C++ Club Meeting Notes

Gleb Dolgich

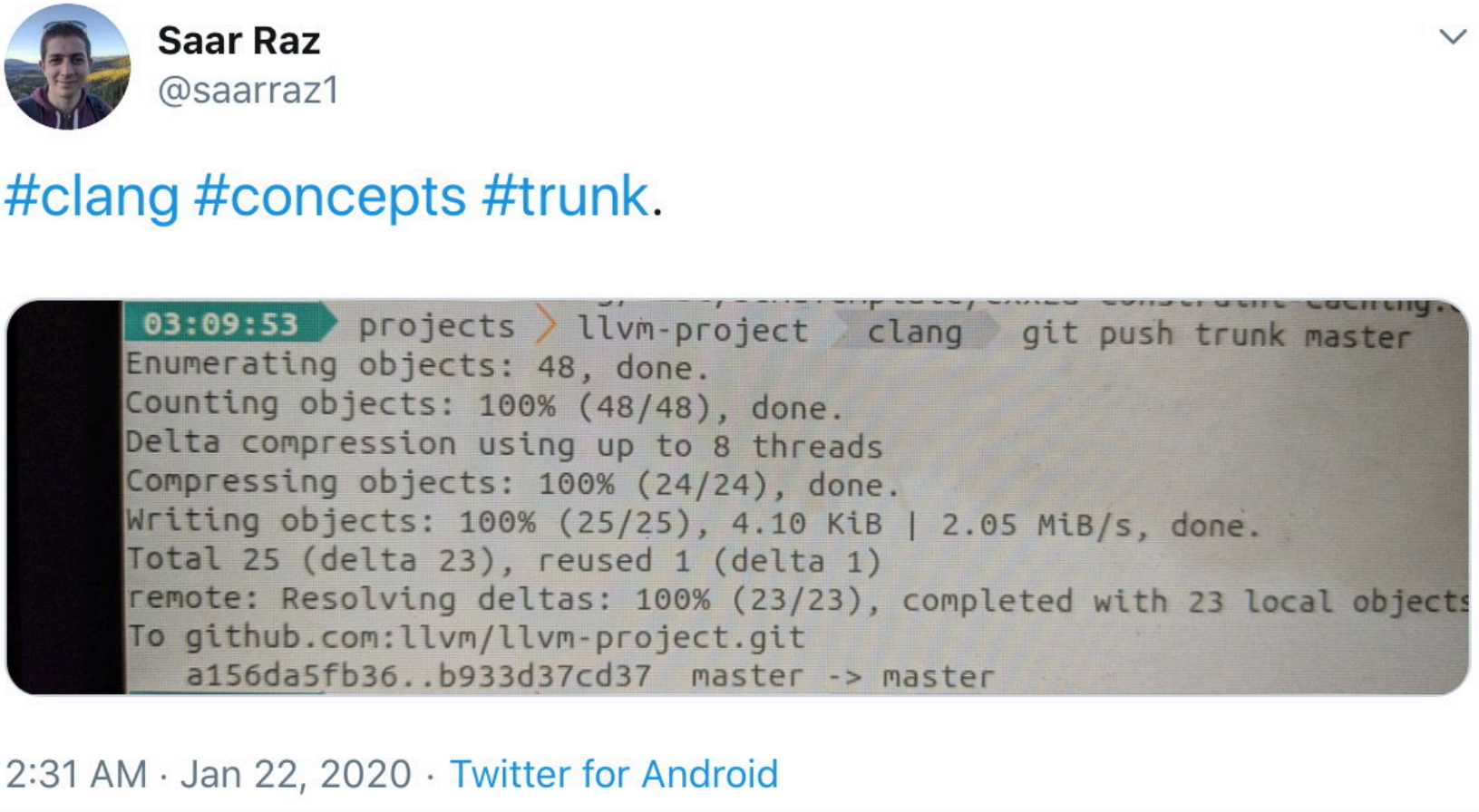
2020-02-06

Follow-up: Aggregates

From CppReference: An aggregate is one of the following types:

- array type
- class type (typically, `struct` or `union`), that has
  - no private or protected *direct* (since C++17) non-static data members
  - no *user-declared constructors* (until C++11)
  - no *user-provided constructors* (explicitly defaulted or deleted constructors are allowed) (since C++11) (until C++17)
  - no *user-provided*, *inherited*, *or explicit constructors* (explicitly defaulted or deleted constructors are allowed) (since C++17) (until C++20)
  - no *user-declared or inherited constructors* (since C++20)
  - no virtual, private, or *protected* (since C++17) base classes
  - no virtual member functions
  - no *default member initializers* (since C++11) (until C++14)

