How C++ Exceptions work

Darius Engler

July 1, 2023

Overview

- The problem of implementing exceptions
- How it's described by the Itanium C++ ABI
- Exception Handling Representation in the ELF Format
- Alternative to std::exception

Simple Example

- The problem:
  - We need to know:
    - Which object to destroy
    - Which handler to take
    - If the handler is in the caller, how to get to the caller

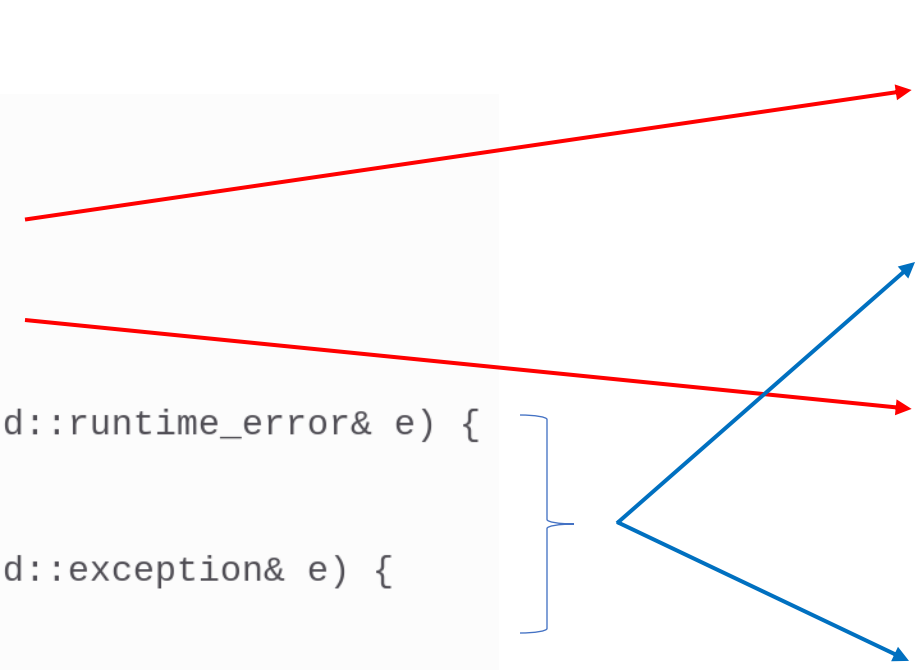
```
1  void foo() {
2      try {
3          bar();
4          A a;
5          bar();
6      }
7      catch (const std::runtime_error& e) {
8          // ...
9      }
10     catch (const std::exception& e) {
11        // ...
12     }
13 }
14
15 int main() {
16     try {
17         foo();
18     }
19     catch (...) {
20         // ...
21     }
22 }
```

Overview of how it works

- Scan the current function for exception handlers
  - Encode try / catch blocks in the binary
- If none is found, go to previous frame
  - Encode stack unwinding information in the binary

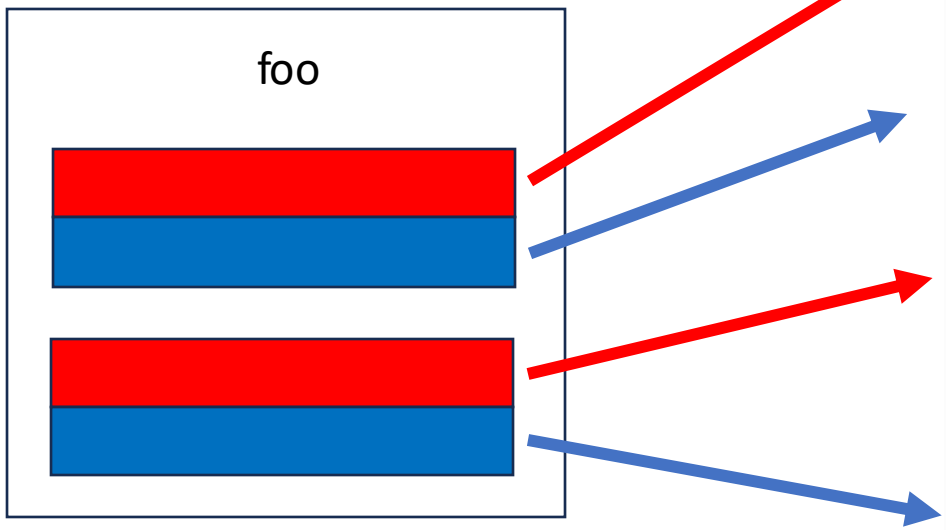
Destroying the right objects

```
void foo() {
    try {
        bar();
        A a;
        bar();
    }
    catch (const std::runtime_error& e) {
        // ...
    }
    catch (const std::exception& e) {
       // ...
    }
}
```



```
void foo() {
    try {
        bar();
    }
    catch (const std::runtime_error& e) {
        // ...
    }
    catch (const std::exception& e) {
       // ...
    }
    try {
        A a;
        bar();
    }
    catch (const std::runtime_error& e) {
      A::~A(a);
        // ...
    }
    catch (const std::exception& e) {
      A::~A(a);
       // ...
    }
}
```

