Error handling in C++

Dawid Pilarski

Current state of error handling

Error codes description

Error handling and performance

There exist two common strategies for error handling:

- error codes
- exceptions

Error codes - example fopen

```
/* fopen example */
#include <stdio.h>
int main ()
{
  FILE * pFile;
  pFile = fopen ("myfile.txt","w");
  if (pFile!=NULL)
  {
    //do stuff
  } else {
    //how do I know if everything is fine?

    switch(errno){
    //
    }
  }
  return 0;
}
```

Error code - better approach

```
// declaration
int sqlite3_open( const char *filename,
                  sqlite3 **ppDb /* OUT: SQLite db handle */);

//usage
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
}
```

What else can be done to improve the code:

- enums
- error should be an input output argument (passed by reference) to force user handle it

C++11: yet another approach to the error codes

There are 3 types, that C++ 11 added to support `std::error_code`

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