Concepts pushed to Clang master



Saar Raz
@saarraz1

#clang #concepts #trunk.

```
03:09:53 projects > llvm-project > clang > git push trunk master
Enumerating objects: 48, done.
Counting objects: 100% (48/48), done.
Delta compression using up to 8 threads
Compressing objects: 100% (24/24), done.
Writing objects: 100% (25/25), 4.10 KiB | 2.05 MiB/s, done.
Total 25 (delta 23), reused 1 (delta 1)
remote: Resolving deltas: 100% (23/23), completed with 23 local objects
To github.com:llvm/llvm-project.git
   a156da5fb36..b933d37cd37  master -> master
```

2:31 AM · Jan 22, 2020 · Twitter for Android

- Reddit

[C++ coroutines] Initial implementation pushed to GCC master

- Message
- Reddit

> *This is not enabled by default (even for -std=c++2a), it needs* ***-fcoroutines***

Structured Exceptions (Win32) and C++

- Raymond Chen: How can I handle both structured exceptions and C++ exceptions potentially coming from the same source?
  - Reddit
- Raymond Chen: Can I throw a C++ exception from a structured exception?

“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 };
```

A hidden gem: `inner_product`

- Article

A hidden gem: `inner_product`

4

Conor Hoekstra @code_report
@cjdb_ns & @TartanLlama

This makes me so incredibly happy! I literally just yesterday googled, C++17 / C++20 zip to see if they had anything, because I wrote some code in both C++ and #Python and Python was so much more beautiful.

```
int solve(int h, vector<int> w, vector<int> l) {
   int p = 0;
   for (int i = 0; i < w.size(); ++i)
      p = max(p, w[i] - l[i] / 4);
   return max(0, p - h);
}
```

```
def solve(h, w, l):
    p = max(a - b//4 for a, b in zip(w, l))
    return max(0, p - h)
```

29w • 03/12/2018 • 17:47

Conor Hoekstra @code_report
@cjdb_ns & @TartanLlama

Also, I just discovered std::inner_product - a beautiful temporary solution to a lack of zip. #cpp #inner_product

```
int solve(int h, vector<int> w, vector<int> l) {
   return max(0, inner_product(begin(w), end(w), begin(l), 0,
      [](auto a, auto b) { return max(a, b); },
      [](auto a, auto b) { return a - b / 4; }) - h);
}
```

27w • 16/12/2018 • 09:30

A new decade, a new tool: libman

- Colby Pike (vector-of-bool)
- Reddit
- GitHub
- Specification

libman is a new level of indirection between package management and build systems.

dds is Drop-Dead Simple build and package manager.

- CppCon 2019: Robert Schumacher “How to Herd 1,000 Libraries”

- Article

Twitter: Pure virtual function syntax (1/2)

Shafik Yaghmour @shafikyaghmour

Rereading “The Design and Evolution of C++”

= 0

syntax was used for pure virtual function in order to avoid having to add a new keyword such as pure or abstract because the feature was added close to the next release.

Twitter: Pure virtual function syntax (2/2)

13.2.3 Syntax

The curious `=0` syntax was chosen over the obvious alternative of introducing a keyword `pure` or `abstract` because at the time I saw no chance of getting a new keyword accepted. Had I suggested `pure`, Release 2.0 would have shipped without abstract classes. Given a choice between a nicer syntax and abstract classes, I chose abstract classes. Rather than risking delay and incurring the certain fights over `pure`, I used the traditional C and C++ convention of using `0` to represent "not there." The `=0` syntax fits with my view that a function body is the initializer for a function and also with the (simplistic, but usually adequate) view of the set of virtual functions being implemented as a vector of function pointers (§3.5.1). In fact, `=0` is not best implemented by putting a `0` in the `vtbl`. My implementation places a pointer to a function called `__pure_virtual_called` in the `vtbl`; this function can then be defined to give a reasonable run-time error.

I chose a mechanism for specifying individual functions pure rather than a way of declaring a complete class abstract because the pure virtual function notion is more flexible. I value the ability to define a class in stages; that is, I find it useful to define some virtual functions and leave the definition of the rest to further derived classes.