

Promise and Future : Return an Exception

This lesson teaches how to return an exception while using `std::promise` and `future` in C++ for multithreading.

The function `executeDivision` displays the result of the calculation or the exception.

```
// promiseFutureException.cpp

#include <exception>
#include <future>
#include <iostream>
#include <thread>
#include <utility>

struct Div{
    void operator()(std::promise<int>&& intPromise, int a, int b){
        try{
            if ( b==0 ){
                std::string errMsg = std::string("Illegal division by zero: ") +
                    std::to_string(a) + "/" + std::to_string(b);
                throw std::runtime_error(errMsg);
            }
            intPromise.set_value(a/b);
        }
        catch (...){
            intPromise.set_exception(std::current_exception());
        }
    }
};

void executeDivision(int nom, int denom){
    std::promise<int> divPromise;
    std::future<int> divResult= divPromise.get_future();

    Div div;
    std::thread divThread(div,std::move(divPromise), nom, denom);

    // get the result or the exception
    try{
        std::cout << nom << "/" << denom << " = " << divResult.get() << std::endl;
    }
    catch (std::runtime_error& e){
        std::cout << e.what() << std::endl;
    }

    divThread.join();
}

int main(){
```

```
int main(){  
  
    std::cout << std::endl;  
  
    executeDivision(20, 0);  
    executeDivision(20, 10);  
  
    std::cout << std::endl;  
  
}
```



The promise deals with the issue that the denominator is 0. If the denominator is 0, it sets the exception as return value:

`intPromise.set_exception(std::current_exception())` in line 20. Following that, the future has to deal with the exception in its try-catch block (lines 33 - 38).

Here is the output of the program.

If possible, use tasks as a safe replacement of condition variables.