Subject: Capture division by zero
Posted by koldo on Fri, 16 Oct 2020 07:15:33 GMT
View Forum Message <> Reply to Message

I would like to capture floating point errors.
For now in case of error the program execution follows, but you can find in the doubles, things like INF or NAN.
This behaviour is very conservative for my apps, and I'd prefer in some situations to capture these situations stopping the execution.

I have tried it unsuccessfully using:
- signal(SIGFPE, ...)
- _controlfp_s()
- _set_se_translator()

Any help will be acknowledged!

PD. Somebody could say "just check that all data is adequate before using it in division, sqrt, ...". That's true, but difficult and cumbersome in some situations using uncontrolled data sources where sometimes even doing lots of "if" to check the consistency of data, that's not enough.

Subject: Re: Capture division by zero
Posted by Didier on Sat, 17 Oct 2020 21:53:10 GMT
View Forum Message <> Reply to Message

Hello Koldo,

Maybe this could help you:
* https://stackoverflow.com/questions/9619014/trapping-quiet-nanhttp ://
* https://en.cppreference.com/w/cpp/numeric/fenv

Subject: Re: Capture division by zero
Posted by Klugier on Sun, 18 Oct 2020 19:21:50 GMT
View Forum Message <> Reply to Message

Hello Koldo,

From my personal experience you should identify all the places when division by zero can occurred and handle error separately. To find all places you should use static analyzer. I saw that cppcheck and clang-tidy have tools to detect problem such this.

I also suggest to cover each of the code that can cause division by zero problem with unit tests. Some time ago I added plugin/gtest to the bazaar to make code more bulletproof. One of the reason programmers use unit tests is to test corner cases easily.

Resources:
- https://sourceforge.net/p/cppcheck/wiki/ListOfChecks/ (Just search of division)

Klugier

Subject: Re: Capture division by zero
Posted by koldo on Sun, 18 Oct 2020 19:38:44 GMT
View Forum Message <> Reply to Message

Thank you Didier, now it works

I have added some additional exception handling in DEBUG mode in SysInfo activated with ExceptionHandler() call.
It includes floating point exception handling.

It might be worth including something like this in main U++, that already includes signal(SIGFPE and other, only valid in Linux.

Subject: Re: Capture division by zero
Posted by koldo on Mon, 19 Oct 2020 05:57:14 GMT
View Forum Message <> Reply to Message

Quote:From my personal experience you should identify all the places when division by zero can occurred and handle error separately. I agree with you. However the DEBUG mode serves to capture unexpected situations, and this is to add more power to DEBUG mode detection of unexpected and nasty events. In addition to numerical errors, I have included unhandled exceptions, pure virtual functions and invalid parameter handler.

As soon as I tried it with one of my applications, I found a division by zero in an initialization that, by pure luck, had never caused any problem. This alone has been worth the effort, and I think I will capture many more ugly situations like this.

Edit: PD: In a perfect world, DEBUG would be useless

Subject: Re: Capture division by zero
Posted by Didier on Mon, 19 Oct 2020 18:54:28 GMT
View Forum Message <> Reply to Message

Hello Koldo and Klugier,

I agree with Koldo, although unit testing is a must have, you cannot assume that ALL is tested 100% and that it will stay that way in the future (specially and very big projects.
So adding so additionnal checks that are application wide is always a good path.

In addition to catching signals I would also print the stack trace automatically (and eventually write automatically to a file) : this is very usefull in production envs (You can get better knowledge of what went wrong at you're clients site)

Although in release mode stacktrace is less precise (du to optimisations), it is stille very helpfull (as long as you conserve the debug info associated to you're release code)

Subject: Re: Capture division by zero
Posted by koldo on Tue, 20 Oct 2020 06:02:42 GMT
View Forum Message <> Reply to Message

Quote:In addition to catching signals I would also print the stack trace automatically (and eventually write automatically to a file) Did you mean a minidump file? it is very easy to get, but I did not include it because I did not know how to analise it.

Subject: Re: Capture division by zero
Posted by Didier on Tue, 20 Oct 2020 18:58:36 GMT
View Forum Message <> Reply to Message

Well I was more thinking about using one of the following options
 * Glibc provides the very handy backtrace() and backtrace_symbols() ==> should be available on windows with CLANG
 * http://boostorg.github.io/stacktrace/stacktrace/getting_star ted.html (never tried)
 * Or with MSVC StackWalker library (never tried neither)

This will print the stack trace as it would be in the debugger : can get very very helpful with client crashes

---

Subject: Re: Capture division by zero
Posted by koldo on Wed, 21 Oct 2020 05:55:42 GMT
View Forum Message <> Reply to Message

In DEBUG mode is clear, all detections are ON even though it produces performance loss.

