C++ Club Meeting Notes

Gleb Dolgich

2018-04-05

Jacksonville 2018 – C++20

- Trip reports
  - Timur Doumler, JetBrains
  - Going Native 65, Microsoft
- Previously
  - Vittorio Romeo
  - Guy Davidson
  - using std::cpp
  - CppCast with Patrice Roy
  - Botond Ballo / Reddit thread

Jacksonville 2018 – C++20 (cont.)

- The Plan (according to Herb Sutter – Thanks Bjarne):
  - Executors: TS in the C++20 timeframe, standard in C++23
  - Networking: C++23 (delayed by Executors)
  - Coroutines: C++20
  - Modules: Partially in C++20 with more in C++23 (blame Google)
  - Contracts: C++20
  - Reflection: TS in C++20 timeframe, standard in C++23
  - Ranges: Core in C++20, cool stuff in C++23

April Fools, C++ edition

No, I’m not going to link to that.

STL on Twitter

MSVC's STL is 99% C++17 feature complete! In VS 2017 15.7, we're shipping Filesystem, Parallel Algorithms (all signatures available, many actually parallelized, more later), `constexpr char_traits` (for `string_view`), Special Math, `hypot(x, y, z)`, `launder()`, and Deduction Guides.

15.7 will contain a partial implementation of Elementary String Conversions in `<charconv>`: integers only. The floating-point part is the last STL feature that needs to be implemented. We're also almost done with LWG issues: 33 added in 15.7, 14 done in future branches, 15 remain.

CLion 2018.1 released

- Announcement
- What's New Video
  - Better C++17 support
  - Support for Clang-tidy configuration files
  - Produce Linux binaries on Windows
  - Built-in Valgrind memcheck in WSL
  - Open single file/folder without CMake
  - Breadcrumbs in the editor
  - Partial Git commits
  - Support for Rust, Fortran, Objective-C/C++

- Home page

C++ auto-generated comparison operators

- Bjarne Stroustrup's post
- N3950: Defaulted comparison operators, by Oleg Smolsky
- N4175: Default comparisons, by Bjarne Strourtrup
- N4239: Defaulted Comparison Using Reflection
- N4401: Defaulted comparison operator semantics should be uniform
- P0221: Proposed wording for default comparisons, revision 2
- P0515: Consistent comparison (the spaceship operator)
- Reddit: Immutable objects in C++

True parallelism, with no concept of threads - Alfred Bratterud - Meeting C++ 2017

- Video
- IncludeOS
- GitHub

Fibers, green threads, channels, lightweight processes, coroutines, pthreads - there are lots of options for parallelism abstractions. But what do you do if you just want your application to run a specific task on a specific core on your machine? In IncludeOS we have proper multicore support allowing you to do just that in C++: assign a task - for instance a lambda - directly to an available CPU. <...> In this talk we'll <...> explore how direct per-core processing can be combined with threading concepts like C++14 fibers or coroutines, without taking away from the simplicity of getting work done uninterrupted.

True parallelism, with no concept of threads - Alfred Bratterud - Meeting C++ 2017 (cont.)

Conclusions:

- You don’t need classical threads to utilize CPU cores
  - Fibers and coroutines can run directly on them
- A pthreads backend requires true blocking
  - Might require fibers, yielding directly to scheduler
- Coroutines TS beats the simplest stack switch
- Stackful coroutines would replace our fibers
- Expect more multicore magic from IncludeOS

- Video
- How undefined signed overflow enables optimizations in GCC
- Guidelines for integers:
  - Signed: when you need normal arithmetic
  - Unsigned: flags, IDs, enumerations (`enum class`), bits
  - Avoid viral promotion of unsigned types wider than `int`
  - Avoid implicit sign conversions: `-Wsign-conversion -Werror`

```
1 int32_t a = -1; uint32_t b = 1;
2 if (a > b) std::cout << "wat"; // `a` --> 0xffffffff
```

C++20: Signed Integers are Two's Complement

Video: Deep Learning with C++ - Peter Goldsborough - Meeting C++ 2017

- YouTube
- Google's TensorFlow C++ API
- How to train a Deep Neural Network using only TensorFlow C++ – GitHub

C++ Memory Model

- Part 1
  - Program order versus memory order
  - Atomic operations
  - Barriers
- Part 2
  - A hardware model to explain acquire/release
  - Sequential Consistency

HPX v1.1 released

- Download
- Changelog
- Requires GCC 5.9+, VS2015.5+, Boost 1.55+, CMake 3.3.2+

HPX is a general purpose parallel C++ runtime system for applications of any scale. It implements all of the related facilities as defined by the C++ Standard. <...> HPX provides the only widely available open-source implementation of the new C++17 parallel algorithms. Additionally, HPX implements <...> large parts of the C++ Concurrency TS, task blocks, data-parallel algorithms, executors, index-based parallel for loops, and many more.

As a C++ developer I think that Rust is...

Reddit

- is a good language but the community is toxic towards people not using Rust
- lacks function overloading, value generics, variadic generics and exceptions
- is much nicer, though I doubt I'll get to use it in my current job any time soon
- solves a non-problem if you use modern C++
- is a topic way too often spawning on this C++ subreddit
- would have been a compelling alternative if it came out 10 years ago
- lacks library support
- less powerful but more user-friendly than C++

STL on Twitter

Stephan T. Lavavej @StephanTLavavej

There's something delightful about implementing C++17 Filesystem's character encoding conversions in MSVC, using the Unicode CAT emoji (U+1F408) as test data, debugging into the code, and being surprised by seeing a little kitty in the debugger. 🐈

12 97

Simon Brand on Twitter

Simon Brand
@TartanLlama

Godwin's Law: C++ Edition -- As an online C++ discussion grows longer, the probability of someone mentioning Rust approaches 1.

28/03/2018, 10:36

27 Retweets **91** Likes