C++ Club Meeting Notes

Gleb Dolgich

2017-10-12

More CppCon 2017 trip reports

- Isabella Muerte
  - “I came to the conclusion that the Modules TS needs more scrutiny and is in dire need of fixing, or we’re all going to regret it.”
- Jens Weller of Meeting C++
  - Highlights the differences between CppCon and Meeting C++
  - “Matt Gottbold” :-)
- Bartek Filipek

CppCon 2017: Steve Carroll & Daniel Moth “Latest & Greatest in Visual Studio for C++ developers”

YouTube

Facebook and Modules TS in GCC

- Post
- GCC's C++ module system will not be ready for next year's GCC 8 release, meaning GCC 9 in 2019 at the earliest.
- PDF
- Wiki

Millennials Are Killing The Modules TS, by Isabella Muerte

Post

- There are many incompatible notions on how modules should work (*and this post is one of them, it seems*)
- Without support of build tools modules are dead
- Compiler should not become build system
- We will have interface modules, and implementation modules (?)
- The entire goal is to give users a guaranteed exported interface. There is no module hierarchy (?), no guaranteed single file module implementation, no de facto way of finding a module.
- Members on the committee cannot even agree as to what a module is.
- What does `extern module` do???

Reddit on “Millennials Are Killing The Modules TS”

Reddit

- A terrible title
- A lot of hand-waiving as well as statements ranging from unsubstantiated to factually incorrect
- Compiler can be a build system to the extent that it already is with header files
- Andrew Sutton: “I am not suggesting that there are not real development problems. I am simply saying that this knee-jerk criticism of and raging against the Modules TS is totally without merit on the basis that the Modules TS is limited in the kinds of problems it can solve.”

Modules intro from the author of build2

- Reddit
- Intro
- Design guidelines
- Modularising existing code

YouTube

Co-arrays are a Fortran standard extension implementing Partitioned Global Address Space (PGAS) model.

- SIMD
- PGAS
- Non-Uniform Memory Access (NUMA)
- Async programming

CppCon 2017: Stephen Dewhurst “Modern C++ Interfaces: Complexity, Emergent Simplicity, SFINAE, and Second Order Properties of Types”

YouTube :: Website

- “Most good bugs are team efforts”
- `constexpr if` in C++17 + return type deduction in C++14 allows greater flexibility
- Years ago policy-based design moved implementation details to interface users
- Recently, the shift is back to interface implementers
- SFINAE-based interface design: increased C++ complexity means we’re embedding our judgment into our interfaces
- C++ is so complex now that it’s actually becoming simpler due to convention, idiom, embedded experience, and DWIM interfaces

“Making new friends” idiom by Dan Saks

Wikibooks

The goal is to simplify creation of friend functions for a class template.

```
1  template<typename T>
2  class Foo {
3     T value;
4  public:
5     Foo(const T& t) { value = t; }
6     friend ostream& operator <<(ostream& os, const Foo<T>& b)
7     {
8        return os << b.value;
9     }
10 };
```

Godbolting your C++ code

JetBrains interview: Matt Godbolt

YouTube

- C++ is copy-based, Java is reference-based
- Const correctness
- Value types
- Builder pattern for value types
- Persistent data structures

In-place containers for fun and profit

Blog post by David Gross

The idea behind `inplace_string` is to get a full replacement of C++17's `std::string`, with an in-place memory storage.

The underlying container is `std::array<CharT, N>` and it uses the famous trick — made popular by fbstring — of storing the remaining size within the last byte of the string.

```
1 using Name = inplace_string<15>;
2 // 16 bytes on stack, size included
3 Name name = "foo";
4 auto it = name.find("r");
5 assert(it == Name::npos);
6 name += "bar";
7 std::string str(name); // implicit string_view construction
```

Meta Stuff

- GitHub (MIT)
- Article

  This is the code I use in my game for all serialization/deserialization/introspection stuff.

- Strongly typed and doesn't use RTTI or virtual functions in any way.
- No dependencies. You have to use modern C++ compiler which supports C++14, though. (VS 2015, GCC 5+, Clang 3.8)
- Serialization is not limited to a particular format.

Article

Goal:

The fastest possible serialization for in-memory structure, assuming that it will be deserialized by **exactly the same executable**.

Use for the Poop emoji

Reddit

This detects whether the Visual Studio project has the proper UTF-8 flags set to compile it correctly.

```
static_assert(
    (static_cast<unsigned char>("💩"[0]) == 0xF0) &&
    (static_cast<unsigned char>("💩"[1]) == 0x9F) &&
    (static_cast<unsigned char>("💩"[2]) == 0x92) &&
    (static_cast<unsigned char>("💩"[3]) == 0xA9),
        "Source or compiler not UTF-8 compliant!"
        " Add flag /utf-8 for Visual Studio");
```

Quote

Dennis Kubes:

> *C is memory with syntactic sugar.*

Quote

Steve Haflich:

> *When your hammer is C++, everything begins to look like a thumb.*