An auto-reset event is just a stupid semaphore

devblogs.microsoft.com/oldnewthing/20060622-00

June 22, 2006

Raymond Chen

When you create an event with the `CreateEvent` function, you get to specify whether you want an auto-reset event or a manual-reset event.

Manual-reset events are easy to understand: If the event is clear, then a wait on the event is not satisfied. If the event is set, then a wait on the event succeeds. Doesn't matter how many people are waiting for the event; they all behave the same way, and the state of the event is unaffected by how many people are waiting for it.

Auto-reset events are more confusing. Probably the easiest way to think about them is as if they were semaphores with a maximum token count of one. If the event is clear, then a wait on the event is not satisfied. If the event is set, then one waiter succeeds and the event is reset; the other waiters keep waiting. (And from our discussion of `PulseEvent`, you already know that it is indeterminate which waiter will be released if there is more than one.)

The gotcha with auto-reset events is the case where you set an event that is already set. Since an event has only two states (set and reset), setting an event that is already set has no effect. If you are using an event to control a resource producer/consumer model, then the "setting an event that is already set" case will result in you appearing to "lose" a token. Consider the following intended pattern:

Producer	Consumer
	Wait
Produce work	
`SetEvent`	
	Wake up and reset event
	Do work
Produce work	

	Wait
`SetEvent`	
	Wake up and reset event
	Do work
…	…

But what if the timing doesn't quite come out? What if the consumer thread is a little slow to do the work (or the producer thread is a little fast in generating it):

Producer	**Consumer**
	Wait
Produce work	
`SetEvent`	
	Wake up and reset event
Produce work	
`SetEvent`	
	Do work
Produce work	
`SetEvent` (has no effect)	
	Wait satisfied immediately
	Reset event
	Do work
	Wait

Notice that the producer produced three work items, but the consumer performed only two of them. The third `SetEvent` had no effect since the event was already set. (You have the same problem if you try to increase a semaphore's token count past its maximum.) If you want the number of wakes to match the number of sets, then you need to use a semaphore with a maximum token count as high as the maximum number of outstanding work items you will support.

Moral of the story: Know your tools, know their limits, and use the right tool for the right job.

Raymond Chen

Follow