

Re: gdb help: debugging a segfault in boost::shared_ptr

Source: <http://linux.derkeiler.com/Newsgroups/comp.os.linux.development.apps/2006-12/msg00054.html>

- *From:* "phear" <phear.dude@xxxxxxxxxx>
 - *Date:* 6 Dec 2006 11:46:51 -0800
-

I am developing a database connectivity plugin for the Remedy Action Request System (ARS). ARS is a platform for developing form based applications, usually for request/ticket/case management. In order to support integrations to multiple database systems through native ARS forms, I need a plugin that can be configured to access multiple configured data sources.

Due to restrictions in the ARS plugin interface and performance requirements, I've developed a database pool for the plugin so the time used to connect to the data sources will be greatly reduced. The pool is very simple in its design. It holds a list of database connections not in use, and will upon request return one of these, or if none is available create a new one. The requester is required to give the adapter back to the pool when done with, so the pool can insert it back to the available list. The pool will also check if the connections are expired before returning them, and remove them if they are.

The shared_ptrs are responsible for cleaning up the database connection. The initial idea was that if an exception occurred in a thread holding a database connection, the shared_ptr would automatically clean it up. This might not be possible though, since the pool might have to keep track of the connections in order to limit the number of open connections. Unless this can be handled on another level.

The pointers are never passed between threads outside of the pool. Access to the pool is synchronized in the only two methods used to access the pool, GetAdapter() and ReleaseAdapter(). The thread safety of the pointers should therefore not be an issue anyway. This is also indicated by the fact that this bug also occurs when using ordinary pointers.

Thanks for enlightening me about the details of shared_ptr and atomic_ptr though. That will probably save my day sometime :)

On Dec 6, 5:48 pm, Joe Seigh <jseigh...@xxxxxxxxxx> wrote:

phear wrote:

Re: gdb help: debugging a segfault in boost::shared_ptr

Re: gdb help: debugging a segfault in boost::shared_ptr

Putting a lock on IDatabaseAdapter.GetAdapter() did not help. Locking Instance::GetEntry fixed it, but this ofcourse defies the purpose of multi-threading.

I have tried using normal pointers instead, and although a lot more rare, it still segfaults occasionally.

I am currently following a lead from valgrind, but I'm not going to hold my breath just yet.[...]

It's not a problem with shared_ptr or with the debugger. The "lock-free" stuff in shared_ptr is a red herring. It's an implementation issue to make it thread-safe and could just have easily been lock based and is lock based on some platforms. Thread-safe means that shared_ptr handle internally shared data safely, i.e. the reference count. It does not mean it is atomically thread-safe like Java pointers and that you don't need some form of external synchronization if the pointers are shared.

If you want to see what an atomically thread-safe refcount pointer looks like, take a look at atomic_ptr in http://atomic_ptr_plus.sourceforge.net/

We can't give you too much specific advice on how to solve your problem since we know few specifics on what you are trying to accomplish.

--

Joe Seigh

When you get lemons, you make lemonade.
When you get hardware, you make software.