What's the point of using a custom timer queue if there is a default one already?

devblogs.microsoft.com/oldnewthing/20150219-00

February 19, 2015

Raymond Chen

A customer observed that when you create a timer via `CreateTimerQueueTimer`, you have the option of passing either `NULL` to get the default timer queue or a handle to a custom timer queue created by the `CreateTimerQueue` function. Why would you want to create a custom timer queue? If you create a custom timer queue, then you know that all the timers in the queue are yours. For example, there might be another component in the process that creates a lot of timers in the default timer queue. If you had put your timers in the default timer queue, then your timers will risk starvation. It's the same sort of logic that might lead you to create a custom heap instead of using the default heap: You create a custom heap if you want to isolate your allocations from other components. Another component that creates a lot of allocations in the default heap may create fragmentation pressure in the default heap, but your custom heap is unaffected. Note, however, that in both cases (timer starvation and heap fragmentation), this is not something you typically worry about because the cost of the insurance often doesn't outweigh the benefit. For example, if you create your own timer queue, then the threads associated with your timer queue cannot be used to service timers from the default timer queue and vice versa, resulting in more active threads in the process than necessary, which increases scheduler overhead and memory usage. Since the threads from the two timer queues are not coordinating their efforts (since that's the point of creating a private timer queue), you can end up with CPU thrashing if both timer queues have collectively created more threads than there are CPU cores, and they are all running. After all, the whole point of a timer queue in the first place is to consolidate a lot of little pieces of work into a small number of threads. Creating a private timer queue is moving in the opposite direction. But if you want to do it, then the facility is there.

For example, if you create a private heap, then you can free all the allocations in that heap by destroying the heap. Similarly, having a private timer queue means that you can cancel and delete all the timers by deleting the timer queue. Though at least in the timer case, you can get the best of both worlds (shared timer queue plus bulk cancellation) by associating all the timers with a cleanup group.

Raymond Chen

Follow