Comparing WaitOnAddress with futexes (futexi? futexen?)

devblogs.microsoft.com/oldnewthing/20170601-00

June 1, 2017

Raymond Chen

Linux has a synchronization primitive called a *futex* which is similar to Windows's `WaitOnAddress` in that both let you create a synchronization object out of nothing. (Well, okay, you need to set aside some memory.)

The basic operations on a futex depend on what color glasses you're wearing. If you are wearing syscall-colored glasses, then the basic operations are WAIT and WAKE. But we're going to wear user-mode-colored glasses, in which the basic operations are UP and DOWN. (In futex-speak, "up" and "down" are verbs. And you thought Microspeak was weird.)

Conceptually, a futex is a 32-bit variable that acts like a semaphore with a maximum count of 1.

To initialize a futex, set it to 0 if you want it to be initially unavailable, or 1 if you want it to be initially available.

"Downing" a futex is what you do when you want to wait for the futex to become available. Atomically decrement the futex, and if the result is 0, then you have successfully claimed the futex. Otherwise, call into the kernel to wait for the futex to become available. (This isn't perfectly accurate, but bear with me.)

"Upping" a futex is what you do when you want to make the futex available. Atomically increment the futex, and if the result is 1, then there are no waiters, and you're done. Otherwise, set it to 1 explicitly (indicating that the futex is now free) and call into the kernel to tell it to release the waiting threads. (It is an error to "up" a futex that is already available.)

In theory, the values of a futex correspond to these states:

- 1 means that the futex is available.
- 0 means that the futex is unavailable, and there are no waiters.
- A negative number means that the futex is unavailable, and the number of waiters is the negative of the value. For example, a value of –2 means that there are two waiters.

In practice, however, any negative number means that the futex is unavailable. The actual number of waiting threads is kept in the kernel. The actual rule for "downing" a futex is that if you down it to a negative number, you should set the futex to −1 and then call into the kernel. Forcing the value to −1 avoids underflow if more than 2 billion threads wait on the futex.[1]

A futex, therefore, is a lightweight version of an auto-reset event. There is at most one token, and the only operations are to claim the token (waiting as necessary for a token to become available) and to release the token.

An obvious difference between futexes and `WaitOnAddress` is that a futex mimics an auto-reset event (or a single-token semaphore, which is basically the same thing), whereas `WaitOnAddress` can be used to mimic a large number of things depending on how you use it.

A notable difference between futexes and `WaitOnAddress` is that the contents of the futex are not your choice; the contents are controlled by the rules for futexes. If you have other information you want to keep track of, you need to keep it somewhere outside the futex. On the other hand, `WaitOnAddress` lets you put anything you want in the memory, and it is typically some central bookkeeping information that you would have needed to keep track of anyway.

A big difference is that futexes work across processes (via shared memory), whereas `WaitOnAddress` works only within a process.

Here's a summary table, because people like tables.

	futex	**WaitOnAddress**
Size	4 bytes	1, 2, 4, or 8 bytes (your choice)
Contents	Controlled by futex	Controlled by you
Access	Must use atomic primitives	No restrictions
Scope	Can be cross-process	Limited to a single process
Operations	"Down" (wait for object)	Wait for value to change
	"Up" (release object, wake waiters)	Wake waiters
Acts like	Event (or single-token semaphore)	Whatever you want
Spurious wakes?	Yes	Yes

Both primitives enter kernel mode only if they need to wait or wake. If there is no contention, then both primitives operate entirely in user mode.

[1] If you can get 2 billion threads to wait on a futex, then I'm both impressed and disappointed. Impressed that you were able to create 2 billion threads in the first place, and disappointed that you have a futex so hot that you managed to get 2 billion threads waiting on it. You should fix that.

Raymond Chen

Follow