Function requirements are cumulative: If you fail to meet any of them, then all bets are off

devblogs.microsoft.com/oldnewthing/20110323-00

March 23, 2011

Raymond Chen

A customer was having problems with the `WaitForMultipleObjects` function:

> We are looking for a clarification of the behavior of `WaitForMultipleObjects`. We have a thread that waits on two handles (call them `Handle1` and `Handle2`) with `WaitFor-MultipleObjects`, `bWaitAll = FALSE`. Under certain conditions, we signal `Handle1` and close `Handle2` from another thread while the wait is in progress. This results in `WAIT_FAILED` being returned from the wait call. MSDN is not clear on what the expected behavior is here. On the one hand, it says
>
> A. When `bWait` is `FALSE`, this function checks the handles in the array in order starting with index 0, until one of the objects is signaled. If multiple objects become signaled, the function returns the index of the first handle in the array whose object was signaled.
>
> This description implies that the wait should never fail, because the function should have noticed that `Handle1` is signaled before it got around to noticing that `Handle2` was invalid.
>
> On the other hand, the documentation also says
>
> B. If one of these handle is closed while the wait is still pending, the function's behavior is undefined.
>
> What behavior is guaranteed here?

Once you violate a constraint (in this case, the constraint that the handles remain valid for the duration of the wait operation), you have failed to meet the prerequisites and the behavior is undefined. You can't say, "Well, sure you say I can't do X, but if you follow exactly the steps given in this description of how signaled objects are selected, then the function wouldn't even have noticed X before coming to a decision, so the fact that I broke one of the rules is irrelevant!" The description of the behavior of the `WaitForMultipleObjects` function when `bWait` is `FALSE` is not an implementation specification. It's a description of how to interpret the behavior of the function. The algorithmic way the function behavior is described is to assist in understanding the behavior; it doesn't mean that the function

actually follows the algorithm step-by-step. (In reality, there is no polling loop, as the algorithmic description implies. All the handles are waited on simultaneously. It's like Lebesgue integration: You integrate over the entire domain at once.) An algorithm-free description of the behavior when `bWait` is false might go like this:

A. When `bWait` is `FALSE`, the function does not return until one of the handles in the array becomes signaled. The return value is the smallest index corresponding to a signaled handle.

This description is purely declarative but gives you no mental model. It's like saying that "Water seeks its own level." Water doesn't have a will that compels it to behave in a certain way, but describing the behavior in that way makes reasoning about water easier.

Raymond Chen

Follow