Confusion around noexcept [duplicate]

Asked 9 years, 5 months ago Modified 9 years, 5 months ago

Viewed 533 times

4

This question already has answers here:

When should I really use noexcept? (10 answers)

Closed 9 years ago.

After watching many videos, reading a book, I am unclear about when and when not to use noexcept.

All of the books say that you should only use noexcept when a function WILL NEVER EVER throw.

I think it should be used otherwise. Many say that functions that allocate shouldn't be noexcept, but what if I don't want to catch those errors, and a call to `std::terminate` is acceptable?

In short, should noexcept be used on functions that will never throw, or on all functions except the ones you would like to catch exceptions from.

IMHO some exceptions don't need to be caught (ie out of memory etc)

c++ c++11 noexcept

Share

Improve this question

Follow

asked Jul 15, 2015 at 3:59

Russell Greene

2,261 • 18 • 31

`some exceptions don't need to be caught (ie out of memory etc)` Seriously? – deviantfan Jul 15, 2015 at 4:02

3 Yes, a call to std::terminate is fine for me. How does one recover from that? – Russell Greene Jul 15, 2015 at 4:03

1 And no, don´t make a throwing function noexcept. If you don´t want to catch it, you can do it like you want, even without noexcept. – deviantfan Jul 15, 2015 at 4:03

1 `How does one recover from that?` Eg. writing some log, closing files, telling the user if he want to terminate or retry, etc.etc. Depends on what the program is doing. – deviantfan Jul 15, 2015 at 4:05

1 To follow up with what @deviantfan is getting at. If you want to terminate your program, terminate your program. Don't rely on your abuse of a keyword to do it for you. If you make a habit of doing things like that, it will kick you in the ass one day. – Taekahn Jul 15, 2015 at 7:01

1 Answer

Sorted by: Highest score (default)

The marker `noexcept` is a gurantee from the developer to the compiler that the function will never throw.

4

So you should add it to functions that you know **Should** never throw. If these functions do for some obscure and unknowable reason throw the only thing the compiler can do is terminate the application (as you guaranteed something that should not happen). Note: from a function marked noexcept you should probably not call another function unless it is also marked noexcept (just like const correctness you need to have noexcept correctness)

Where you should use it:

```
swap()   methods/functions.
         Swap is supposed to be exception safe.
         Lots of code works on this assumption.

Move Constructor
Move Assignment.
         The object you are moving from should alrea
         fully formed so moving it is usually a proc
         swapping things around.

         Also be marking them noexcept there are cer
         optimizations in the standard library that

   Note: This is usually the case.
         If you can not guarantee that move/swap sem
         exception safe then do not mark the functio
```

You don't want to call terminate on all exceptions. Most of the time I would allow the exception to unwind the stack calling destructors and releasing resources correctly.

It it is not caught then the application will terminate.

But most complex applications should be resilient to exceptions. Catch discard the initiated task log the

exception and wait for the next command. Sometimes you still want to exit but just because my smudge operation in the graphics application fails does not mean I want the application to exit ungracefully. I would rather that the smudge operation is abandoned resource re-claimed and the application goes back to normal operations (so I can save exit and restart).

Share Improve this answer Follow

edited Jul 15, 2015 at 7:02

answered Jul 15, 2015 at 6:46

Loki Astari

264k ● 86 ● 342 ● 571