Another reason not to do anything scary in your DllMain: Inadvertent deadlock

devblogs.microsoft.com/oldnewthing/20040128-00

January 28, 2004

Raymond Chen

Your DllMain function runs inside the loader lock, one of the few times the OS lets you run code while one of its internal locks is held. This means that you must be extra careful not to violate a lock hierarchy in your DllMain; otherwise, you are asking for a deadlock.

(You do have a lock hierarchy in your DLL, right?)

The loader lock is taken by any function that needs to access the list of DLLs loaded into the process. This includes functions like GetModuleHandle and GetModuleFileName. If your DllMain enters a critical section or waits on a synchronization object, and that critical section or synchronization object is owned by some code that is in turn waiting for the loader lock, you just created a deadlock:

```
// global variable
CRITICAL_SECTION g_csGlobal;
// some code somewhere
EnterCriticalSection(&g_csGlobal);
... GetModuleFileName(MyInstance, ..);
LeaveCriticalSection(&g_csGlobal);
BOOL WINAPI
DllMain(HINSTANCE hinstDLL, DWORD fdwReason,
        LPVOID lpvReserved)
{
  switch (fdwReason) {
  ...
  case DLL_THREAD_DETACH:
   EnterCriticalSection(&g_csGlobal);
   ...
  }
  ...
}
```

Now imagine that some thread is happily executing the first code fragment and enters g_csGlobal, then gets pre-empty. During this time, another thread exits. This enters the loader lock and sends out DLL_THREAD_DETACH messages while the loader lock is still

held.

You receive the DLL_THREAD_DETACH and attempt to enter your DLL's g_csGlobal. This blocks on the first thread, who owns the critical section. That thread then resumes execution and calls GetModuleFileName. This function requires the loader lock (since it's accessing the list of DLLs loaded into the process), so it blocks, since the loader lock is owned by somebody else.

Now you have a deadlock:

- g_cs owned by first thread, waiting on loader lock.
- Loader lock owned by second thread, waiting on g_cs.

I have seen this happen. It's not pretty.

Moral of the story: Respect the loader lock. Include it in your lock hierarchy rules if you take any locks in your DllMain.

Raymond Chen

Follow