What's in a Name?

Document number: P1921R0
Date: 2019-10-07
Project: Programming Language C++, Library Evolution Working Group
Reply-to: Nevin "☺" Liber, nliber@anl.gov

Table of Contents

Introduction **1**

Motivation and Scope **1**

Impact on the Standard **2**

Design Decisions **2**

Technical Specifications **2**

Acknowledgements **3**

References **3**

Introduction

Naming is very important, but our current process is both inefficient and doesn't necessarily give us a better chance at coming up with great names. We can do better.

Motivation and Scope

Naming is very important, and naming is hard. Here are some of the reasons why:

- They stick around forever.
- They set usage expectations.
- They can affect the design of a type. For instance, P0201 was originally proposed as cloned_ptr<T>, but subsequent discussion discovered that *pointer* wasn't the correct shape and the type ultimately evolved into polymorphic_value<T>.

However, the process for naming, commonly referred to as bike shedding, is ad hoc. It is an inefficient use of our face to face time at committee meetings. It doesn't necessarily give us a better chance at great names:

- We don't have time to reflect and think about the names being picked. We typically spend a few minutes brainstorming names, put a bunch up to see what sticks, and pick one. Trusting a gut reaction is not a good way to pick a longstanding name.

- It has been suggested that "If you want to participate in naming, sit in LEWG". That is just not practical as no one knows when LEWG will suddenly partake in a naming discussion. For instance, unique_function did not change its name to any_invocable until the face to face discussion of the sixth revision of the paper.
- "We can change the name later." This both puts more risk into a paper being accepted into the Working Draft as well as the bar for change is much higher once a paper is accepted into the Working Draft, requiring a later (in the shipping cycle) and larger (in terms of participants) discussion.

Impact on the Standard

No direct impact.

Design Decisions

To formalize this, here is the first pass describing a new naming process:

1. All name changes start as papers in the mailing. This puts the committee as a whole on notice that a naming discussion will take place, so interested parties can be in the room.
2. Email discussion between publication and before the meeting. This is to avoid making naming a bottleneck that takes multiple meetings between giving notice and actually discussing a name change.
3. Hard cutoff of the email discussion a few days before the meeting. This gives interested parties time to ruminate over the names being discussed.
4. If new names are discovered during the email discussion that some would prefer, they should be provided encouragement to quickly write a single paper with the alternate names and reasons behind them (no more lists of names to see what sticks), similar to papers that are encouraged to be written during the meeting. It is not required that the author of the original proposal write such a paper, as sometimes a name change is extensive enough that the author may no longer wish to support such a proposal.
5. During the meeting, briefly discuss the new names in the various papers (as we would discuss any other paper) and take a straw poll to pick one.

Technical Specifications

This could be added to SD-8 or new different standing document should we so choose to document our process.

Acknowledgements

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration) responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering, and early testbed platforms, in support of the nation's exascale computing imperative. Additionally, this research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.

References

P0201 A polymorphic value-type for C++
P0288 any_invocable
SD-8 Standard Library Compatibility
N4830 Working Draft, Standard for Programming Language C++