JUNE 16TH, 2025

P3637R0 INHERIT STD::META::EXCEPTION FROM STD::EXCEPTION

VICTOR ZVEROVICH

NEVIN LIBER

MICHAEL HAVA

P3560 ERROR HANDLING IN REFLECTION

Hagenberg

- POLL: `std::meta::exception` should inherit from `std::exception` (explore required modifications for `what()`)

SF	F	N	A	SA
4	3	5	1	3

- Attendance: 20 (IP) + 3 (R)
- Author's Position: SA
- Outcome: No Consensus
  - *VZ/NL/MH: but close*

- POLL: Forward P3560R1 to LWG for C++26

SF	F	N	A	SA
5	9	0	2	1

- Attendance: 21 (IP) + 3 (R)
- Author's Position: SF
- Outcome: Consensus in favor
- A: Name `exception` for ADL reasons
- **SA: Not inheriting from `std::exception`**

THE WHAT() ISSUE

- Later discussion showed encoding concern based on outdated information from LWG4087 Standard exception messages have unspecified encoding
  - Already addressed in [exception]:

```
constexpr virtual const char what () const noexcept;
```

Returns: An implementation-defined NTBS, which during constant evaluation is encoded with the ordinary literal encoding.

Remarks: The message may be a null-terminated multibyte string, suitable for conversion and display as a `wstring`. The return value remains valid until the exception object from which it is obtained is destroyed or a non-`const` member function of the exception object is called.

THE WHAT() ISSUE

- Encoding-wise this is compatible with P3560
- This addresses the motivation for not inheriting from `std::exception`

WHY INHERIT FROM STD::EXCEPTION?

- Consistency
  - This would be the first standard exception not inheriting from `std::exception`
  - Users typically inherit from `std::exception` as well
- Generic exception handling
  - It is a very common pattern to catch `std::exception` (including for tests, logging, etc.) when more specific info is not needed
  - Adding a new hierarchy means they have to add another `catch` block

WHY INHERIT FROM STD::EXCEPTION?

Minor differences from other standard exceptions

- Copy operations can throw
- Move operations exist

- Neither of these violates [exception]:

Except where explicitly specified otherwise, each standard library class T that derives from class exception has the following publicly accessible member functions, each of them having a non-throwing exception specification:
— default constructor (unless the class synopsis shows other constructors)
— copy constructor
— copy assignment operator

MAIN CHANGES TO P3560

```
class exception : public std::exception {
 private:
  optional<string> what_; // exposition only
  u8string u8 what_;      // exposition only
  info from_;             // exposition only
  source_location where_; // exposition only

 public:
  constexpr const char* what() const noexcept override;
 ~~consteval string what() const noexcept;~~
  // ...
};
```

```
consteval exception(u8string_view what, /* ... */) noexcept;
```

Effects : Initializes u8`what_` with `what` , `from_` with `from` and `where_` with `where`. If `what` can be represented in the ordinary literal encoding, initializes `what_` with `what`, transcoded from UTF-8 to the ordinary literal encoding.

CURRENT STATUS OF P3560

- The author of P3560 already added this to P3560R1!
    - Modulo bugs, such as they accidentally used private inheritance. They will have that fixed in P3560R2.
    - P3560 author(s) are now in favor of this change!

- We need LEWG to poll this.

- This research used resources from the Argonne Leadership Computing Facility, a U.S. DOE Office of Science user facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. DOE under Contract No. DE-AC02-06CH11357.

Argonne NATIONAL LABORATORY

U.S. DEPARTMENT of ENERGY