However in RELEASE mode is not so clear. I would support to include and enable systems that without performance penalty would catch unexpected errors. IMHO from an end user point of view, it is better an ugly panic window than closing the program window with no warning.

Do you agree? Would you work with me in this direction?

Subject: Re: Capture division by zero
Posted by Didier on Wed, 21 Oct 2020 18:29:49 GMT
View Forum Message <> Reply to Message

Hello Koldo,

I would be a pleasure to work with you on this simple yet important subject (and that not so many developpers know about)
I even have a simple package that contains just the print stacktrace (only for linux though)

Maybe this could also end up with a build option : "keep a symbols file in release mode", like you do when you strip a binary so the stacktrace in realease mode can be interpreted

One question : there is the 'Crash' package in Uppsrc, I don't know how to use it but I suspect it could be of great interest...

Subject: Re: Capture division by zero
Posted by Klugier on Mon, 26 Oct 2020 22:51:46 GMT
View Forum Message <> Reply to Message

Hello Koldo,

Influential people think that divisions by zero or negative square roots are admissible in a good code. I do not agree.

This is not the problem. Error handling or problematic value handling is very important. In production software you should always do that without compromising. The problem with "CrashHandler" is that it forces error handling to all package consumers. It is not even an option - you just create global variable:

static CrashHandler crash;

Library or component provider should avoid it at all cost. You should in the documentation tell that certain function could raise exception or returns error value. If your code could read external value and then it could cause problem you are obligated to validate it and in case of problem do something with it.

At the beginning of the thread I suggest to use static analyzer to find all this divided by zero problems and handle it properly. This is the best strategy for library and professional software. I agree that minidumps are extremely helpful to identify problems on production, but it must be handled on application level - never on library level.

```
/**
 * PUBLIC API DOCUMENTATION
```

```
 * In case when divider is 0 std::runtime_error is thrown.
 */
int Calculate(int divider)
{
    if (divider == 0)
    {
        throw std::runtime_error("Calculate(): Divider can not be zero...");
    }

    return 20 / divider;
}

// Now the client could deiced what to do with the exception - it can be handled locally or globally in main function, but it is still the consumer choice what to do with it
// In case of global handler provided by library the decisions were taken from application creator.


// ... Below line may even decrese the performance, because on each new allocation NewHandler will be called.
std::set_new_handler(NewHandler);
std::set_unexpected(UnexpectedHandler); // <- What if somebody defines this handler and you overriding it - you should check this at least...
```

To be clear my overall goal is to pick up the quality of our packages. Never decrees or compromise on error handling, so you are my ally.

Klugier

Subject: Re: Capture division by zero
Posted by mirek on Tue, 27 Oct 2020 08:27:48 GMT
View Forum Message <> Reply to Message

Didier wrote on Wed, 21 October 2020 20:29
One question : there is the 'Crash' package in Uppsrc, I don't know how to use it but I suspect it could be of great interest...

Sins of the past. Long time ago that was used for something with crash dumps. Will remove.

Mirek

Subject: Re: Capture division by zero
Posted by koldo on Tue, 27 Oct 2020 09:58:06 GMT
View Forum Message <> Reply to Message

CrashHandler works only in DEBUG and is connected to TheIDE so when debugging you can capture floating point (FP) and other problems. Production code could not use CrashHandler.

This feature is already included in U++ (file App.cpp has signal(SIGFPE, and more), although it inside an #ifdef Linux.

Division by zero and other FP errors, are errors and have to be debugged. Just imagine two situations:
- The calculation of your bank account balance is infinite
- The controller that determines the direction of your car with automatic driving sets an infinite angle

I do not want my code to fall in these situations.

Subject: Re: Capture division by zero
Posted by mirek on Tue, 27 Oct 2020 10:53:52 GMT
View Forum Message <> Reply to Message

koldo wrote on Tue, 27 October 2020 11:36Quote:Ah, that explains it. So you have like 2 weeks of experience with this issue After thirty five years programming, I think it's better late than never.
If I'd be an expert in this issue, I wouldn't ask for advice.

Well, you did not seem to ask for an advice. You actually seemed to force an answer on us

Mirek

Subject: Re: Capture division by zero
Posted by koldo on Tue, 27 Oct 2020 16:08:34 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 27 October 2020 11:53koldo wrote on Tue, 27 October 2020 11:36Quote:Ah, that explains it. So you have like 2 weeks of experience with this issue After thirty five years programming, I think it's better late than never.
If I'd be an expert in this issue, I wouldn't ask for advice.

