

## Re: sched\_userret priority adjustment patch for sched\_4bsd

**Source:** <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/arch/2004-09/0211.html>

---

**From:** Stephan Uphoff ([ups\\_at\\_tree.com](mailto:ups_at_tree.com))

**Date:** 09/28/04

To: Julian Elischer <[julian@elischer.org](mailto:julian@elischer.org)>

Date: Mon, 27 Sep 2004 22:27:51 -0400

On Mon, 2004-09-27 at 18:10, Julian Elischer wrote:

> *Stephan Uphoff wrote:*

>

> > *On Mon, 2004-09-27 at 14:43, John Baldwin wrote:*

> >

> >

> >>>> @@ -272,7 +272,7 @@

> >>>> {

> >>>>

> >>>> *mtx\_assert(&sched\_lock, MA\_OWNED);*

> >>>> *- if (td->td\_priority < curthread->td\_priority)*

> >>>> *+ if (td->td\_priority < curthread->td\_ksegrp->kg\_user\_pri)*

> >>>> *curthread->td\_flags |= TDF\_NEEDRESCHED;*

> >>>> }

> >>>>

> >>>>

>

> *in sched\_userret() we do:*

> *kg = td->td\_ksegrp;*

> *if (td->td\_priority != kg->kg\_user\_pri) {*

> *mtx\_lock\_spin(&sched\_lock);*

> *td->td\_priority = kg->kg\_user\_pri;*

> *mtx\_unlock\_spin(&sched\_lock);*

> *}*

>

> *but we don't actually take any action in the case where the thread is heading out to userland with*

> *a priority of less importance than a waiting thread. That happens in*

> *AST() where we also set it down*

> *but only in the case of TDF\_NEEDRESCHED being set.*

>

> *it would make more sense to ALWAYS to the TDF\_NEEDRESCHED clause, in*

> *userret()*

> *based on the user priority... i.e. the priority would be reduced going*

> *to userland.*

freebsd-arch: Re: sched\_userret priority adjustment patch for sched\_4bsd

- > *Unfortunately this would stop one of the reasons to for priorityu*
- > *raising in BSD.*
- >
- > *The priority of a thread that waits for IO is raised not only to make it*
- > *start again in the kernel*
- > *as an interactive thread, but also so that it can run into userland too*
- > *and get some priority*
- > *for actually USING the new data/input..*

Thanks – I wasn't aware of this.

Isn't there a high potential for abuse?

A client/server programs constantly refreshing priority by waiting for requests/replies comes to mind. If a client/server pair constantly talks to each other they could eat a lot of cpu time.

I have to think about this some more.

Stephan

---

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"