

Re: [RFC, PATCH] locks: remove posix deadlock detection

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-10/msg10819.html>

- *From:* Trond Myklebust <trond.myklebust@xxxxxxxxxxx>
 - *Date:* Sun, 28 Oct 2007 23:26:12 -0400
-

On Sun, 2007-10-28 at 16:41 -0600, Matthew Wilcox wrote:

On Sun, Oct 28, 2007 at 05:50:30PM -0400, Trond Myklebust wrote:

You can't fix the false EDEADLK detection without solving the halting problem. Best of luck with that.

I can see that it would be difficult to do efficiently, but basically, this boils down to finding a circular path in a graph. That is hardly an unsolvable issue...

Bzzt. You get a false deadlock with multiple threads like so:

Thread A of task B takes lock 1
Thread C of task D takes lock 2
Thread C of task D blocks on lock 1
Thread E of task B blocks on lock 2

We currently declare deadlock at this point (unless the deadlock detection code has changed since I last looked at it), despite thread A being about to release lock 1. Oh, and by the way, thread E is capable of releasing lock 1, so you can't just say "well, detect by thread instead of by task".

So the only way I can see to accurately detect deadlock is to simulate the future execution of all threads in task B to see if any of them will release lock 1 without first gaining lock 2. Which, I believe, is halting-equivalent.

As several people have told you, the SUSv3 section on fcntl and deadlocks reads as follows:

"A potential for deadlock occurs if a process controlling a locked region is put to sleep by attempting to lock another

Re: [RFC, PATCH] locks: remove posix deadlock detection

process' locked region. If the system detects that sleeping until a locked region is unlocked would cause a deadlock, fcntl() shall fail with an [EDEADLK] error."

There is no mention there or anywhere else of a need to make exceptions when dealing with threads. The posix locking model is `_process_` based, and so our deadlock detection only needs to take that into account. If programmers choose to play tricky little games with threads, then it is their responsibility to ensure that the application doesn't get into a situation where the posix deadlock detection model breaks down.

Trond

—

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxx
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>