C/C++: Lecture 8

Vorobev D.V

23.10.2020

std::uncaught_exception

It returns True $\Leftrightarrow$ Stack unwinding is currently in progress

```
// before C++11
bool uncaught_exception() throw();

// since C+11
bool uncaught_exception() noexcept;
```

std::uncaught_exception

```
struct Foo {
    ~Foo() {
        if(std::uncaught_exception()) {
            std::cout << "~Foo() : normall call";
        } else {
            std::cout << "~Foo() : stack unwinding";
        }
    }
};

int main() {
    Foo f;
    try {
        Foo f;
        throw std::runtime_error("test");
    } catch (const std::exception& e) {
        std::cout << e.what();
    }
}
```

Levels of exception guarantee

Samples were taken from: CppCon 2019: Ben Saks
"Back to Basics: Exception Handling and Exception Safety"

No guarantee

If the function throws an exception $\Rightarrow$ leaks are possible + invariants may be destroyed

No guarantee

```
class file {
    public:
        file(char const* name, char const* mode);
        ~file() noexcept;
        bool is_open() const noexcept;
        void put(int i);
        void put(char const* s);
    private:
        FILE* pf;
};
```

No guarantee

```
file::file(char const* name, char const* mode) :
    pf {fopen(name, mode)}
{
}

file::~file() noexcept {
    if (pf != nullptr) {
        fclose(pf);
    }
}
```

No guarantee

```
void f(char const* n) {
    FILE* outf = fopen(n, "w");
    if (outf != nullptr) {
        fprintf(outf, "The value is: \%d", g());
        fclose(outf);
    }
}
```

Basic guarantee

If the function throws an exception $\Rightarrow$ no leaks + invariants preserved

Basic guarantee

```
void f(char const* n) {
    file outf(n, "w");
    if (outf.is_open()) {
        outf.put("The value is: ");
        outf.put(g());
    }
}
```

Strong guarantee

If the function throws an exception $\Rightarrow$ the program state is the state before the call

"commit or rollback semantics"="the operation is applied or not applied at all"

Strong guarantee

```
void f(char const* n) {
    // Part 1
    int temp = g();

    // Part 2
    file outf(n, "w");
    if (outf.is_open()) {
        outf.put("The value is:");
        outf.put(temp);
    }
}
```

Nothrow guarantee

A function never throws an exception

Nothrow guarantee

```
void f(char const* n) noexcept {
    try {
        file outf(n, "w");
        if (outf.is_open()) {
            outf.put("The value is: ");
            outf.put(g());
        }
    } catch(...) {
        //handling
    }
}
```