Runtime to compile-time dispatch in C++

Jackie Kay
ϟ ACCU 2017 ϟ

About me

- Software engineer specialized in robotics
- Modern C++ and metaprogramming fangirl
- jackieokay.com
- @jackayline

Why do this?

- Dynamically selecting types
- Variant visitation
- State machines
- Deserialization

Mapping from compile-time to runtime

Easy as 🍰 :

```
template<size_t CompileTime>
struct a {
  int runtime = CompileTime;
};

constexpr int compile_time = 42;
int runtime = compile_time;
```

Mapping from runtime to compile-time

```
int runtime = argv[1];

constexpr int compile_time = runtime;  // ERROR
```

😟

A C++17 Approach

```
struct linear_match {
  template<size_t M>
  void apply() {
    if (index == M) {
      std::cout << M << "\n";
      // could use M to index into a tuple, etc.
    }
  }

  template<size_t ...I>
  void apply_sequence(std::index_sequence<I...>&&) {
    (apply<I>(), ...);
  }

  const unsigned index;
};
```

Usage:

```
const unsigned i = atoi(argv[1]);
// BENCHMARK_SIZE defined by preprocessor
constexpr auto N = BENCHMARK_SIZE;

linear_match{i}.apply_sequence(
  std::make_index_sequence<N>{});
```

Can we do better?

CS101: use binary search!

```
struct binary_search_match {
  template<int M, int L, int R>
  void apply() {
    if (index == M) {
      std::cout << M << std::endl;
    } else if (index > M) {
      constexpr auto L2 = M + 1;
      this->apply<(L2 + R) / 2, L2, R>();
    } else if (index < M) {
      constexpr auto R2 = M - 1;
      this->apply<(L + R2) / 2, L, R2>();
    }
  }

  const unsigned index;
};
```

Benchmarks

...

~~Benchmarks~~

- Saw no difference in runtime cost
- Started reaching sizes that crashed my compiler (15000)
- Binary search example had larger code size and slower compilation times
- I have a lot of questions!