std::future

a (možné) novinky v C++20

Josef Doležal

Co je to future

- Zjednodušený přístup k paralelním výpočtům
- Abstrakce nad standardními vlákny
- Spolu s std::promise slouží k synchronizaci vláken na úrovni jazyka

future v C++11

Vytvoření future

```
future<User> userFuture = async(get_user_profile);

future<int> computationFuture = async(()[]{
        return complexComputation();
});
```

Počkání na výsledek

```
userFuture.get();  // Čeká na výsledek a vrátí hodnotu
userFuture.wait(); // Pouze čeká na výsledek
```

future v C++11

Využití více future najednou

```
// Uložiště pro futures
std::vector<std::future<int>> futures;

// Vytvoření futures
for(int i = 0; i < 10; ++i) {
        futures.push_back (std::async(complexComputation, i));
}

// Počkání na výsledek
for(auto &e : futures) {
        std::cout << e.get() << std::endl;
}
```

Možné novinky v C++20

Vytváření future pomocí **std::make_ready_future** a **std::make_exceptional_future**

```
future<int> compute(int x) {
        if (x < 0) return make_exceptional_future<int>(-1);
        if (x == 0) return make_ready_future<int>(0);

        future<int> f1 = async([]() { return complex(x); });
        return f1;
}
```

Možné novinky v C++20

Řetězení futures

```
auto futureChain = async([]() { return complexComputation(); })
                                    .then([](future<int> f) { return relatedComputation(f.get()) })
                                   .then( ... );
futureChain.get();
```

Možné novinky v C++20

Nové **std::when_all** a **std::when_any**

```
vector<future<int>> myFutures = …;

auto future_any = when_any(myFutures.begin(), myFutures.end());

auto future_all = when_all(myFutures.begin(), myFutures.end());
```

Zdroje

- www.modernescpp.com/index.php/monads-in-c
- www.modernescpp.com/index.php/promise-and-future
- www.open-std.org/jtc1/sc22/wg21/docs/papers/2013/n3721.pdf

Děkuji za pozornost