If only DLLs can get DllMain notifications, how can an EXE receive a notification when a thread is created (for example)?

devblogs.microsoft.com/oldnewthing/20141016-00

October 16, 2014

Raymond Chen

When a DLL is loaded, it receives a `DLL_PROCESS_ATTACH` notification, and when it is unloaded (or when the process terminates), it gets a `DLL_PROCESS_DETACH` notification. DLLs also receive `DLL_THREAD_ATTACH` notifications when a thread is created and `DLL_THREAD_DETACH` notifications when a thread exits. But what if you are an EXE? EXEs don't have a `DllMain`, so there is no way to receive these notifications.

The trick here is to hire a lackey.

Create a helper DLL, called, say, `LACKEY.DLL`. Your EXE links to the lackey, and the lackey's job is to forward all `DllMain` notifications back to your EXE. The DLL would naturally have to have a way for your EXE to provide the callback address, so you might have a function `RegisterLackeyCallback`.

```
typedef BOOL (CALLBACK *LACKEYNOTIFICATION)(DWORD dwReason);
LACKEYNOTIFICATION g_lackeyNotification;
void RegisterLackeyCallback(LACKEYNOTIFICATION lackeyNotification)
{
 g_lackeyNotification = lackeyNotification;
}
BOOL WINAPI DllMain(
    HINSTANCE hinstDLL, DWORD dwReason, LPVOID lpReserved)
{
 if (g_lackeyNotification) g_lackeyNotification(dwReason);
 return TRUE;
}
```

Of course, it is rather extravagant to hire a lackey just for this one task, so you will probably just add lackey responsibilities to some other DLL you've written.

I don't know if there's a name for this design pattern, so I'm just going to call it the *hired lackey* pattern.

Raymond Chen

Follow