C++ Club Meeting Notes

Gleb Dolgich

2018-02-22

Pre-Jacksonville C++ Committee meeting papers

List

p0713r1: Identifying Module Source

p0713r1

P0906R0: Improvement suggestions for the Modules TS, by Jens Maurer

P0906R0

- Different access paths to a type (term: “attendant entity”)
- Definitions of inline functions (also: `constexpr`)
- Allow vacuous exports
- Exclude private members from attendant entities

P0909R0: Module TS Supports Legacy Integration, by Steve Downey (Bloomberg)

P0909R0

```
// facility module interface unit
#include <facility.h> // outside the purview of module facility

export module facility;

export namespace facility {}
export using facility::Foo;  // exports complete type Foo
export using facility::func; // exports both func from facility
export using facility::BAR;  // exports BAR, and associated enumerators
```

P0923r0

Proposed changes:

- Internal-Linkage Entities Cannot Be Found From Outside
- ADL is Consistent

P0924r0: Modules: Context-Sensitive Keyword, by Nathan Sidwell (Facebook)

P0924r0

P0925r0: Modules: Unqualified Using Declarations, by Nathan Sidwell (Facebook)

P0925r0

The proposal is that an unqualified name could be employed in a *using-declaration*, in at least some circumstances:

- *Using-declarations* may take an unqualified name
- Such an unqualified *using-declaration* must find a binding in the current namespace
- Only unqualified *using-declarations* may be exported

Modules and macros

- p0947R0: Another take on Modules, by Richard Smith (Google)
  - Macro Export
- p0955r0: Modules and Macros, by Bjarne Stroustrup

P0501R3: C++ Module TS Issues List, by Gabriel Dos Reis

P0501R3

P0797R1: Handling Concurrent Exceptions with Executors, by Matti Rintala, Michael Wong, Carter Edwards, Patrice Roy, Gordon Brown, Mark Hoemmen

P0797R1

Problem: In concurrent execution it is possible for several parallel executions to throw exceptions asynchronously. If more than one of these exceptions end up in the same thread of execution, the situation is problematic, since C++ allows only one exception to propagate at any time.

`exception_list` in Parallelism TS 2

Alternatives: OpenMP, SYCL (based on OpenCL), HPX, HSA.

This paper proposes a set of new exception handling properties for the executors property mechanism: `no_exceptions`, `single_exception`, `single_exception_reduction`, and `multiple_exceptions`.

Video: CppCon 2017: Andrew Sutton “Reflection”

- P0590: Type-encoded reflection
- Clang fork

YouTube

YouTube

- A C++17 constexpr function:
  - is not virtual
  - takes literal types and returns a literal type
  - is =delete, =default, or doesn't contain:
    - an asm definition
    - a goto statement
    - an identifier label
    - a try block
    - a definition of a variable of non-literal type (or static or thread_local) for which no initialization is performed
- lambda operator() is constexpr by default if possible

Practical `constexpr` - Jason Turner - Meeting C++ 2017 (cont.)

- Problematic:
  - broken standard library implementations
  - poor standard library support for algorithms
  - a cascade of 'viral' `constexpr` usage
- Worth it for:
  - code simplification
  - reduction of dynamic allocations
  - startup performance
  - runtime performance

Constructors are not functions: you cannot take an address of a constructor.

Practical constexpr - Jason Turner - Meeting C++ 2017 (cont.)

```
1 #include <array>
2 #include <bitset>
3 #include <cstddef>
4
5 constexpr std::size_t N = 100000;
6 std::array<std::bitset<N>, N> elems;
7
8 int main(){}
```

GCC uses 9GB RAM to build this.

Practical constexpr - Jason Turner - Meeting C++ 2017 (cont.)

