

Re: WaitForSingleObject() will not deadlock

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2007-07/msg00561.html>

- *From:* Joseph M. Newcomer <newcomer@xxxxxxxxxxxxx>
 - *Date:* Fri, 06 Jul 2007 16:28:43 -0400
-

So EDEADLCK is a condition that can be detected in no more than 2 CPU clock cycles, right? I'd like to see the EXACT SEQUENCE OF INSTRUCTIONS issued in the locking sequence, and I would have to compare that with the EXACT SEQUENCE OF INSTRUCTIONS issued by, say, a CRITICAL_SECTION, before I'd be willing to opine that one is faster than the other.

My issue about the 2 CPU clock cycles is that once the lock is set, you can test the is-visited flag in two instructions. So the cost of a successful lock must be identical to the cost of a failed lock, and this has to be compared to the cost of a successful recursive lock.

One thing I really learned during my 15 years of doing performance measurement is that without actual live data to back an opinion about performance, any opinion about performance is completely useless. Only measurements matter. I'm willing to grant that the non-recursive mutex MIGHT be faster, but without substantiating data and controlled experiments to validate that data, I have no reason to believe that this is faster than the use of a recursive mutex and an is-visible flag. And I have no reason to believe that the recursive mutex is faster, either.

Note the exact sequence of instructions can still be misleading because of issues of cache writeback required, so a single instruction with a LOCK prefix can be 20-100 times slower than the same instruction without the LOCK prefix. Maybe. But instruction counts are not a bad first approximation.

joe

On Thu, 05 Jul 2007 22:59:25 -0700, Frank Cusack <fcusack@xxxxxxxxxxxxx> wrote:

On Thu, 05 Jul 2007 16:50:31 -0700 Frank Cusack <fcusack@xxxxxxxxxxxxx> wrote:

I will post code.

cycle detection using non-recursive mutex:

```
[frank@shell:~]$ cat cycle.c
```

```
/*  
 * cycle.c  
 */
```

```
#ifndef _REENTRANT
```


Re: WaitForSingleObject() will not deadlock

more robust than one that can do it with just one pass?

```
    }

    int
    main(int argc, char *argv[])
    {
    int i;
    struct node_t *head, *node;
    /* we will create a cycle 9->4 */
    struct node_t *node4, *node9;

    pthread_mutexattr_t attr;
    if (pthread_mutexattr_init(&attr))
    exit(1);
    /* make sure we create mutexes that detect deadlock */
    if (pthread_mutexattr_settype(&attr, PTHREAD_MUTEX_ERRORCHECK))
    exit(1);

    head = malloc(sizeof(struct node_t));
    (void) pthread_mutex_init(&head->mutex, &attr); /* assume success */
    head->val = 0;
    head->next = NULL;

    node = head;
    for (i = 1; i < 10; ++i) {
    node->next = malloc(sizeof(struct node_t));
    if (!node->next)
    exit(1);
    node = node->next;
    (void) pthread_mutex_init(&node->mutex, &attr); /* assume success */
    node->val = i;
    if (i == 4)
    node4 = node;
    else if (i == 9)
    node9 = node;
    }

    detect_cycle(head);
    /* create a cycle */
    node9->next = node4;
    detect_cycle(head);

    return 0;
    }
[frank@shell:~]$ gcc -o cycle cycle.c -lpthread
[frank@shell:~]$ ./cycle
no cycle detected
cycle detected
[frank@shell:~]$
```

Re: WaitForSingleObject() will not deadlock

-frank

Joseph M. Newcomer [MVP]

email: newcomer@xxxxxxxxxxxxx

Web: <http://www.flounder.com>

MVP Tips: http://www.flounder.com/mvp_tips.htm

.