Subject: Deadlock - what LeaveGMutexAll does?
Posted by pete82 on Sun, 01 Jan 2012 11:35:28 GMT
View Forum Message <> Reply to Message

Hi!
I'm creating simple network monitoring application. I've two threads - one thread serves network and the other is main. I've also some lua bindings, so i can do some fast prototyping and modify code on the fly (which works). Lua is single threaded, so i've mutex, call it lM.
Now because i've overriden Paint which is called with GuiLock held (as i understand), i have to acquire GuiLock and then lM to prevent deadlocks (the Paint is served in my lua code) for every lua access (i'm painting in lua often anyway).
I've also some timers which are timing GUI refreshes from network thread, and the refreshes are called from main thread (this was not intentional, but seems good to me).
I've still some deadlocks, which i'm unable to solve. So i fired up good old ollydbg (theide was not showing correct info) and it seems to me, that the problem is LeaveGMutexAll in Ctrl::ICall.
The network thread is holding lM and GuiLock, the timer fires or smt else and the GuiLock is released. So the first thread can acquire it, but can't acquire lM.
Or am i wrong? How can i solve this?
The call sequence(or stack) is smt like this

main->timer->GuiLock->waiting on lM to call timer function
thread->GuiLock->lM->Lua->refresh->waiting in UPP

I can provide more precise stack traces, but it takes some time to reproduce...
Thnx for answers.

Subject: Re: Deadlock - what LeaveGMutexAll does?
Posted by mirek on Sat, 07 Jan 2012 09:50:04 GMT
View Forum Message <> Reply to Message

Well, looks like classical deadlock...

I guess for now the only thing to comment here is the purpose LeaveGMutexAll... "GMutex" (mutex for GUI) is designed to work recursively, means Lock/Unlock can be nested in single thread. GMutex keeps 'level' of those locks/unlocks.

Then in ICall we need to "completely unlock" GMutex, so that if called from another thread than non-main, main thread can lock GMutex regardless of how many times it was locked in that another thread. So LeaveGMutexAll unlocks 'level' times and returns level, so that after finishing the function in the main thread, it can be locked that many times again (and gui locking in other thread is not corrupted).

Subject: Re: Deadlock - what LeaveGMutexAll does?
Posted by pete82 on Sat, 07 Jan 2012 12:21:34 GMT
View Forum Message <> Reply to Message

I solved it by queing all "Refresh" and all the other GUI calls to the main thread (which is quite uncomfortable).

Thank you for explanation, but do i understand correctly (also looking at the code), that the main thread is unlocking lock which it doesn't own? This looks really bad to me. The classic method to avoid deadlocks is to acquire them in some fixed order, but it fails then.
Basically, can this happen? (GM-GMutex,OM-other mutex,T-thread,MT-main thread)
T: GM->OM->GM unlocked, preempted
MT:waiting on GM->GM->waiting on OM
T: ->waiting on GM

Subject: Re: Deadlock - what LeaveGMutexAll does?
Posted by mirek on Sat, 07 Jan 2012 14:04:53 GMT
View Forum Message <> Reply to Message

pete82 wrote on Sat, 07 January 2012 07:21I solved it by queing all "Refresh" and all the other GUI calls to the main thread (which is quite uncomfortable).

Thank you for explanation, but do i understand correctly (also looking at the code), that the main thread is unlocking lock which it doesn't own?

No, non-main thread is unlocking its own lock on GUI, so that main thread can proceed.
