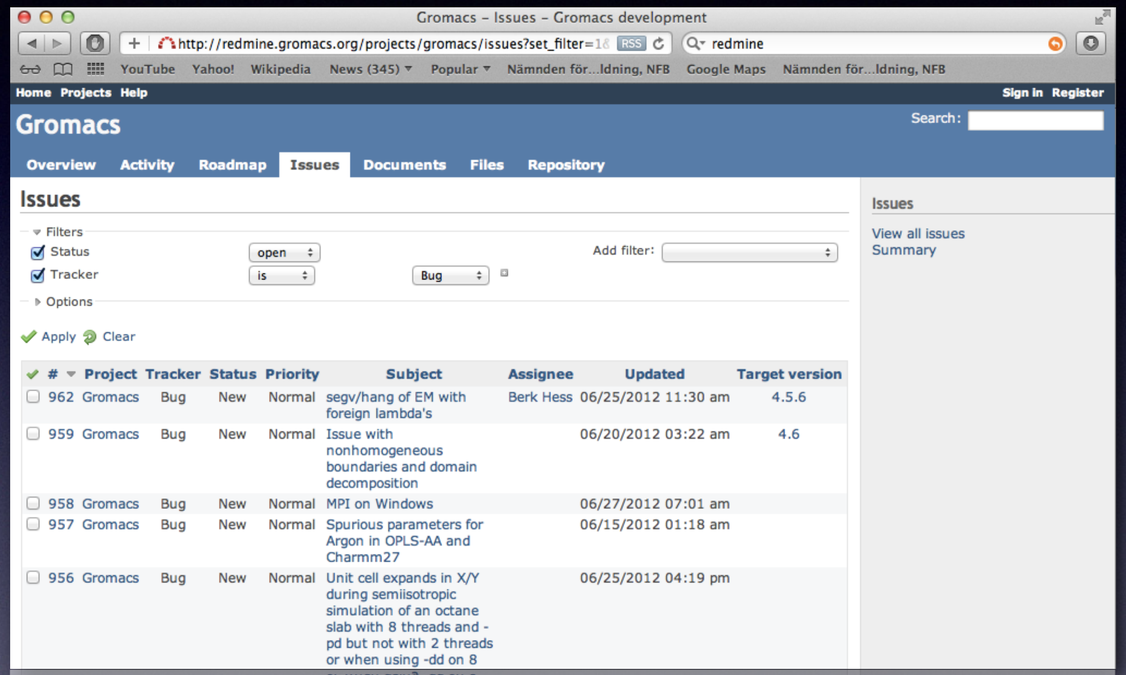
- Catches Cmake build errors
- Catches Google test unit test failures

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

Book-keeping
Bugtracking
Feature tracking
Developer discussions

Redmine issue tracking



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

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

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
```

License Considerations

GPLv2

GPLv3

Dual license?
Exceptions/encryption?

LGPLv2.1

BSD

Academia-friendly?
Business-friendly?
EU-friendly?

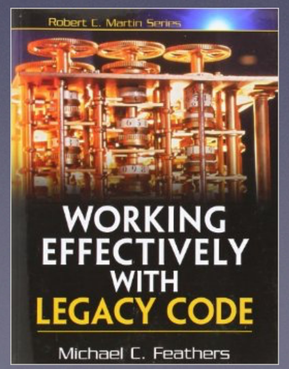
Licenses are tools - decide what you want to achieve, and pick one that helps you!

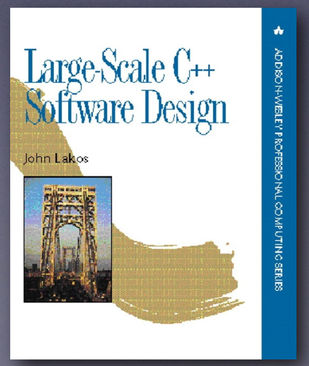
Communities & Cultures

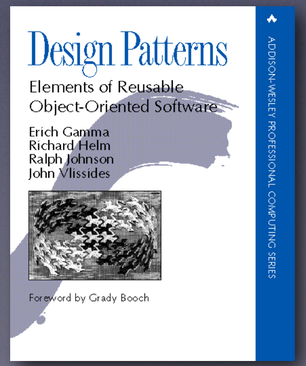
- Engaging in a larger community is a great way to boost your scientific career
- You learn much better by working with others
- Picking up these skills provides a good-job-guarantee
- Open software is critical for open science
- Ultimate meritocracy: We have accepted code from unknown undergraduates, and rejected mine
- Internal culture can be tough/blunt, and politically incorrect. Many scientists and programmers work insanely hard and are passionate about their art.
- Hard to demand respect, but you earn it -
  the currency that builds your karma is high quality code, but when you start out you can also build a good reputation by helping with code review!

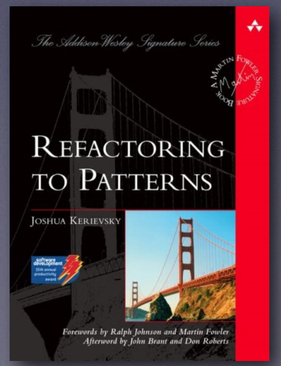
Some good reading

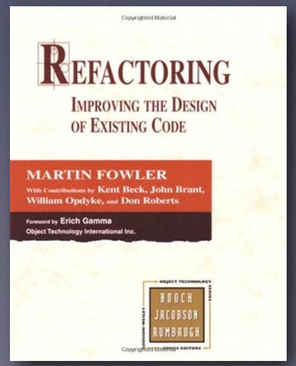
- Working effectively with legacy code [Michael Feathers]
- Large-scale C++ software design [John Lakos]
- Design Patterns - Elements of Reusable Object-oriented software [Gamma, Helm, Johnson, Vlissides] “Gang of four”
- Refactoring to Patterns [Joshua Kerievsky]
- Refactoring - improving the design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

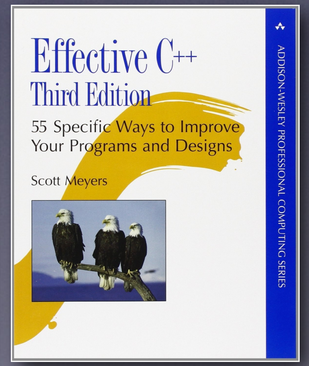












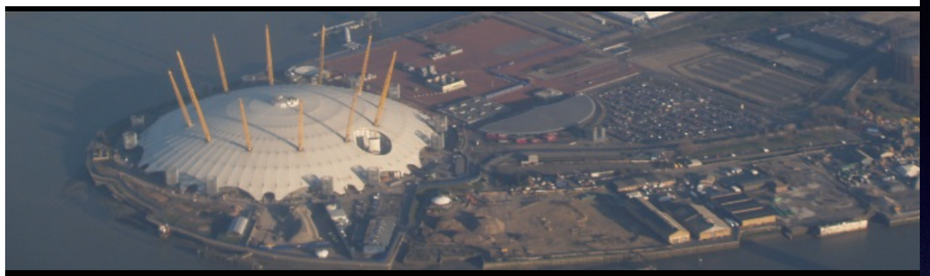
http://randomascii.wordpress.com/category/floating-point/

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



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

Use the source, Luke

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)!