Futuristic Error Handling

Error handling in C++ today and tomorrow

Dawid Pilarski

dawid.pilarski@panicsofware.com

Introduction

Why am I here?

Why should we bother with error handling?

Recommendable error handling mechanism

Which error mechanism would you choose?

- error codes?
- exceptions?

Error codes nowadays

The error codes.

- Old. C-compatible. Comes from assembly time.

The error codes.

- Old. C-compatible. Comes from assembly time.
- Machine friendly.

The error codes.

- Old. C-compatible. Comes from assembly time.
- Machine friendly.
- Super fast.

The error codes.

- Old. C-compatible. Comes from assembly time.
- Machine friendly.
- Super fast.
- Used till today.

Error code example

```c
int sqlite3_open( const char *filename, sqlite3 **ppDb );
```

Error code example

```c
int sqlite3_open( const char *filename, sqlite3 **ppDb );
```

```c
int open_status = sqlite3_open(/* ... */ );
if(open_status == SQLITE_OK){
  // make use of opened database
} else if( open_status == SQLITE_CANTOPEN_ISDIR ) {
  // handle the error
}
```

Handle the error

How to handle the error correctly?

Handle the error

How to handle the error correctly?

- `std::terminate()`

Handle the error

How to handle the error correctly?

- `std::terminate()`
- take the error callback

Handle the error

How to handle the error correctly?

- `std::terminate()`
- take the error callback
- propagate the error to the caller

Error codes - propagation

propagation

```
void foo_bar(int& errc /*...*/){
  errc = foo();
  // ...
  errc = bar();
  // ...
}
```

error translation

```
void foo_bar(foo_bar_errc errc&){
  foo_errc ferrc = foo();
  errc = translate_foo(ferrc);
  // ...
  bar_errc berrc = bar();
  errc = translate_foo(berrc);
}
```

C-style error codes summary

So we can see serious disadvantages (except for obvious advantages):

- success path same as error path

C-style error codes summary

So we can see serious disadvantages (except for obvious advantages):

- success path same as error path
- boiler plate code

C-style error codes summary

So we can see serious disadvantages (except for obvious advantages):

- success path same as error path
- boiler plate code
- cluttering code with translations

Error codes - modern approach

standard library support - what do we need?

- A way to define new error codes
- A way to distinguish domain of the error codes
- And to fix as many C-style issues as possible

Standard library support - what we get?

three types:

- `std::error_code`
- `std::error_category`
- `std::error_condition`

std::error_code in action

```
std::error_code errc;

is_regular_file("non_existent_directory", errc);

std::cout << errc << std::endl;
std::cout << errc.value() << std::endl;
std::cout << errc.message() << std::endl;
std::cout << errc.category().name() << std::endl;
```

```
$ system:2
$ 2
$ Nie mozna odnalexć określonego pliku.
$ system
```