

Re: A new Critical Section for high contention situations

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.kernel/2008-05/msg00226.htm>

- *From:* "Kürpat" <xx@xxxxxx>
 - *Date:* Fri, 16 May 2008 09:58:53 +0300
-

Yes, you are right. I implemented the class for a certain problem. I need to protect a queue which is pounded by a number of threads. To synchronize enqueues and dequeues I used CRITICAL_SECTION but because of high contention kernel transition was high and performance was low. Anyway, your warning is very important for all general purpose implementations.

"Remy Lebeau" <no.spam@xxxxxxxxxxxx> wrote in message
<news:OLIA73xtIHA.4560@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>

"Kürsat" <xx@xxxxxx> wrote in message
<news:O3NzUVrtIHA.3716@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>

Oh my God! I am so sorry, using "if" in place of "while" makes almost every code illogical. I am sorry again, below is updated code:

Your code is still missing an important feature that every synchronization object MUST support. If a thread calls enter() and then calls enter() again before calling leave() (such as from nested code making local calls to enter/leave()), enter() MUST EXIT WITHOUT BLOCKING, and the lock must NOT be released until leave() is called TWICE. Otherwise a deadlock occurs. Your code is not handling those possibilities yet. The first time enter() is called, your lock is NOT_OWNED, so it gets updated to OWNED and enter() exits. However, the next time enter() is called on the same thread, the lock may still be OWNED, and you would then enter an endless loop since the owning thread can never call leave(). The only possibility would be if another thread called leave(), but that would be impossible since enter() could never exit in those threads, either.

Gambit

Re: A new Critical Section for high contention situations