

C++ Club UK

Gleb Dolgich

2019-10-24

Mailing

- Proposal of `std::upto`, `std::indices` and `std::enumerate`
- A Unified Executors Proposal for C++

https://www.reddit.com/r/cpp/comments/dhm138/wg21_the_201910_mailing_is_now_available/

- Oh boy (RX) and doubling down
- Oh boy (Graphics)

Corentin: Executors are the least useful part of "A Unified Executors Proposal"

Eric Niebler:

It makes me happy to see the power of the Scheduler/Sender/Receiver recognized.

Much of traditional Rx assumes runtime polymorphism and garbage collection / ref counting. That's obviously a poor fit for C++. The sender/receiver design skews toward static polymorphism w/Generic interfaces, and explicit memory mgmt.

The factoring of `submit` into `connect/start` gives more flexible ownership semantics, and aligns the design conceptually with coroutines, making coroutines an efficient way of expressing sender/receiver.

Once you introduce concurrency with non-overlapping scopes, the lifetime of your async operation state no longer corresponds to a simple C++ scope. That's why explicit memory mgmt and ownership become issues where they wouldn't be otherwise.

One of coroutines great strengths is that they let us map async lifetime back to C++ scopes. Under the hood, the coroutine is carved up into callbacks, and the operation state (coroutine frame) is getting managed explicitly, but all that gets hidden from you by the `coro` types.

`Sender/receiver` is a library reification of this separation. You might argue (and some have) that we should just drop `sender/receiver` and use coroutines everywhere for everything “async”. That’s appealing, but goes too far. Coroutines, for all their value, come up short sometimes.

C++ has a long sad history of giving old things new names.

Map and fold would have been better than transform and accumulate, and don't get me started on vector.

Sender/receiver share some pedigree with Observable/Observer, but they really are different beasts ... by necessity.

The intended purpose of sender/receiver is to be a base lingua franca for all C++ async libraries. I expect end-users to interface at much higher levels of abstraction in their own code.

<https://quuxplusone.github.io/blog/2019/07/03/announcing-coro-examples/>

<https://github.com/Quuxplusone/coro>

wg21.link cheatsheet

wg21.link - WG21 redirect service.

Usage:

```
wg21.link/nXXXX
wg21.link/pXXXX
wg21.link/pXXXXrX
  Get paper.
```

```
wg21.link/standard
  Get working draft.
```

```
wg21.link/cwgXXX
wg21.link/ewgXXX
wg21.link/lwgXXX
wg21.link/lewgXXX
wg21.link/fsXXX
wg21.link/editXXX
  Get issue.
```

```
wg21.link/index.json
wg21.link/index.ndjson
wg21.link/index.txt
wg21.link/specref.json
  Get everything.
```

```
wg21.link/
  Get usage.
```

```
wg21.link/<something else>
  Get 404.
```

Sources:

- <http://www.open-std.org/jtc1/sc22/wg21/docs/papers/{2002..2018}/>,
- http://www.open-std.org/jtc1/sc22/wg21/docs/cwg_{active,defects,closed}.html,
- <http://cplusplus.github.io/EWG/ewg-{active,complete,closed}.html>,
- <http://cplusplus.github.io/LWG/lwg-{active,defects,closed}.html>,
- <http://issues.isocpp.org/>,
- <https://github.com/cplusplus/draft/tree/master/papers>,
- [https://github.com/cplusplus/draft issues](https://github.com/cplusplus/draft/issues) and [pull-requests](https://github.com/cplusplus/draft/pull-requests),
- the private wiki of the C++ committee, and
- a few manual additions.

<https://devblogs.microsoft.com/cppblog/addresssanitizer-asan-for-windows-with-msvc/>

https://www.reddit.com/r/cpp/comments/d6k7mt/address_sanitizer_is_coming_to_msvc/

https://www.reddit.com/r/cpp/comments/dm1emb/addresssanitizer_asan_for_windows_with_msvc/

Are there any memory safety libraries for C++?

https://www.reddit.com/r/cpp/comments/d0hguz/are_there_any_memory_safety_libraries_for_c/

<https://github.com/duneroadrunner/SaferCPlusPlus/>

<https://github.com/deplinenoise/ig-memtrace>

MemTrace is a memory debugging tool developed internally at Insomniac Games.

<https://github.com/ivmai/bdwgc>

The Boehm-Demers-Weiser conservative C/C++ Garbage Collector (libgc, bdwgc, Boehm-gc) <https://www.hboehm.info/gc/>