Well, you did not seem to ask for an advice. You actually seemed to force an answer on us

Mirek
In previous post, I was asking for advice to Didier and anybody. You are also invited

---

Subject: Re: Capture division by zero
Posted by Didier on Tue, 27 Oct 2020 17:58:47 GMT
View Forum Message <> Reply to Message

Hello Koldo,

Quote:f your code considers these situations as normal, my friend, you are screwed
I don't know what made you think that was my point of view, and it is not at all my vision (rather the contrary), the only thing I say is that code comes from everywhere and quality is not always as it should be.
So everything that can be done to detect and correct this must be done and made available (including detecting divisions by 0, overflows, ...)

And besides, no one is perfect, errors are everywhere all the time (especially with big teams): so you have to deal with them

---

Subject: Re: Capture division by zero
Posted by koldo on Tue, 27 Oct 2020 18:21:40 GMT
View Forum Message <> Reply to Message

Didier wrote on Tue, 27 October 2020 18:58Hello Koldo,

Quote:f your code considers these situations as normal, my friend, you are screwed
I don't know what made you think that was my point of view, and it is not at all my vision (rather the contrary), the only thing I say is that code comes from everywhere and quality is not always as it should be.
So everything that can be done to detect and correct this must be done and made available (including detecting divisions by 0, overflows, ...)

And besides, no one is perfect, errors are everywhere all the time (especially with big teams): so you have to deal with them
Sorry Didier

There must be a misunderstanding. My point of view is debugging all errors, including floating point.
Absolutely I was not referring to you and your point of view.

Subject: Re: Capture division by zero
Posted by jjacksonRIAB on Mon, 25 Jul 2022 01:37:48 GMT
View Forum Message <> Reply to Message

Quote:
The calculation of your bank account balance is infinite

Anyone have an opinion on this one? I'm of the opinion that floats should not be used to represent any kind of currency or bank balance and I tended to replace them in C# with Decimal when I worked for an accounting department. I haven't checked but does U++ have a fixed-point decimal type for dealing with currency and other numbers where float accuracy is not good enough? If not

maybe one should be made.

EDIT Just realized I didn't provide a reason why not to use them. What I found is it wasn't NaN-stuff (though that could be a problem), but mainly because floats have a tendency to accumulate errors over successive calculations - that and the fact that you might want Banker's rounding or some other kind of rounding algorithm. Another reason I ditched floats is because they can't produce consistent results on different architectures, different CPUs, different families of CPUs, or even the same CPU. In one particularly egregious case I had a bad video driver produce wildly inaccurate results on a single accountant's machine and I could only guess that some of the computation was being offloaded to the GPU.

Looking further into C#'s decimal type, it appears it uses arbitrary precision floats underneath. There are number of libraries for doing this in C++, some listed here:
https://en.wikipedia.org/wiki/List_of_arbitrary-precision_ar ithmetic_software

Since they are still floats it's not a cure all for the problem of cumulative errors so I was wrong about that. That said, I still think a fixed-point decimal class would be desirable.

Subject: Re: Capture division by zero
Posted by mirek on Mon, 25 Jul 2022 07:39:01 GMT
View Forum Message <> Reply to Message

jjacksonRIAB wrote on Mon, 25 July 2022 03:37
Anyone have an opinion on this one? I'm of the opinion that floats should not be used to represent any kind of currency or bank balance and I tended to replace them in C# with Decimal when I worked for an accounting department. I haven't checked but does U++ have a fixed-point decimal type for dealing with currency and other numbers where float accuracy is not good enough? If not maybe one should be made.

Interesting question. For what is worth, I was doing applications dealing with money for years and while I know the theory, I have simple used double the whole time, to no ill effects.

I guess that in the end, you are usually rounding to 2 decimals after the point. You would need to sum millions of values to get a difference in printed output, so for the most time, it is just fine.

That said, Decimal datatype would be fine.

BTW, I do not think float or not float is the issue here, but underlying representation. Now I was not looking into this for some time, but I think that to be "financial correct" you need decimal representation instead of binary (some form of BCD) so that you can represent decimal fractional numbers exactly.

Mirek

Subject: Re: Capture division by zero
Posted by jjacksonRIAB on Mon, 25 Jul 2022 09:30:25 GMT
View Forum Message <> Reply to Message

Thanks, Mirek, always worth hearing your opinion on the subject. Haven't done a lot of research into floating point because I'm not a heavy math guy so much of the math here:

https://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.h tml

was beyond me. Also watched this video about posits and found the subject interesting.

https://www.youtube.com/watch?v=aP0Y1uAA-2Y

Perhaps I'm just leery about the entire subject because other devs have warned me off doing it. I'll look more into BCD, the last I heard about that was in an x86 assembly class relating to hardware instructions like AAA and packed decimal we never went into. I'm assuming there are SIMD instructions for the same concept.