C++ Club Meeting Notes

Gleb Dolgich

2018-03-15

- Blog post by Hans Klabbers
- Accio Dependency Manager, by Corentin

Arno Schödl – A Practical Approach to Error Handling

- ▶ YouTube
- ▶ think-cell
- ▶ The main product plugs into PowerPoint to provide chart auto-layout
- ▶ Patches PowerPoint code at run time (client-server architecture)
- ▶ They disassemble PowerPoint and related libraries using IDA to find entry points (working against undocumented APIs)!

Arno Schödl – A Practical Approach to Error Handling (cont.)

▶ Strong exception guarantee is very hard to achieve in application code: too many state changes
▶ What to do on error:
  ▶ Collect information, send report home
  ▶ Stop further reporting, as program state is undefined
  ▶ Despite the above, carry on and never terminate!
  ▶ Server stops processing certain type of requests if too many threads hang
  ▶ Reproduce error in the lab, handle reproducible errors only
▶ Asserts can be wrong! Don't terminate on assert, call home instead, otherwise developers will be afraid to add asserts!
▶ Reporting server gets 1 crash report per second on average

An Introduction to Reflection in C++

- ▶ Terminology:
    - ▶ "*Introspection* is the ability to inspect a type and retrieve its various qualities. You might want to introspect an object's data members, member functions, inheritance hierarchy, etc. And you might want to introspect different things at compile time and runtime."
    - ▶ "*Metaobjects* are the result of introspection on a type: a handle containing the metadata you requested from introspection. If the reflection implementation is good, this metaobject handle should be lightweight or zero-cost at runtime."
    - ▶ "*Reification* is a fancy word for "making something a first-class citizen", or "making something concrete". We will use it to mean mapping from the reflected representation of objects (metaobjects) to concrete types or identifiers."

▶ "Run-time type information/identification is a controversial C++ feature, commonly reviled by performance maniacs and zero-overhead zealots. If you've ever used dynamic_cast, typeid, or type_info, you were using RTTI."

▶ Macros: "Hana and MPL can adapt a user-defined POD-type into an introspectible structure which tuple-like access. The code is quite formidable and verbose. That's because there's a separate macro case for adapting a struct of specific sizes. For example, all structs with 1 member map to a particular macro, all structs with 2 members map to the next macro, etc." Max 62 members per struct.

▶ "The current pinnacle of POD introspection in C++14 is a library called **magic_get** (Precise and Flat Reflection, pre-Boost `boost::pfr`) by Antony Polukhin – https://github.com/apolukhin/magic_get – C++17 implementation doesn't use macros, just templates and structured bindings." Still, max 101 members per struct.
▶ Other languages: Python, Java, C#, Go

An Introduction to Reflection in C++ (cont.)

- ▶ C++20: `reflexpr`: "The `reflexpr` proposal, by Matúš Chochlík, Axel Naumann, and David Sankel, introduces several "metaobjects" which are accessed by passing a type to the new `reflexpr` operator."
- ▶ C++20: `operator$`: "Andrew Sutton and Herb Sutter wrote *A design for static reflection*, which introduces the reflection operator, $, as a way of getting object metadata out of a class, namespace, etc. (Using $ has been argued out of favor because it is common in legacy code, particularly in code generation and template systems which are not necessarily valid C++ but produce C++ sources.)"
- ▶ "The fundamental design difference between these two papers is whether the result of the reflection operator is a value or a type."
- ▶ "Concepts are design prerequisite for both of the reflection papers."

Smf RPC

smf is a new RPC system and code generation like gRPC, Cap'n'Proto, Apache Thrift, etc, but designed for microsecond tail latency.

- ▶ Home page
- ▶ Code
- ▶ Reddit

Based on Seastar (code) by ScyllaDB.

Frozen: a header-only, `constexpr` alternative to gperf for C++14 users

Header-only library that provides zero-cost initialization for immutable containers and various algorithms.

- ▶ Code (Apache 2.0)
- ▶ Intro

```cpp
#include <frozen/unordered_set.h>

constexpr frozen::unordered_set<int, 3> keys = {1,2,4};

int some_user(int key) {
    return keys.count(key);
}
```

Video

How we know if a code base is good or bad?

Getting rid of inheritance and virtual functions in favour of templates and value
semantics.

Video

Data should be as encapsulated as possible => free functions lead to better encapsulation than member functions.

How to test a private function?

▶ `friend` test function (bad)
▶ `#define private public` (worse)
▶ free function operating on data in the class (better)

Fun with (user-defined) attributes – Jonathan Müller – Meeting C++ 2017

- ▶ Video

Syntax:

- ▶ Simple identifiers: [[foo]]
- ▶ Pack expansion: [[foo...]]
- ▶ Multiple attributes: [[foo, bar, baz]]
- ▶ Arguments: [[foo(a, "b", 3.14, +)]]
- ▶ Namespaces: [[foo::bar]]
- ▶ Using declaration: [[using foo: bar, baz]]

```
1  [[noreturn]] void handle_error(std::string msg)
2  {
3      throw std::runtime_error(std::move(msg));
4  }
```

alignas changes generated code!

```
1 alignas(128) char cacheline[128];
```

```cpp
void func(int param);

[[deprecated("pass a parameter instead")]]
void func()
{
    func(0);
}
```

```
1  switch (selected_option)
2  {
3      <...>
4      case option::b:
5          if (foo)
6              log_error("option::b may not be used when foo is set");
7          // otherwise b is exactly like c so continue on
8          [[fallthrough]];
9      case option::c:
10         <...>
11 }
```

```
1 [[nodiscard]]
2 int error();
```

```cpp
[[maybe_unused]]
static bool is_valid(int param);

void do_something(int a, [[maybe_unused]] int b)
{
    <...>
}
```

Compiler-specific attributes change generated code, C++ standard ones don't
(except `alignas`).

- ▶ YouTube
- ▶ P0633: Exploring the design space of metaprogramming and reflection

```
constexpr { // metaprogram
    for... (auto x : $x.member_variables()) {
        // do stuff with x
    }
}
```

Injection of new declarations or statements

```
1  template<enum E>
2  const char* to_string(E value)
3  {
4      switch (value) constexpr {
5          for... (auto e : $E.enumerators())
6              -> { case e.value(): return e.name(); } // Injection
7      }
8  }
```

Source code literals

```
 1  auto frag = __fragment class C { // name C is optional
 2      int x;
 3      int foo() {
 4          return x + this->y;
 5      }
 6  };
 7  auto frag = __fragment class Node {
 8      Node* next;
 9      Node* prev;
10  };
11  auto ns = __fragment {
12      case 0: return 42;
13  };
```

UB will delete your null checks

- Post

VIM Clutch

▶ GitHub

VIM Clutch is a hardware pedal for improved text editing speed for users of the magnificent VIM text editor. When the pedal is pressed down, the pedal types "i" causing VIM to go into Insert Mode. When released, it types and you are back in Normal Mode.

Twitter

Nicole is a Monoid in... ↑ 11 Replies, 3 Quotes
@ubsanitizer

I just realized I can use unicode in standard C++ programs. This is wonderful :D

```
using ℝ = double;

struct ℝ² {
  ℝ x₀;
  ℝ x₁;
};
```

This is a valid C++ program that's compiled by all major compilers. I'm so happy :')

170 Likes	66 Retweets	Thread ❯