MSVC versions are crazy

MSC	1.0	_MSC_VER == 100	
MSC	2.0	_MSC_VER == 200	
MSC	3.0	_MSC_VER == 300	
MSC	4.0	_MSC_VER == 400	
MSC	5.0	_MSC_VER == 500	
MSC	6.0	_MSC_VER == 600	
MSC	7.0	_MSC_VER == 700	
MSVC++	1.0	_MSC_VER == 800	
MSVC++	2.0	_MSC_VER == 900	
MSVC++	4.0	_MSC_VER == 1000	(Developer Studio 4.0)
MSVC++	4.2	_MSC_VER == 1020	(Developer Studio 4.2)
MSVC++	5.0	_MSC_VER == 1100	(Visual Studio 97 version 5.0)
MSVC++	6.0	_MSC_VER == 1200	(Visual Studio 6.0 version 6.0)
MSVC++	7.0	_MSC_VER == 1300	(Visual Studio .NET 2002 version 7.0)
MSVC++	7.1	_MSC_VER == 1310	(Visual Studio .NET 2003 version 7.1)
MSVC++	8.0	_MSC_VER == 1400	(Visual Studio 2005 version 8.0)
MSVC++	9.0	_MSC_VER == 1500	(Visual Studio 2008 version 9.0)
MSVC++	10.0	_MSC_VER == 1600	(Visual Studio 2010 version 10.0)
MSVC++	11.0	_MSC_VER == 1700	(Visual Studio 2012 version 11.0)
MSVC++	12.0	_MSC_VER == 1800	(Visual Studio 2013 version 12.0)
MSVC++	14.0	_MSC_VER == 1900	(Visual Studio 2015 version 14.0)
MSVC++	14.1	_MSC_VER == 1910	(Visual Studio 2017 version 15.0)
MSVC++	14.11	_MSC_VER == 1911	(Visual Studio 2017 version 15.3)
MSVC++	14.12	_MSC_VER == 1912	(Visual Studio 2017 version 15.5)
MSVC++	14.13	_MSC_VER == 1913	(Visual Studio 2017 version 15.6)
MSVC++	14.14	_MSC_VER == 1914	(Visual Studio 2017 version 15.7)
MSVC++	14.15	_MSC_VER == 1915	(Visual Studio 2017 version 15.8)
MSVC++	14.16	_MSC_VER == 1916	(Visual Studio 2017 version 15.9)

A de-facto standard C++ project layout, by Colby Pike
<vectorofbool@gmail.com>

- [Reddit post 1](#)
- [Reddit post 2](#)
- [Pitchfork GitHub repo](#)
- [Pre-paper](#)
- [Bloomberg BDE physical code organization](#)

Closing the gap: cross-language LTO between Rust and C/C++

<http://blog.llvm.org/2019/09/closing-gap-cross-language-lto-between.html>

Reddit descended into an irrelevant but heated discussion on the term "C/C++".

What's the difference between "STL" and "C++ Standard Library" ?

<https://stackoverflow.com/questions/5205491/whats-the-difference-between-stl-and-c-standard-library>

https://www.reddit.com/r/cpp/comments/c90sxa/whats_the_difference_between_stl_and_c_standard/

STL is a maintainer of MSVC's implementation of the C++ Standard Library.

<https://www.scapix.com/introduction/>

<https://github.com/scapix-com/scapix>

https://www.reddit.com/r/cpp/comments/cjvc1u/automatic_c_bindings_for_various_languages/

https://www.reddit.com/r/cpp/comments/ckf44x/scapix_java_link_modern_c17_jni_wrapper_library/

Swish

- <https://github.com/lamarr/swish>
- https://www.reddit.com/r/cpp/comments/c8q0wh/a_modern_c_17_http_client_library_for_humans/

CPR

- <https://github.com/whoshuu/cpr>
- <https://whoshuu.github.io/cpr/>

<https://github.com/martinmoene/span-lite>

A C++20-like span for C++98, C++11 and later in a single-file header-only library.

- Implementation <https://github.com/kokkos/mdspan> (BSD 3-Clause)
 - Intro <https://github.com/kokkos/mdspan/wiki/A-Gentle-Introduction-to-mdspan>
 - r/cpp https://www.reddit.com/r/cpp/comments/cl127i/mdspan_productionquality_reference_implementation/
- Kokkos <https://github.com/kokkos/kokkos>
- Multi-dimensional strided array views in Magnum <https://blog.magnum.graphics/backstage/multidimensional-strided-array-views/>
- P0009R9 **mdspan**: A Non-Owning Multidimensional Array Reference <http://wg21.link/p0009r9>
- CppCast with Bryce Adelstein Lelbach <https://cppcast.com/bryce-lelbach-mdspan/>

Enums, warnings, and default

<https://brevzin.github.io/c++/2019/08/01/enums-default/>

https://www.reddit.com/r/cpp/comments/cubah9/enum_switch_warnings/

GCC & Clang:

```
1 | -Wswitch-enum
```

Reddit

- PyTorch <https://pytorch.org/features> -- has a pure C++ front end <https://pytorch.org/cppdocs/>
- TensorFlow for C++
https://www.tensorflow.org/api_docs/cc
- Shogun <https://www.shogun.ml/>

The sad history of Unicode printf-style format specifiers in Visual C++

<https://devblogs.microsoft.com/oldnewthing/20190830-00/?p=102823>

- [Reddit](#)

Introducing Magnum Python Bindings

<https://blog.magnum.graphics/announcements/introducing-python-bindings/>

<https://github.com/pybind/pybind11>

AnyDuck : A Value Type Erased Type

Steve Downey: <https://www.sdowney.org/2019/07/anyduck-a-value-type-erased-type/>

Template meta-programming: Some testing and debugging tricks

<https://cukic.co/2019/02/19/tmp-testing-and-debugging-templates/>

<https://www.nextptr.com/tutorial/ta1193988140/how-cplusplus-using-or-aliasdeclaration-is-better-than-typedef>

<https://github.com/berkeley-container-library/bcl>

https://www.reddit.com/r/cpp/comments/d9xncc/c_interview_questions/

Sturgeon's Law:

90% of everything is crap.