Finding the right handler




```
void foo() {
    try {
        bar();
    }
    catch (const std::runtime_error& e) {
        // ...
    }
    catch (const std::exception& e) {
        // ...
    }
    try {
        A a;
        bar();
    }
    catch (const std::runtime_error& e) {
      A::~A(a);
        // ...
    }
    catch (const std::exception& e) {
      A::~A(a);
       // ...
    }
}
```

Stack unwinding

```
test(int):
    // <- return address is at (RSP)
    push    rbp
    // <- return address is at (RSP + 8)
    // <- previous RBP value is at (RSP)
    mov     rbp, rsp
    // <- the return address is at (RBP + 8)
    // <- previous RBP value is at (RBP)
    sub     rsp, 32
    mov     DWORD PTR [rbp-20], edi
    mov     DWORD PTR [rbp-4], 10
    call    func()
    mov     DWORD PTR [rbp-8], eax
    mov     edx, DWORD PTR [rbp-4]
    mov     eax, DWORD PTR [rbp-8]
    add     eax, edx
    leave
    // <- the return address is at (RSP)
    ret
```

ELF sections

SHDR

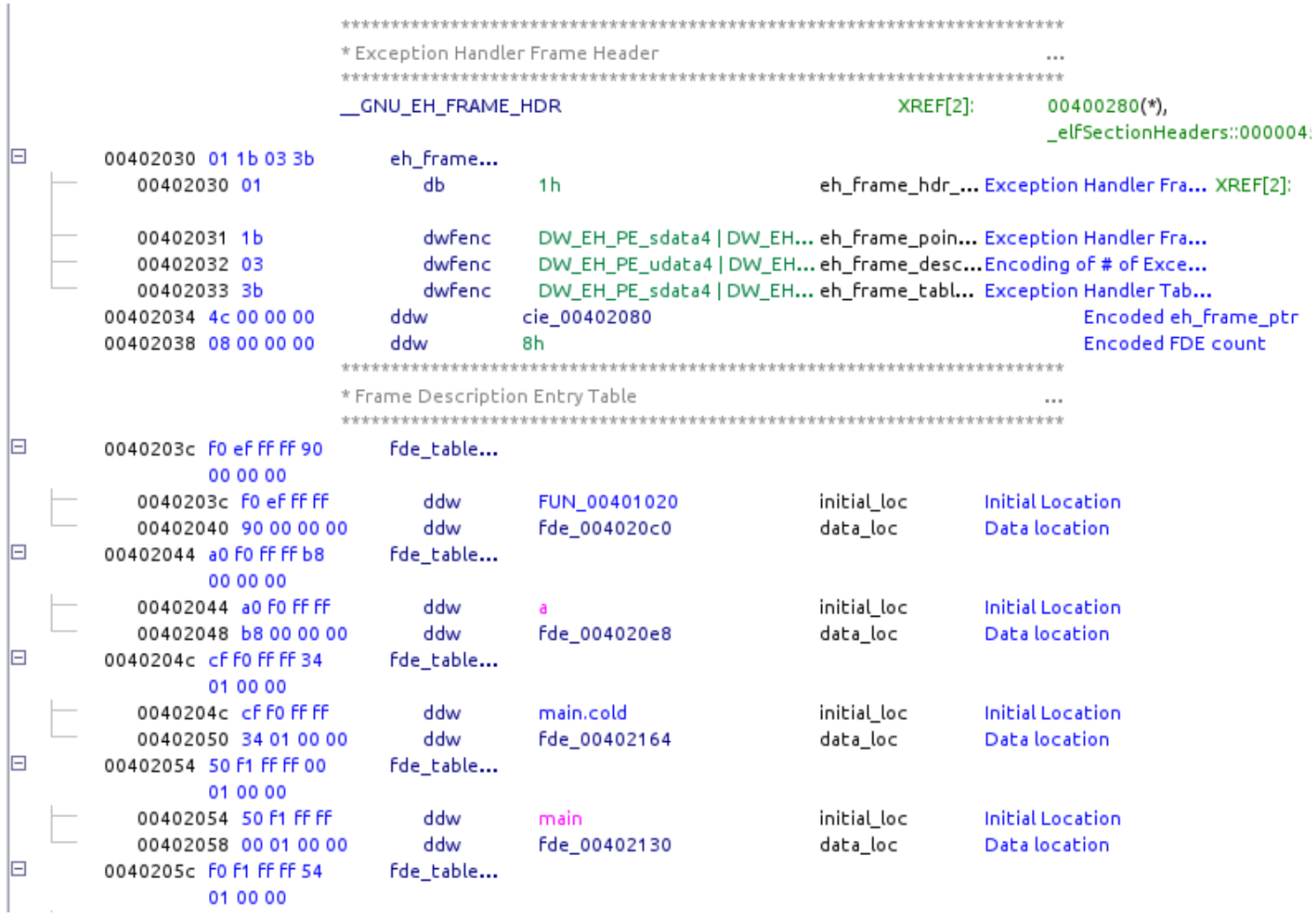
```
16 .eh_frame_hdr 00000054  0000000000402018  0000000000402018  00002018  2**2
                 CONTENTS, ALLOC, LOAD, READONLY, DATA
17 .eh_frame     00000180  0000000000402070  0000000000402070  00002070  2**3
                 CONTENTS, ALLOC, LOAD, READONLY, DATA
18 .gcc_except_table 0000004c  00000000004021f0  00000000004021f0  000021f0  2**2
                 CONTENTS, ALLOC, LOAD, READONLY, DATA
```

PHDR

```
EH_FRAME off    0x0000000000002018 vaddr 0x0000000000402018 paddr 0x0000000000402018 align 2**2
         filesz 0x0000000000000054 memsz 0x0000000000000054 flags r--
```

.eh_frame_hdr

- Based on DWARF
- List of FDE Pointers



Frame Descriptor Entry (FDE) / Common Information Entry (CIE)

