autoscale: true

13 July 2017

Figure 1: Inline

Toronto tweets

Billy O'Neal @MalwareMinigun 15h

Frustrated by concepts advocates calling the abbreviated function template syntax "natural".

Toronto, Ontario

Billy O'Neal @MalwareMinigun 15h

There is nothing about programming languages that is natural. Using that word has demeaning connotations for anyone who isn't a fan of that.

Billy O'Neal @MalwareMinigun

OTOH "terse" has negative connotations for that syntax. Maybe we should say "short

Toronto tweets

Eric Niebler (@ericniebler)
12/07/2017, 16:48

@AlisdairMered @MalwareMinigun @dgregor79 @stefanusdutoit I expect we'll get the abbreviated function syntax back in time for C++20.

Alisdair Meredith (@AlisdairMered)
12/07/2017, 16:44

@MalwareMinigun @ericniebler @dgregor79 @stefanusdutoit I am uncomfortable with short syntax, but fear it may still be the right answer. We have two years to work out the details; sooner is better

---

Billy O'Neal (@MalwareMinigun)
12/07/2017, 16:37

@ericniebler @dgregor79 @AlisdairMered @stefanusdutoit I'm a bit sad that we said we want to encourage further work with the short syntax though. :/

Eric Niebler (@ericniebler)
12/07/2017, 16:23

@dgregor79 @AlisdairMered @stefanusdutoit Requires clauses are still in, template<Iterator I> syntax is still in, deduction constraints currently in limbo, everything else deferred.

Doug Gregor (@dgregor79)
12/07/2017, 15:17

@AlisdairMered @stefanusdutoit Stripped down to what?

Alisdair Meredith (@AlisdairMered)
12/07/2017, 12:23

Exception handling in constructors

- Function `try` block
- https://medium.com/cpp-station/exception-handling-in-constructors-26bf4c811b46

Exception handling in constructors

```
A::A(int& n) try : B(n), cObject(0), x(-1), y(n) {
}
catch(Exception& e) {
    //Option 1: Throw same exception
    //Option 2: Log and thrown another exception
    //Option 3: Exception thrown automatically on reaching end of handler
}
catch(AnotherException& e) {
    //Option 1: Throw same exception
    //Option 2: Log and thrown another exception
    //Option 3: Exception thrown automatically on reaching end of handler
}
```

Exception handling in constructors

- Use with constructors that have initializer list that calls a user-defined constructor that can throw.
- You can replace the exception being thrown and cause some useful side effects such as log the failure.

The rise of the new language MC++

http://cppdepend.com/blog/?p=171

- Folly by Facebook as a modern C++ library example
- `auto` everywhere
- `nullptr`
- `std::shared_ptr`

The rise of the new language MC++

- Scoped enums
- `static_assert`
- variadic templates
- range `for` loops
- `std::initializer_list`
- `noexcept`

The rise of the new language MC++

- `std::thread`
- unordered containers
- `default` and `delete` special member functions
- `override`
- lambdas
- `std::move` (doesn't move) and `std::forward` (doesn't forward)

To move or not to move

http://www.stroustrup.com/move.pdf

Discusses issues with generating implicit copy and move operations.

> *To have move useful, it must be implicitly generated in many cases. To minimize surprises and bugs, no move operations should be generated for classes with a user-specified copy, move, or destructor. To keep the rules consistent, the generation of copy operations should be deprecated for classes with a user-specified copy, move, or destructor.*

Ninja build system

- Built for speed: instant incremental builds
- Linux, Windows; optimised for big C++ projects
- Works with gyp, CMake, etc. (possibly premake)
- https://ninja-build.org
- https://github.com/ninja-build/ninja
- Used by Chrome, Android, LLVM

Thinking Outside the Synchronisation Quadrant

Kevlin Henney, ACCU 2017

https://www.youtube.com/watch?v=UJrmee7o68A

- synchronization only needed for shared mutable data
- instead of threads why not use message passing between processes
- “processes” can be lightweight (Erlang)
- agents react to messages
- no synchronization needed within message handlers

‘Meaningful’ casts

Vittorio Romeo – Meeting C++ 2015

YouTube (12m)

Range-checked casts

```
1 template <typename TOut, typename TIn>
2 constexpr auto to_num(const TIn& x) noexcept {...}
```

```
1 int a{10};
2 to_num<float>(a);
3 to_num<int>(-1); // OK
4 to_num<unsigned int>(-1); // Run-time assertion
5 to_num<float>(NAN); // Run-time assertion
```

Also: `from_enum`, `to_enum`, support for `std::aligned_storage_t`, casts along inheritance hierarchy, etc.

By Thomas Becker

Table of Contents

Scott Meyers (Going Native, 2013)

Video (1h 15m)

- Understand `std::move` and `std::forward`
- Declare functions `noexcept` whenever possible
- Make `std::threads` unjoinable on all paths

Scott Meyers (Going Native, 2013)

```
1  class ThreadRAII {
2  public:
3      typedef void (std::thread::*RAIIAction)();
4      ThreadRAII(std::thread&& thread, RAIIAction a)
5      : t(std::move(thread)), action(a) {}
6      ~ThreadRAII()
7      {if (t.joinable()) (t.*action)();} // race condition?
8      std::thread& get() {return t;}
9  private:
10     RAIIAction action;
11     std::thread t;
12 };
13
14 ThreadRAII t1(std::thread(doThisWork), &std::thread::join);
15 ThreadRAII t1(std::thread(doThisWork), &std::thread::detach);
```

Outcome v2 is feature complete

- Code
- Documentation
- Reddit thread

```
outcome::result<int/*, std::error_code*/>
convert(const std::string& str) noexcept;
```

Boost review feedback addressed.

Undefined Behavior in 2017

by John Regehr (Professor of Computer Science, University of Utah, USA) and Pascal Cuoq

Article

- Goal 1: Every UB (yes, all ~200 of them, we'll give the list towards the end of this post) must either be documented as having some defined behavior, be diagnosed with a fatal compiler error, or else – as a last resort – have a sanitizer that detects that UB at runtime.
- Goal 2: Every UB must either be documented as having some defined behavior, be diagnosed with a fatal compiler error, or else have an optional mitigation mechanism that meets the requirements above.

Presenting Code

- Use dark-on-light colour scheme
- Avoid dark backgrounds as syntax-highlighted code becomes invisible or very hard to read
- Consider effects of video compression
- Think about colour-blind developers