

## locking down kqueue (was some other completely unrelated topic)

*Source:* <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/arch/2004-04/0053.html>

---

*From:* John-Mark Gurney ([gurney\\_j\\_at\\_efn.org](mailto:gurney_j_at_efn.org))

*Date:* 04/16/04

Date: Thu, 15 Apr 2004 15:39:20 -0700  
To: "Brian F. Feldman" <[green@freebsd.org](mailto:green@freebsd.org)>

Brian F. Feldman wrote this message on Thu, Apr 15, 2004 at 16:36 -0400:

> *John-Mark Gurney* <[gurney\\_j@efn.org](mailto:gurney_j@efn.org)> wrote:  
> > *How does this sound to people? I have some code starting to implement*  
> > *this, but I haven't gotten very far with it yet...*  
>  
> *You know, now I think Seigo's right on the money that due to the nature of*  
> *the recursion of kqueue's current implementation, it's impossible to get*  
> *right with this train of thought. So, let's redesign:*  
> *\* The kqueue object the user controls needs a mutex.*  
> *\* The lists (selinfo, mostly) that knotes are on need locking.*  
> *\* The filterops that kqueue calls out MUST be called with some*  
> *kind of locking on the lists that the kqueue is on, and the*  
> *user MUST be able to grab any kind of lock from inside a*  
> *filterop.*

You need to be more specific on this.. which filterops should be allowed to grab any locks? In my opinion, filterops should only be allowed to grab a limited number of locks and these are the object lock and the kqueue (list) locks as necessary...

> *\* At some point the object being observed must call back into*  
> *kqueue to add itself. We'll end up getting deadlocks if the*  
> *locks kqueue holds are not the ones required to add the*  
> *object to klists and held when we do the f\_attach().*

Yes, we need to prevent lock order inversion..

> *\* Anything that calls KNOTE() or KNOTE\_ACTIVATE() directly*  
> *will end up recursing back on itself if we don't convert*  
> *it to putting the new event on a work queue. How*  
> *filt\_procattach() calls KNOTE\_ACTIVATE() or filt\_proc()*  
> *calls kqueue\_register() is a very good example.*

Hmmm. I'm going to have to mull on the filt\_proc problem a bit..

freebsd-arch: locking down kqueue (was some other completely unrelated topic)

- > \* *The functions that need to be exported are:*
- > \* *KNOTE()/KNOTE\_ACTIVATE() <- put on a workqueue*
- > \* *kqueue\_register() <- put on a workqueue*
- > \* *knote klist/(kn\_selnext) linking and unlinking*
- > \* *knote klist/(kn\_selnext) is disappearing*
- > *The last two are the only ones that are not called recursively*
- > *and should have easy locking semantics.*

Personally, I'd prefer to invert the logic, and have linking/unlinking and disappearing done via a work queue, and KNOTE/kqueue\_register done in line if possible...

It's also difficult because we need to optimize for both cases of long existing events, and ONE\_SHOT events where we are constantly adding/removing events...

The reason I say this is because I have a visit for a webserver that uses multiple processors but a single kqueue to handle events, and if we use ONE\_SHOT, then we are guaranteed that we are notified of each event once... We need to make sure that the work queue will not be a significant problem...

- > *Nothing is currently designed to work with anything even remotely not*
- > *looking like spl(), so we have to either flatten it out (using workqueues)*
- > *or change semantics so that when KNOTE() is called it acts like the closure*
- > *that we pretend it is. Of course, the easy way to do this is with a worker*
- > *queue/condvar/mutex/thread. What other ways do we have available to turn*
- > *KNOTE() into a closure, bearing in mind that the entire point of the*
- > *mechanism is that there is no memory allocation at the time of event*
- > *generation -- only when events are defined (by the user or recursively by*
- > *other events).*

The proc case should be treated special as it is an [ab]use of the kevent system with following children... I'm looking at it more..

--

John-Mark Gurney

Voice: +1 415 225 5579

"All that I will do, has been done, All that I have, has not."

---

freebsd-arch@freebsd.org mailing list

<http://lists.freebsd.org/mailman/listinfo/freebsd-arch>

To unsubscribe, send any mail to "freebsd-arch-unsubscribe@freebsd.org"