- FDE
  - PC Range
  - LSDA pointer
  - Stack unwinding info
- CIE
  - Stack unwinding info
  - Personality function

```
                        ****************************************************************************
                        * Frame Descriptor Entry                                                   ...
                        ****************************************************************************
                        fde_00402130                                    XREF[1]:     00402058(*)
00402130 30 00 00 00        ddw     30h                                     (FDE) Length
00402134 24 00 00 00        ddw     cie_00402110                            (FDE) CIE Reference Pointer
00402138 48 f0 ff ff        ddw     main                                    (FDE) PcBegin
0040213c 95 00 00 00        ddw     95h                                     (FDE) PcRange
00402140 04                 uleb128 4h                                      (FDE) Augmentation Data Length
00402141 63 00 00 00        ddw     lsda_exception_table_004021a4           (FDE Augmentation Data) LSDA Data Poi...
00402145 42 0e 10 8c        db[31]                                          (FDE) Call Frame Instructions
         02 4a 0e 18
         86 03 41 0e ...
```

```
                        ****************************************************************************
                        * Common Information Entry                                                 ...
                        ****************************************************************************
                        cie_00402110                                    XREF[2]:     00402134(*), 00402168(*)
00402110 1c 00 00 00        ddw     1Ch                                     (CIE) Length
00402114 00 00 00 00        ddw     0h                                      (CIE) ID
00402118 01                 db      1h                                      (CIE) Version
00402119 7a 50 4c 52        ds      "zPLR"                                  (CIE) Augmentation String
         00
0040211e 01                 uleb128 1h                                      (CIE) Code Alignment
0040211f 78                 sleb128 -8h                                     (CIE) Data Alignment
00402120 10                 db      10h                                     (CIE) Return Address Register Column
00402121 07                 uleb128 7h                                      (CIE) Augmentation Data Length
00402122 9b                 dwfenc  DW_EH_PE_sdata4 | DW_EH_PE_pcrel | DW_EH_PE_indirect  (CIE Augmentation Data) Personality Fun...
00402123 f5 1e 00 00        ddw     <EXTERNAL>::__gxx_personality_v0         (CIE Augmentation Data) Personality Fun...
00402127 1b                 dwfenc  DW_EH_PE_sdata4 | DW_EH_PE_pcrel         (CIE Augmentation Data) LSDA Personali...
00402128 1b                 dwfenc  DW_EH_PE_sdata4 | DW_EH_PE_pcrel         (CIE Augmentation Data) FDE Encoding
00402129 0c 07 08 90        db[7]                                           (CIE) Initial Instructions
         01 00 00
```

Language-Specific Data Area (LSDA)

- Parsed by the personality function
- try/catch blocks of each function

```
                    ****************************************************************************
                    * Language-Specific Data Area                                              ...
                    ****************************************************************************
                    lsda_exception_table_004021f0                    XREF[2]:    00402145(*),
                                                                                 _elfSectionHeaders::000004d0(*)
004021f0 ff         dwfenc    DW_EH_PE_omit | DW_EH_PE_omit                      (LSDA) LPStart Encoding
004021f1 9b         dwfenc    DW_EH_PE_sdata4 | DW_EH_PE_pcrel | DW_EH_PE_indirect (LSDA) TType Encoding
004021f2 1d         uleb128   1Dh                                                (LSDA) TType Offset
004021f3 01         dwfenc    DW_EH_PE_uleb128 | DW_EH_PE_absptr                 (LSDA) Call Site Table Encoding
004021f4 0a         uleb128   Ah                                                 (LSDA) Call Site Table Length

                    * (LSDA) Call Site Record                                                ...
                    ****************************************************************************
004021f5 1f         uleb128   LAB_0040130f                                       (LSDA Call Site) IP Offset
004021f6 05         uleb128   5h                                                 (LSDA Call Site) IP Range Length
004021f7 92 01      uleb128   LAB_00401382                                       (LSDA Call Site) Landing Pad Address
004021f9 05         uleb128   5h                                                 (LSDA Call Site) Action Table Offset
```

Try block

catch address

Language-Specific Data Area (LSDA)

- Information about which type of exception is accepted by each catch block

```
                    ****************************************************************************
                    * (LSDA) Action Table                                                      ...
                    ****************************************************************************
004021ff 02             sleb128     2h                                  (LSDA Action Table) Type Filter
00402200 00             sleb128     0h                                  (LSDA Action Table) Next-Action Refere...
                    ****************************************************************************
                    * (LSDA) Action Record                                                    ...
                    ****************************************************************************
00402201 01             sleb128     1h                                  (LSDA Action Table) Type Filter
00402202 7d             sleb128     -3h                                 (LSDA Action Table) Next-Action Refere...
                    ****************************************************************************
                    * (LSDA) Action Record                                                    ...
                    ****************************************************************************
00402203 00             sleb128     0h                                  (LSDA Action Table) Type Filter
00402204 7d             sleb128     -3h                                 (LSDA Action Table) Next-Action Refere...
00402205 00 00 00       db[3]                                           -- alignment pad
```

```
                    ****************************************************************************
                    * (LSDA) Type Table                                                        ...
                    ****************************************************************************
00402208 10 1e 00 00        ddw         std::exception::typeinfo                    Type Reference
0040220c 04 1e 00 00        ddw         std::runtime_error::typeinfo                Type Reference
```

