Spurious wakeup

A **spurious wakeup** happens when a thread wakes up from waiting on a condition variable that's been signaled, only to discover that the condition it was waiting for isn't satisfied. It's called spurious because the thread has seemingly been awakened for no reason. But spurious wakeups don't happen for no reason: they usually happen because, in between the time when the condition variable was signaled and when the waiting thread finally ran, another thread ran and changed the condition. There was a race condition between the threads, with the typical result that sometimes, the thread waking up on the condition variable runs first, winning the race, and sometimes it runs second, losing the race.

On many systems, especially multiprocessor systems, the problem of spurious wakeups is exacerbated because if there are several threads waiting on the condition variable when it's signaled, the system may decide to wake them all up, treating every `signal( )` to wake one thread as a `broadcast( )` to wake all of them, thus breaking any possibly expected 1:1 relationship between signals and wakeups.[1] If there are ten threads waiting, only one will win and the other nine will experience spurious wakeups.

To allow for implementation flexibility in dealing with error conditions and races inside the operating system, condition variables may also be allowed to return from a wait even if not signaled, though it is not clear how many implementations actually do that. In the Solaris implementation of condition variables, a spurious wakeup may occur without the condition being signaled if the process is signaled; the wait system call aborts and returns `EINTR`.[2] The Linux p-thread implementation of condition variables guarantees it will not do that.[3][4]

Because spurious wakeups can happen whenever there's a race and possibly even in the absence of a race or a signal, when a thread wakes on a condition variable, it should always check that the condition it sought is satisfied. If it is not, it should go back to sleeping on the condition variable, waiting for another opportunity.

References

1. Raymond Chen (February 1, 2018). "Spurious wake-ups in Win32 condition variables" (https://devblogs.microsoft.com/oldnewthing/20180201-00/?p=97946). Retrieved May 9, 2020.
2. "Interrupted Waits on Condition Variables (Solaris Threads Only)" (https://docs.oracle.com/cd/E19455-01/806-5257/gen-24356/index.html). Oracle Corporation. Retrieved May 9, 2020.
3. "pthread_cond_wait(3) - Linux man page" (https://linux.die.net/man/3/pthread_cond_wait). die.net. Retrieved May 9, 2020. "These functions shall not return an error code of [EINTR]."
4. "pthread_cond_timedwait, pthread_cond_wait - wait on a condition" (http://pubs.opengroup.org/onlinepubs/9699919799/functions/pthread_cond_timedwait.html). The Open Group. 2018. Retrieved May 9, 2020.