In C++17:

- std::array is constexpr-enabled, except for std::array::fill
- std::string_view is fully constexpr-enabled (except buggy construction from const char* in constexpr context)
- constexpr algorithms: min, max, minmax, and _element (others can be enabled too)
- std::optional and std::variant are not constexpr
- std::pair and std::tuple are constexpr-enabled, except for swap and operator=
- std::begin, std::end, std::cbegin, std::cend, std::next, std::prev, std::distance, std::advance are all constexpr
- std::swap and std::exchange are not constexpr

Easy steps to modernize your C++ code

CppDepend Blog

Inspector: A drop-anywhere C++ REPL

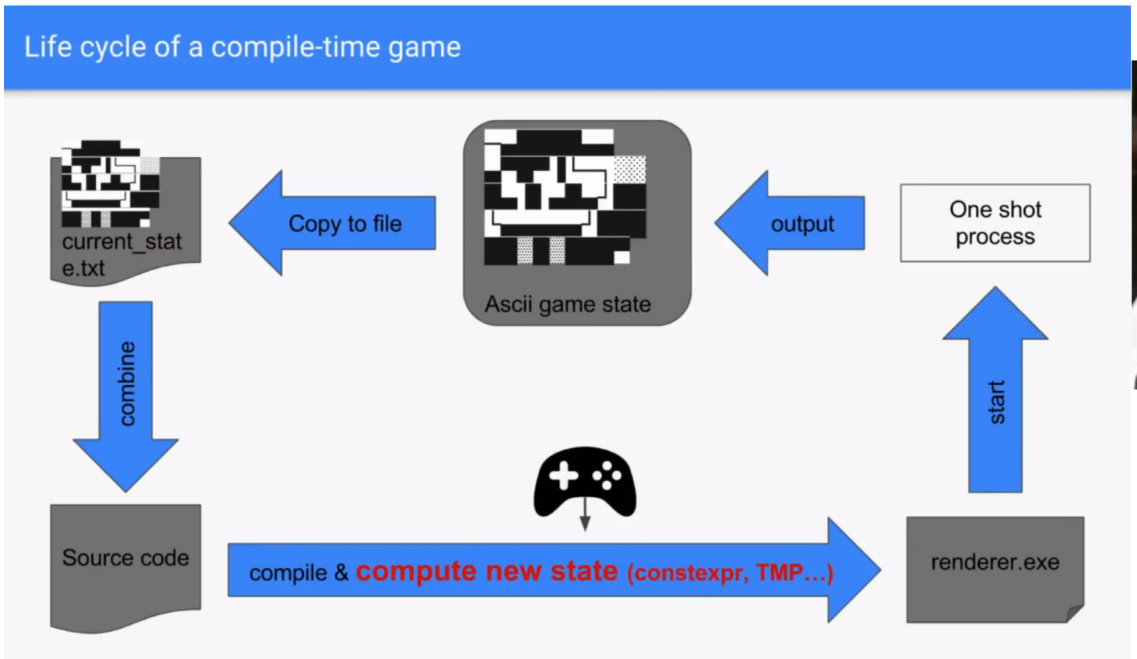
- GitHub
- Video
- Slides

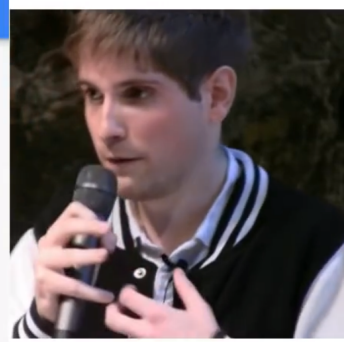
```cpp
#include <iostream>
#include <string>

int main(int argc, char** argv) {
    int a = 1;
    std::string b = "hello world";
#include INSPECTOR
    std::cout << "second break." << std::endl;
#include INSPECTOR
}
```

- Video
- Reddit
- GitHub

Jean Guegant - Meta Crush Saga: a C++17 compile-time game (cont.)





Jean Guegant - Meta Crush Saga: a C++17 compile-time game (cont.)

Why?

- Blazing fast runtime execution
- Safer than Rust
- Fewer side effects than Haskell
- More cryptic than Perl
- More awkward than any JavaScript framework
- Obtain the title of “Lead Senior C++ Over-Engineer” at work
- Use your CPU to warm yourself during winter

Compile-time shenanigans

- Snake
- Ray tracer

Compile-time Tetris

- Post
- GitHub
- Reddit
- HackerNews
- Slashdot

Market share of the most used C/C++ IDEs in 2018, statistics and estimates

Post

IDE	Share
Visual Studio	28.43%
Vim	16.54%
Qt Creator	11.64%
Visual Studio Code	10.31%
CLion	8.91%

Quote

Kelsey Hightower (@kelseyhightower):

> *You haven't mastered a tool until you understand when it should not be used.*