Call frame instructions (CFI)

- List of instructions
- Used to compute the CFA

  (Canonical Frame Address)
- Used to restore previous value of registers

Instruction	High 2 Bits	Low 6 Bits	Operand 1	Operand 2
DW_CFA_advance_loc	0x1	delta		
DW_CFA_offset	0x2	register	ULEB128 offset	
DW_CFA_restore	0x3	register		
DW_CFA_nop	0	0		
DW_CFA_set_loc	0	0x01	address	
DW_CFA_advance_loc1	0	0x02	1-byte delta	
DW_CFA_advance_loc2	0	0x03	2-byte delta	
DW_CFA_advance_loc4	0	0x04	4-byte delta	
DW_CFA_offset_extended	0	0x05	ULEB128 register	ULEB128 offset
DW_CFA_restore_extended	0	0x06	ULEB128 register	
DW_CFA_undefined	0	0x07	ULEB128 register	
DW_CFA_same_value	0	0x08	ULEB128 register	
DW_CFA_register	0	0x09	ULEB128 register	ULEB128 offset
DW_CFA_remember_state	0	0x0a		
DW_CFA_restore_state	0	0x0b		
DW_CFA_def_cfa	0	0x0c	ULEB128 register	ULEB128 offset
DW_CFA_def_cfa_register	0	0x0d	ULEB128 register	
DW_CFA_def_cfa_offset	0	0x0e	ULEB128 offset	
DW_CFA_def_cfa_expression	0	0x0f	BLOCK	
DW_CFA_expression	0	0x10	ULEB128 register	BLOCK
DW_CFA_offset_extended_sf	0	0x11	ULEB128 register	SLEB128 offset
DW_CFA_def_cfa_sf	0	0x12	ULEB128 register	SLEB128 offset
DW_CFA_def_cfa_offset_sf	0	0x13	SLEB128 offset	
DW_CFA_val_offset	0	0x14	ULEB128	ULEB128
DW_CFA_val_offset_sf	0	0x15	ULEB128	SLEB128
DW_CFA_val_expression	0	0x16	ULEB128	BLOCK
DW_CFA_lo_user	0	0x1c		
DW_CFA_hi_user	0	0x3f		

Call frame instructions (CFI)

```
                                        // cfa = rsp + 8
                                        // rip = cfa - 8
test(int):
        .cfi_startproc
        push    rbp
        .cfi_def_cfa_offset 16          // cfa = rsp + 16
        .cfi_offset 6, -16              // rbp = cfa - 16
        mov     rbp, rsp
        .cfi_def_cfa_register 6         // cfa = rbp + 16
        sub     rsp, 32
        mov     DWORD PTR [rbp-20], edi
        mov     DWORD PTR [rbp-4], 10
        call    func()
        mov     DWORD PTR [rbp-8], eax
        mov     edx, DWORD PTR [rbp-4]
        mov     eax, DWORD PTR [rbp-8]
        add     eax, edx
        leave
        .cfi_def_cfa 7, 8               // cfa = rsp + 8
        ret
        .cfi_endproc
```

```
CIE
  DW_CFA_def_cfa: r7 (rsp) ofs 8
  DW_CFA_offset: r16 (rip) at cfa-8
FDE
  DW_CFA_advance_loc: 4 to 0000000000401024
  DW_CFA_def_cfa_offset: 16
  DW_CFA_advance_loc: 9 to 000000000040102d
  DW_CFA_def_cfa_offset: 8
```

To summarize

- Walk through all the object files in the process
  - Use either /proc/self/maps or libc functions (e.g. "dl_iterate_phdr")
  - Check whether the current IP is within the object's address range
  - Find the EH_FRAME PHDR
  - Parse .eh_frame_hdr
  - Walk through each FDE from the binary search table
    - Parse associated CIE
    - Check that the current IP is within the range of the current FDE
    - Call the personality function
      - Walk through each LSDA entry to find the try block that matches
      - Check if the exception typeinfo matches the type filter of the catch block
    - If the personality doesn't find any handler, go to previous stack frame
      - Run the CFI program of the CIE and the FDE to compute the last PC

