Couldn’t we fix the lackey catastrophe by using #pragma init_seg(user)?

devblogs.microsoft.com/oldnewthing/20170508-00

May 8, 2017

Raymond Chen

The lackey catastrophe is the nickname I gave to a problem where the lackey hired by the C runtime to destruct global objects runs too late. Specifically, it may run after other DLLs have already shut down, which means that you will have a problem if there is an object whose destructor relies on one of those already-shut-down DLLs.

Commenter ZLB remarked, “One way to fix this would be to use #pragma init_seg(user) to force the object to be constucted last and destructed first.”

The `init_seg` pragma helps control the order in which objects within a module are constructed and destructed. But note the phrase “within a module”.

The global variables in a module are constructed in three phases. The “compiler” phase is reserved for the compiler, and its objects construct first. The “lib” phase is intended for libraries (and clearly, in this context, we mean static libraries), and its objects construct next. The “user” phase is intended for application variables, and its objects construct last. (Destruction is in reverse order of construction.)

But all of these phases are internal to the C runtime. They control the order in which the C runtime constructs and destructs objects as part of standard module initialization and termination. But they don’t have any effect on when the module as a whole is initialized or terminated.

Suppose you have Super Elite status at an airline, eligible for priority boarding for any flight. You get to the airport for the 4pm flight to San Francisco, and the moment they announce that they are commencing boarding, you saunter up to the gate, flash your Super Elite status badge, and stroll right onto the airplane ahead of everybody else.

That’s great and all, but remember: You’re all on the same airplane.

Getting Super Elite priority boarding on the 4pm flight to San Francisco won’t get you to San Francisco ahead of the 1pm flight. That flight left three hours ago, and in fact it already landed even before you set foot on the airplane.

Using the `init_seg` pragma lets you tweak the order in which objects are constructed and destructed, relative to other objects in the module. But the issue here is that the module itself is being terminated too late. You're all on the same airplane flight, and that flight takes off too late. Nothing you do with your flight is going to get you there ahead of that other module, because that other module already landed.

A more mundane explanation as to why the `init_seg(user)` pragma won't help is that `init_seg(user)` is the default initialization segment. If you don't specify an initialization segment, then you get `user`. Explicitly asking for `user` doesn't change anything. That option is provided for completeness.

Raymond Chen

Follow