What was the point of the GMEM_SHARE flag?

devblogs.microsoft.com/oldnewthing/20041102-00

November 2, 2004

Raymond Chen

The GlobalAlloc function has a GMEM_SHARE flag. What was it for?

In 16-bit Windows, the GMEM_SHARE flag controlled whether the memory should outlive the process that allocated it. By default, all memory allocated by a process was automatically freed when that process exited.

Passing the GMEM_SHARE flag suppressed this automatic cleanup. That's why you had to use this flag when allocating memory to be placed on the clipboard or when you transfer it via OLE to another process. Since the clipboard exists after your program exits, any data you put on the clipboard needs to outlive the program. If you neglected to set this flag, then once your program exited, the memory that you put on the clipboard would be cleaned up, resulting in a crash the next time someone tried to read that data from the clipboard.

(The GMEM_SHARE flag also controlled whether the memory could be freed by a process other than the one that allocated it. This makes sense given the above semantics.)

Note that the cleanup rule applies to global memory allocated by DLLs on behalf of a process. Authors of DLLs had to be careful to keep track of whether any particular memory allocation was specific to a process (and should be freed when the process exited) or whether it was something the DLL was planning on sharing across processes for its own internal bookkeeping (in which case it shouldn't be freed). Failure to be mindful of this distinction led to bugs like this one.

Thank goodness this is all gone in Win32.

Raymond Chen

Follow