Fun fact about CFI instructions

Instruction	High 2 Bits	Low 6 Bits	Operand 1	Operand 2
DW_CFA_advance_loc	0x1	delta		
DW_CFA_offset	0x2	register	ULEB128 offset	
DW_CFA_restore	0x3	register		
DW_CFA_nop	0	0		
DW_CFA_set_loc	0	0x01	address	
DW_CFA_advance_loc1	0	0x02	1-byte delta	
DW_CFA_advance_loc2	0	0x03	2-byte delta	
DW_CFA_advance_loc4	0	0x04	4-byte delta	
DW_CFA_offset_extended	0	0x05	ULEB128 register	ULEB128 offset
DW_CFA_restore_extended	0	0x06	ULEB128 register	
DW_CFA_undefined	0	0x07	ULEB128 register	
DW_CFA_same_value	0	0x08	ULEB128 register	
DW_CFA_register	0	0x09	ULEB128 register	ULEB128 offset
DW_CFA_remember_state	0	0x0a		
DW_CFA_restore_state	0	0x0b		
DW_CFA_def_cfa	0	0x0c	ULEB128 register	ULEB128 offset
DW_CFA_def_cfa_register	0	0x0d	ULEB128 register	
DW_CFA_def_cfa_offset	0	0x0e	ULEB128 offset	
DW_CFA_def_cfa_expression	0	0x0f	BLOCK	
DW_CFA_expression	0	0x10	ULEB128 register	BLOCK
DW_CFA_offset_extended_sf	0	0x11	ULEB128 register	SLEB128 offset
DW_CFA_def_cfa_sf	0	0x12	ULEB128 register	SLEB128 offset
DW_CFA_def_cfa_offset_sf	0	0x13	SLEB128 offset	
DW_CFA_val_offset	0	0x14	ULEB128	ULEB128
DW_CFA_val_offset_sf	0	0x15	ULEB128	SLEB128
DW_CFA_val_expression	0	0x16	ULEB128	BLOCK
DW_CFA_lo_user	0	0x1c		
DW_CFA_hi_user	0	0x3f		

2.5 DWARF Expressions

DWARF expressions describe how to compute a value or specify a location. They are expressed in terms of DWARF operations that operate on a stack of values.

A DWARF expression is encoded as a stream of operations, each consisting of an opcode followed by zero or more literal operands. The number of operands is implied by the opcode.

In addition to the general operations that are defined here, operations that are specific to location descriptions are defined in Section 2.6 on page 38.

Fun fact about CFI instructions

Operation	Code	No. of Operands	Notes
DW_OP_rot	0x17	0	
DW_OP_xderef	0x18	0	
DW_OP_abs	0x19	0	
DW_OP_and	0x1a	0	
DW_OP_div	0x1b	0	
DW_OP_minus	0x1c	0	
DW_OP_mod	0x1d	0	
DW_OP_mul	0x1e	0	
DW_OP_neg	0x1f	0	
DW_OP_not	0x20	0	
DW_OP_or	0x21	0	
DW_OP_plus	0x22	0	
DW_OP_plus_uconst	0x23	1	ULEB128 addend
DW_OP_shl	0x24	0	
DW_OP_shr	0x25	0	
DW_OP_shra	0x26	0	
DW_OP_xor	0x27	0	
DW_OP_bra	0x28	1	signed 2-byte constant
DW_OP_eq	0x29	0	
DW_OP_ge	0x2a	0	
DW_OP_gt	0x2b	0	
DW_OP_le	0x2c	0	
DW_OP_lt	0x2d	0	
DW_OP_ne	0x2e	0	
DW_OP_skip	0x2f	1	signed 2-byte constant
DW_OP_lit0	0x30	0	
DW_OP_lit1	0x31	0	literals 0 .. 31 = (DW_OP_lit0 + literal)
...			
DW_OP_lit31	0x4f	0	

Continued on next page

It's turing complete!

