The management of memory for resources in 16-bit Windows

devblogs.microsoft.com/oldnewthing/20040202-00

February 2, 2004

Raymond Chen

In a previous entry I threatened to discuss the way resources were managed in 16-bit Windows.

In 16-bit Windows, resources were not loaded until explicitly requested.

- The FindResource function located the entry for the resource in the resource directory and returned it in the form of a HRSRC.
- The LoadResource function took that resource handle, allocated some movable memory (HGLOBAL), and loaded the referenced resources off the disk into that memory.
- The LockResource function took that global handle and locked it, returning a pointer to the resource bytes themselves.
- The UnlockResource function unlocked the global handle.
- The FreeResource function freed the memory that had been allocated to hold the resource.

Actually, it was more complicated than this. Additional bookkeeping ensure that if two people tried to load the same resource, the same memory block got used for both, and the FreeResource didn't actually free the memory until the reference count went back to zero.

Actually, it was even more complicated than this. If the resource was marked DISCARDABLE, then the memory wasn't actually freed when the reference count dropped to zero. Instead, the global handle was marked as discardable (GMEM_DISCARDABLE), so the handle remained valid, but when the system came under memory pressure, the memory behind the handle would get freed, and the next time you did a LoadResource, it would get reloaded from disk.

So now you know what that DISCARDABLE keyword in resource files means. Or at least what it used to mean. Win32 doesn't do any of this; the DISCARDABLE flag is ignored but remains for compatibility.

Raymond Chen

Follow