

P3105

constexpr std::uncaught_exceptions()

Author: Jan Schultke
Presenter: Jan Schultke
Audience: SG18
Project: ISO/IEC 14882 Programming Languages — C++,
ISO/IEC JTC1/SC22/WG21

Tokyo 2024
東京

Contents

1. Introduction
2. Motivating example
3. Proposal
4. Implementation

1. Introduction

Status quo

- `throw` cannot be used in a constant expression (`[[expr.const]]`)
- Some proposals seek change:
 - **P2996R1: Reflection for C++26** recommends exception handling for reflections
 - **P3068R0: Allowing exception throwing in constant-evaluation**
- Regardless:
 - `std::uncaught_exceptions()` can be `constexpr` (proposed)
 - `std::current_exception()` can be `constexpr` (proposed)

Goals

1. Future-proof existing code for `constexpr` exceptions.
2. Eliminate special cases in `constexpr` code.

2. Motivating example

`std::scope_success` (Library Fundamentals TS v3) invokes a function object when it goes out of scope without an exception being thrown.

```
scope_success::~~scope_success() noexcept( /* ... */) {  
    if (this->uncaught_on_creation >= std::uncaught_exceptions()) {  
        this->exit_function();  
    }  
}
```

```
constexpr scope_success::~~scope_success() noexcept( /* ... */) {  
    if (std::is_constant_evaluated() ||  
        this->uncaught_on_creation >= std::uncaught_exceptions()) {  
        this->exit_function();  
    }  
}
```

3. Proposal

Update [\[uncaught.exceptions\]](#) and [\[exception.syn\]](#):

```
constexpr int uncaught_exceptions() noexcept;
```

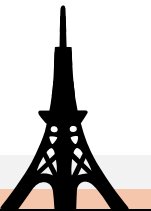
Update [\[propagation\]](#) and [\[exception.syn\]](#):

```
constexpr exception_ptr current_exception() noexcept;
```

Update [\[propagation\]](#) :

- `exception_ptr` becomes a literal type.
- Uses of `exception_ptr` null pointers become constant expressions.

Update feature-detection in [\[version.syn\]](#).



4. Implementation

```
constexpr int uncaught_exceptions() noexcept {
    if constexpr {
        return 0;
    }
    // TODO: what uncaught_exceptions() normally does ...
}
```

Other changes

- Analogous change to `current_exception`
- Add `constexpr` to `exception_ptr` members.
- Make sure that inline functions don't break ABI (`[gnu::used]`).
- **If `constexpr throw` becomes a thing, it's not so simple ...**
 - `exception_ptr` is simple to update (it's a class, wrapping `void*`)

References

Thomas Köppe; **N4806**: Working Draft, C++ Extensions for Library Fundamentals, Version 3
<https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/n4806.html>

Wyatt Childers et al.; **P2996R1**: Reflection for C++26
<https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2021/p2417r0.pdf>

Hana Dusíková; **P3068R0**: Allowing exception throwing in constant-evaluation
<https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2024/p3068r0.pdf>

Jan Schultke; **P3105** `constexpr std::uncaught_exceptions()` (latest revision)
<https://eisenwave.github.io/cpp-proposals/constexpr-uncaught-exceptions.html>