Demo

```cpp
#include <iostream>

void bar(const char* flag) {
    throw 1;
}

void foo(const char* flag) {
    try {
        bar(flag);
    }
    catch (...) {
        std::cout << "Failure\n";
        exit(0);
    }
}

int main(int argc, char** argv) {
    try {
        foo(argv[1]);
    }
    catch (...) {
        std::cout << "Success\n";
    }
}
```

```
00000068 000000000000005c 0000006c FDE cie=00000000 pc=00000000004010c0..00000000004010e5
  DW_CFA_advance_loc: 4 to 00000000004010c4
  DW_CFA_def_cfa_expression (DW_OP_const1u: 0; DW_OP_breg12 (r12): 0; DW_OP_deref_size: 1; DW_OP_const1u: 112; DW_OP_
ne; DW_OP_plus; DW_OP_breg12 (r12): 1; DW_OP_deref_size: 1; DW_OP_const1u: 97; DW_OP_ne; DW_OP_plus; DW_OP_breg12 (r1
2): 2; DW_OP_deref_size: 1; DW_OP_const1u: 115; DW_OP_ne; DW_OP_plus; DW_OP_breg12 (r12): 3; DW_OP_deref_size: 1; DW_
OP_const1u: 115; DW_OP_ne; DW_OP_plus; DW_OP_breg12 (r12): 4; DW_OP_deref_size: 1; DW_OP_const1u: 119; DW_OP_ne; DW_O
P_plus; DW_OP_breg12 (r12): 5; DW_OP_deref_size: 1; DW_OP_const1u: 111; DW_OP_ne; DW_OP_plus; DW_OP_breg12 (r12): 6;
DW_OP_deref_size: 1; DW_OP_const1u: 114; DW_OP_ne; DW_OP_plus; DW_OP_breg12 (r12): 7; DW_OP_deref_size: 1; DW_OP_cons
t1u: 100; DW_OP_ne; DW_OP_plus; DW_OP_bra: 5; DW_OP_breg7 (rsp): 32; DW_OP_skip: 2; DW_OP_breg7 (rsp): 16)
```

```
$ ./demo.elf aIBFA
Failure
$ ./demo.elf eqoqoq
Failure
$ ./demo.elf password
Success
$
```

Alternative to std::exception

- C++23 std::expected
- Container for expected result / error code
- Can be used in combination with std::visit / std::variant

```cpp
auto someFunc() -> std::expected<SomeResult, std::variant<InvalidFile, FileNotFound>>
{
    return std::unexpected(FileNotFound{"a.bin"});
    //return std::unexpected(InvalidFile{"a.bin", "Invalid Header"});
    //return SomeResult{4};
}

int main()
{
    auto res = someFunc();
    if (res)
    {
        std::cout << "Result: " << res.value().computed << "\n";
    }
    else
    {
        std::visit(overloaded{
            [] (const InvalidFile& arg) {
                std::cout << "Invalid File: \"" << arg.filename << "\"; details: " << arg.details << "\n";
            },
            [] (const FileNotFound& arg) {
                std::cout << "File Not Found: \"" << arg.filename << "\"\n";
            },
        }, res.error());
    }
}
```

Sources

- https://itanium-cxx-abi.github.io/cxx-abi/abi-eh.html
- http://itanium-cxx-abi.github.io/cxx-abi/exceptions.pdf
- http://web.archive.org/web/20220524085632/https://www.intel.com/content/dam/www/public/us/en/documents/guides/itanium-software-runtime-architecture-guide.pdf
- https://refspecs.linuxfoundation.org/LSB_5.0.0/LSB-Core-generic/LSB-Core-generic/ehframechpt.html
- https://refspecs.linuxfoundation.org/LSB_5.0.0/LSB-Core-generic/LSB-Core-generic/dwarfext.html
- https://dwarfstd.org/doc/DWARF5.pdf
- https://wiki.dwarfstd.org/Exception_Handling.md
- https://martin.uy/blog/understanding-the-gcc_except_table-section-in-elf-binaries-gcc
- https://en.cppreference.com/w/cpp/utility/variant/visit
- https://en.cppreference.com/w/cpp/utility/expected

Questions?