If more than one object causes a WaitForMultipleObjects to return, how do I find out about the other ones?

devblogs.microsoft.com/oldnewthing/20150409-00

April 9, 2015

Raymond Chen

There is often confusion about the case where you call `WaitForMultipleObjects` (or its variants), passing `bWaitAll = FALSE`, and more than one object satisfies the wait.

> When *bWaitAll* 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.
>
> The function modifies the state of some types of synchronization objects. Modification occurs only for the object or objects whose signaled state caused the function to return.

The worry here is that you pass `bWaitAll = FALSE`, and two objects satisfy the wait, and they are both (say) auto-reset events.

The first paragraph says that the return value tells you about the event with the lowest index. The second paragraph says that the function modifies the state of the "object or objects" whose signaled state caused the function to return.

The "or objects" part is what scares people. If two objects caused the function to return, and I am told only about one of them, how do I learn about the other one?

The answer is, "Don't worry; it can't happen."

If you pass `bWaitAll = FALSE`, then only one object can cause the function to return. If two objects become signaled, then the function declares that the lowest-index one is the one that caused the function to return; the higher-index ones are considered not have have caused the function to return.

In the case of the two auto-reset events: If both events are set, one of them will be chosen as the one that satisfies the wait (the lower-index one), and it will be reset. The higher-index one remains unchanged.

The confusion stems from the phrase “object or objects”, causing people to worry about the case where `bWaitAll = FALSE` and there are multiple objects which cause the function to return.

The missing detail is that when you pass `bWaitAll = FALSE`, then at most one object can cause the function to return. (“At most”, because if the operation times out, then no object caused the function to return.)

The presence of the phrase “or objects” is to cover the case where you pass `bWaitAll = TRUE`.

Raymond Chen

Follow