

Přechod backendu systému FRED na moderní C++

Miroslav Franc • miroslav.franc@nic.cz • 24. listopadu 2017

Osnova

- Co je to FRED?
- Proč moderní C++?
- Co je nového v moderním C++?
- Co je potřeba k refaktorizaci?
- Stručný příklad
- Největší problémy
- Co nám to přinese?



Co je to FRED?

- Free Registry for ENUM and Domains
- Skupina open source nástrojů pro správu doménového registru
- Platfoma: C/C++/Python + PostgreSQL + Linux
- <http://fred.nic.cz>
- Backend převážně v C++



Proč moderní C++

- C++03 je update C++98 (první standard)
- vývoj C++ se zrychluje
- podpora v překladačích je dobrá
- kód je kratší a některé problémy jsou pryč
- C++14 vs. C++17



Moderní C++



Move sémantika

```
class Foo
{
    // ...
    Foo(const Foo & foo);
    Foo(Foo && foo);
    Foo & operator=(const Foo & foo);
    Foo & operator=(Foo && foo);
    // ...
};
```



Chytré ukazatele

```
std::unique_ptr<Base> move_only_ptr =  
    std::make_unique<Derived>(0);
```

```
const std::unique_ptr<Base> scoped_ptr =  
    std::make_unique<Derived>(1);
```

```
std::shared_ptr<Base> reference_counted_ptr =  
    std::make_shared<Derived>(2);
```



Automatická dedukce typů

```
auto str = std::to_string(42);
```

```
auto x = std::atol("42");
```

```
auto y = 40L + 2;
```

```
auto foo() { return 42L; }
```

```
template<typename T, typename U>
```

```
auto add(const T & a, const U & b) > decltype(a + b)
```

```
{
```

```
    return a + b;
```

```
}
```



constexpr

```
constexpr long long factorial(long long n)
{
    return n < 2 ? 1LL : n * factorial(n - 1);
}
```

```
int main()
{
    char buf[factorial(5)];
    // ...
}
```



range-based for loop

```
int something[] = { 1, 2, 3, 4, 5, 6 };  
for (int &n: something)  
    n *= n;
```



lambda funkce

```
const int something[] = { 1, 2, 3, 4, 5, 6 };  
int sum = 0;  
std::for_each(std::begin(something), std::end(something),  
[&sum](const auto & n){ sum += n; }  
);  
  
// std::accumulate :)
```



default a delete

```
struct Foo
```

```
{
```

```
    Foo() = default;
```

```
    Foo(const Foo & foo) = delete;
```

```
    Foo(Foo && foo);
```

```
    Foo & operator=(const Foo & foo) = delete;
```

```
    Foo & operator=(Foo && foo);
```

```
    // ...
```

```
};
```



nullptr literál

```
void foo(int *) { std::cout << "pointer"; }  
void foo(int) { std::cout << "integer"; }
```

```
int main()  
{  
    foo(nullptr);  
}
```



override a final

```
struct Base
```

```
{  
    virtual void foo(int *);  
    virtual void bar(int *) final;  
};
```

```
struct Derived final : Base
```

```
{  
    void foo(int *) override;  
};
```



Šablony proměnných a aliasů

```
template<typename T>  
using Matrix = std::vector<std::vector<T>>;
```

```
// Matrix<int> foo;
```

```
template<typename T>  
constexpr T pi = T(3.1415926535897932385L);
```

```
// cout << pi<int> << endl;
```



Silně typované výčtové typy

```
enum class Cities : std::uint8_t
{
    Prague = 1,
    Brno = 2,
    Ostrava = 4
};

int main()
{
    const int a = static_cast<int>(Cities::Brno);
    // ...
}
```



noexcept

```
struct Foo
```

```
{
```

```
    Foo(Foo &&) noexcept;
```

```
    Foo(const Foo &);
```

```
    // ...
```

```
};
```

```
if (noexcept(Foo(std::move(Foo()))))
```

```
    move_all_foos(); // no way to rollback
```

```
else
```

```
    copy_all_foos_and_then_destroy_the_originals();
```



Co je potřeba k refaktorizaci?

- automatizovat
- libclang a clang-tidy
- kompilační databáze (json)
- autotools a bear (build ear)
- oba překladače + všechny testy



Názorný příklad

```
$ bear -o compile_commands.json make
```

```
[  
  {  
    "command": "c++ -Wall -std=c++14 -g -O2 -I... -DNDEBUG...",  
    "directory": "/build",  
    "file": "/src/neco.cc"  
  },  
  ...  
]
```

```
$ run-clang-tidy.py -fix \  
-checks='-*,modernize-make-unique' -header-filter='.*'
```



Největší problémy

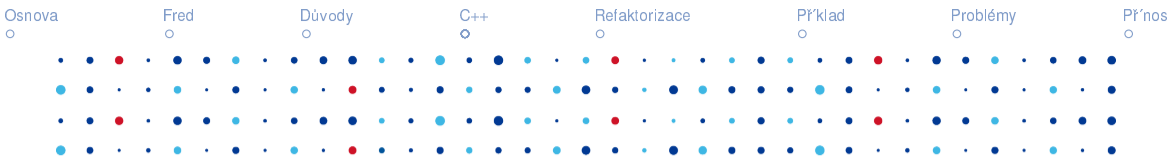
- chytré ukazatele
- specifikace výjimek
- několik případů nedefinovaného chování
- nekompatibilita mezi clangem a gcc
- clang-tidy není úplně stabilní
- revize coding style



Co nám to přinese?

- čistší a kratší kód
- efektivnější běh
- možnost používat nové věci
- podpora pouze novějších systémů
- problémy s balíčkováním





Děkuji za pozornost

Miroslav Franc • miroslav.franc@nic.cz

