

Forum:[Hack Forums](#)

Thread: [std::shared_ptr destructor](#)

zNinja - December 14, 2017, 7:19 am

Hi, I have `std::map` of shared pointers,

Shared pointers are created and destroyed in main thread, but objects which pointer holds are used in multiple threads.

Is there are possibility when shared pointer is destroyed in main thread it destroys object which hi holds in other thread.

I mean that destructor of object which shared pointer holds is called from other thread then main where shared pointer is destroyed?

0x45 - December 14, 2017, 10:11 am

Since it's a pointer, yes, if memory is cleared in main thread, other threads would be pointing to empty memory as well

zNinja - December 14, 2017, 1:23 pm

(Dec 14, 2017 11:11 AM)0x45 Wrote: [\[CITE\]](#) Since it's a pointer, yes, if memory is cleared in main thread, other threads would be pointing to empty memory as well

I removed shared pointers from project to be sure.

Of course, I know that other threads will point to an empty memory, but now for knowledge purpose.

I am sure destructor will be called only once, but are you saying that it is not predictable in what thread destructor will be called if shared pointer is destroyed in main thread?

0x45 - December 14, 2017, 2:52 pm

if you have `del p_thread` in main thread then destructor for that pointer will be called only in main thread.

zNinja - December 14, 2017, 4:38 pm

Thanks, I think the same.

For some strange reason I catch my destructor is called from other thread then main, so it comes to my mind is there are something about shared pointers that i do not know.

Other what comes to my mind is when assertions fail in other thread and `abort()` is called to close the program, probably in that case object do not have some predicted order in what thread they are destroyed.

0xDEAD10CC - December 15, 2017, 11:21 am

Smart pointers are objects, there's nothing special about them, they are local variables that are owned by a specific thread just like anything else, they only differ from regular pointers when it comes to the lifetime of that object and what happens when they go out of scope essentially. It's quite clear if you look up RAI.