C++ Club Meeting Notes

Gleb Dolgich

2018-07-05

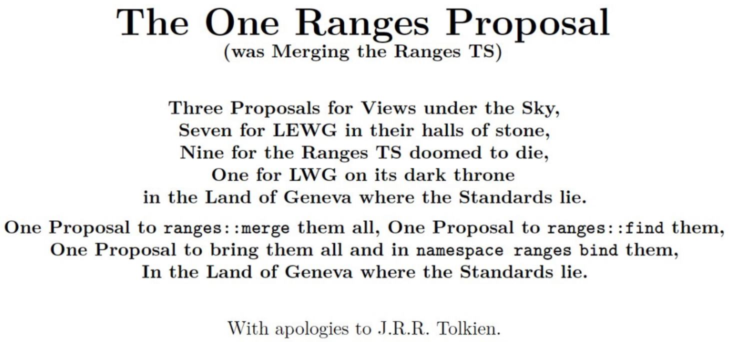
Rapperswil reports

- Corentin
- Herb Sutter
  - *"I sometimes delay my trip report until the post-meeting mailing is available so that everyone can see the latest papers"*
  - On contracts: *"Having first-class contracts support it is the first major "step 1" of reforming error handling in C++ and applying 30 years' worth of learnings."*
  - On `std::bad_alloc`: *"Consider handling heap exhaustion (out-of-memory, OOM) differently from other errors."*
  - On feature test macros: *"Some experts still disagree, and we respect their views, but in my view these feature test macros are an important and pragmatic help to improve the speed of adoption of new standard C++ features."*
  - Reddit thread
- J.Daniel Garcia's Contracts presentation and slides

Post-Rapperswil mailing

- Mailing 2018-06

The One Ranges Proposal, by Casey Carter



The One Ranges Proposal

(was Merging the Ranges TS)

Three Proposals for Views under the Sky,
Seven for LEWG in their halls of stone,
Nine for the Ranges TS doomed to die,
One for LWG on its dark throne
in the Land of Geneva where the Standards lie.

One Proposal to `ranges::merge` them all, One Proposal to `ranges::find` them,
One Proposal to bring them all and in `namespace ranges` bind them,
In the Land of Geneva where the Standards lie.

With apologies to J.R.R. Tolkien.

The tightly-constrained design space of convenient syntaxes for generic programming

- Corentin – there is a questionnaire at the end

  The compiler (and by extension, tools) needs no syntax whatsoever to distinguish concepts, types, values, type-concepts, value-concepts.

  Some people like syntax so much, Bjarne calls them the “syntax people”.

- Combined proposal: “Yet another approach for constrained declarations”

James McNellis on C++ Core Guidelines

- Issue: The entirety of “NL: Naming and layout rules” should be removed
- Reddit discussion

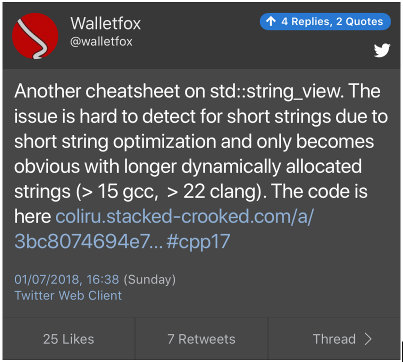
Forward declarations

- Google C++ Guidelines

East Const Central

- Jon Kalb

Twitter: `string_view` lifetime



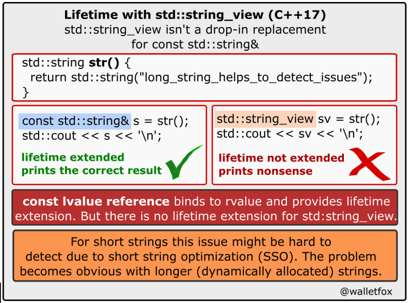
Walletfox
@walletfox

4 Replies, 2 Quotes

Another cheatsheet on std::string_view. The issue is hard to detect for short strings due to short string optimization and only becomes obvious with longer dynamically allocated strings (> 15 gcc, > 22 clang). The code is here coliru.stacked-crooked.com/a/3bc8074694e7... #cpp17

01/07/2018, 16:38 (Sunday)
Twitter Web Client

25 Likes | 7 Retweets | Thread



Lifetime with std::string_view (C++17)
std::string_view isn't a drop-in replacement for const std::string&

```
std::string str() {
  return std::string("long_string_helps_to_detect_issues");
}
```

```
const std::string& s = str();
std::cout << s << '\n';
```

lifetime extended
prints the correct result ✓

```
std::string_view sv = str();
std::cout << sv << '\n';
```

lifetime not extended
prints nonsense ✗

const lvalue reference binds to rvalue and provides lifetime extension. But there is no lifetime extension for std:string_view.

For short strings this issue might be hard to detect due to short string optimization (SSO). The problem becomes obvious with longer (dynamically allocated) strings.

@walletfox

Namespace tricks

- The Old New Thing
- Follow-up: My namespace importing trick imported the same three namespaces into each top-level namespace, yet it worked?

Be Mindful with Compiler-Generated Move Constructors

- Post
    - Reddit

function2

Improved and configurable drop-in replacement to `std::function` that supports move only types, multiple overloads and more.

- Website
- Code

Quote

Tom Cargill:

> *The first 90% of the code accounts for the first 90% of the development time. The remaining 10% of the code accounts for the other 90% of the development time.*