

Re: Obscure mutex problem

Source: <http://linux.derkeiler.com/Newsgroups/comp.os.linux.development.apps/2007-09/msg00028.html>

- *From:* Ulrich Eckhardt <doomster@xxxxxxxx>
 - *Date:* Wed, 05 Sep 2007 07:44:41 +0200
-

David Given wrote:

The problem is that on all the machines I have access to, it runs fine... and since I can't duplicate the problem, I can't debug it! All I have are anecdotal reports from users. I've managed to persuade some to run customised versions and provided assorted debug logs, but there doesn't seem to be any common factor to allow me to figure out why it works (reliably) on some machines and fails (reliably) on others.

I think logfiles are the wrong way. Rather, make them connect to the process using gdb and give you stack traces of all threads.

The way the program works is as follows: there are multiple threads of execution, where normally only one is running at a time.

This is definitely not true on a multi-CPU machine, where two threads can run at the same time. Keep that in mind!

– when initially writing the code, I discovered that I had to set `PTHREAD_MUTEX_RECURSIVE` to make the mutexes work at all... but then later found this was no longer necessary. No doubt this is due to another change I made, but I still don't understand it.

Well, either you design your code to only require a single locking operation, then it doesn't have to be recursive or you don't. Adding 'recursive' doesn't fix anything, it just hides a broken design.

<http://spey.cvs.sourceforge.net/spey/spey/src/Threadlet.cc?view=markup>

Well, from looking at it, just some notes:

– uses macros for creating loops, fails to avoid double evaluation of macro parameters

Re: Obscure mutex problem

- uses a nonportable GCC extension (typeof)
- actively hiding errors by e.g. casting the returnvalue of `pthread_key_create`
- using C-style casts
- catching exceptions by value instead of reference-to-const
- dereferencing a pointer after invoking 'delete' on it
- should use RAII for managing locks
- you are starting a thread that uses a virtual function of an object that might not be fully constructed
- as someone noted, you are not controlling compiler-generated copy constructor and assignment operator

Note that these all look like superficial problems but I'd fix those first and then start tackling the real problem. Odds are that the real problems go away once you cleaned up your code. BTW: I'd suggest getting the book "C++ Coding Guidelines" by Sutter and Alexandrescu, it lists such pitfalls and explains them and many others further.

Uli

.