The view from a Standard Library Implementer

Marshall Clow

C++ Alliance

May 9, 2019

About me

I have been involved in Boost since 2001. I am the author of Boost.Algorithm, the maintainer of a few other libraries, a release manager, mailing list moderator, and a former member of the Boost Steering Committee.

I have been working on LLVM for nine years, and on libc++ for about seven, and I am the "code owner" for libc++.

I am also the chairman of the Library Working Group of the C++ Standards Committee.

I work for the C++Alliance, a US-based non-profit organization.

Contact info:

1. Email: mclow.lists@gmail.com
2. Slack: marshall
3. IRC: mclow

Outline of this talk

1. Working on the standard library
2. How does this compare with Boost?
3. General Comments on library work
4. Boost and the Standard library

Working on the standard library
Challenges and Benefits

Challenges for the standard library

1. Has to be everything for everyone
2. Has to provide the same functionality on different systems.
3. ABI Stability
4. Hostile Environment (macros, etc)

Advantages for the standard library

1. Written Specification
2. `__names`
3. Compiler Intrinsics

How does this compare with Boost?

Challenges for Boost

Boost libraries are sometimes considered "proving grounds" or "staging areas" for future standardization. Many of the same concerns for the standard libraries also apply to boost libraries.

1. Hostile Environment (macros, etc)
2. Has to be everything for everyone
3. Support for old (buggy) compilers
4. Has to provide the same functionality on different systems.

Advantages for Boost

1. ABI Stability
2. Velocity
3. (Comparatively) easy to add new libraries
4. "Non-mainstream" topics

General Comments on library work

Things a library's users might care about

1. User code stability
2. Readable documentation
3. Ability to Debug
4. Porting to another platform

Debugability

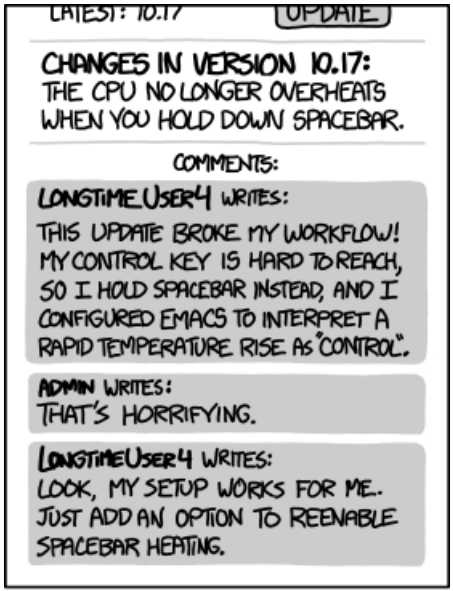
1. Static Assertions
2. Assertions
3. Debug Mode
4. Performance at -O0

Hyrum's law

Hyrum's law states:

> *With a sufficient number of users of an API,*
> *it does not matter what you promise in the contract:*
> *all observable behaviors of your system*
> *will be depended on by somebody.*

I agree that this is true, and at the same time reject the implications.



EVERY CHANGE BREAKS SOMEONE'S WORKFLOW.

What do you do when you don't know what alternative to choose?

Suppose you are implementing a library, and you have two (or more) choices about how to implement a feature. Each has different advantages and/or disadvantages.

How do you know which one to pick?

1. Make your best choice
2. Gather evidence
3. Listen to your users
4. Re-evaluate your decision based on the feedback

Tools to improve libraries

There are a ton of tools available today to help you improve your library. Use them!

1. Test frameworks
2. Static analyzers
3. Dynamic analyzers
4. Refactoring tools
5. Code Coverage tools
6. Documentation generators
7. Fuzzing tools

The general attitude towards libraries

If there's a demonstrated code generation benefit, performance trumps readability.
This is especially true in a std::lib implementation. Very few people care about readability or maintainability of std::lib code and tons of people care about its performance.

– Howard Hinnant

Boost and the Standard library

Boost can produce libraries that probably shouldn't be in the standard

Bryce gave four reasons for putting something in the standard library

1. Generally useful - vocabulary types, etc.
2. Wrappers for non-portable facilities (filesystem, threading, atomic, networking)
3. Hard to get right - `shared_ptr`
4. Things that the compiler has to provide

What kind of things **do not** belong in the standard library?

Boost can produce things that **might** go into the standard library

Bryce gave four reasons for putting something in the standard library

1. Generally useful - vocabulary types, etc.
2. Wrappers for non-portable facilities
3. Hard to get right - `shared_ptr`
4. Things that the compiler has to provide

What kind of things **should be** in the standard library?

Thank you

Links

1. C++Alliance: https://www.cppalliance.org
2. http://www.hyrumslaw.com
3. http://howardhinnant.github.io/coding_guidelines.html