Driver porting: the preemptible kernel

[Posted February 24, 2003 by corbet]

This article is part of the LWN Porting Drivers to 2.6 series.

One significant change introduced in 2.6 is the preemptible kernel. Previously, a thread running in kernel space would run until it returned to user mode or voluntarily entered the scheduler. In 2.6, if preemption is configured in, kernel code can be interrupted at (almost) any time. As a result, the number of challenges relating to concurrency in the kernel goes up. But this is actually not that big a deal for code which was written to handle SMP properly - most of the time. If you have not yet gotten around to implementing proper locking for your 2.4 driver, kernel preemption should give you yet another reason to get that job done.

The preemptible kernel means that your driver code can be preempted whenever the scheduler decides there is something more important to do. "Something more important" could include re-entering your driver code in a different thread. There is one big, important exception, however: preemption will not happen if the currently-running code is holding a spinlock. Thus, the precautions which are taken to ensure mutual exclusion in the SMP environment also work with preemption. So most (properly written) code should work correctly under preemption with no changes.

That said, code which makes use of per-CPU variables should take extra care. A per-CPU variable may be safe from access by other processors, but preemption could create races on the same processor. Code using per-CPU variables should, if it is not already holding a spinlock, disable preemption if the possibility of concurrent access exists. Usually, macros like `get_cpu_var()` should be used for this purpose.

Should it be necessary to control preemption directly (something that should happen rarely), some macros in `<linux/preempt.h>` will come in helpful. A call to `preempt_disable()` will keep preemption from happening, while `preempt_enable()` will make it possible again. If you want to re-enable preemption, but don't want to get preempted immediately (perhaps because you are about to finish up and reschedule anyway), `preempt_enable_no_resched()` is what you need.

Normally, rescheduling by preemption takes place without any effort on the part of the code which is being scheduled out. Occasionally, however, long-running code may want to check explicitly to see whether a reschedule is pending. Code which defers rescheduling with `preempt_enable_noresched()` may want to perform such checks, for example, when it reaches a point where it can afford to sleep for a while. For such situations, a call to `preempt_check_resched()` will suffice.

One interesting side-effect of the preemption work is that it is now much easier to tell if a particular bit of kernel code is running within some sort of critical section. A single variable in the task structure now tracks the preemption, interrupt, and softirq states. A new macro, `in_atomic()`, tests all of these states and returns a nonzero value if the kernel is running code that should complete without interruption. Among other things, this macro has been used to trap calls to functions that might sleep from atomic contexts.

(Log in to post comments)

Driver porting: the preemptible kernel

Posted Mar 30, 2003 17:27 UTC (Sun) by **goatbar** (guest, #10402) [Link]

I want to start off by saying how much I appreciate these series of articles. Thank you!

I haven't even come close to using 2.5 yet, but I have been using kernels >= 2.4.18 with the RML preempt patch. Is this patch the same or pretty close to the preemptible code in the 2.5.xx kernels?

Reply to this comment

typo: preempt_enable_noresched()

Posted Feb 10, 2004 9:42 UTC (Tue) by **scottt** (subscriber, #5028) [Link]

should be preempt_enable_no_resched()

Reply to this comment

Driver porting: the preemptible kernel

Posted Oct 13, 2004 12:14 UTC (Wed) by **Wipro** (guest, #25350) [Link]

Thanks for an excellent article.

I want to know how to test whether a given driver module is SMP safe and pre-emption safe or not?

Reply to this comment

Driver porting: the preemptible kernel

Posted Nov 29, 2004 14:14 UTC (Mon) by **giomanunta** (guest, #26281) [Link]

Thanks a lot

Reply to this comment