Subject: Re: [RFC][PATCH] Make access to taks's nsproxy liter
Posted by Oleg Nesterov on Wed, 08 Aug 2007 17:19:55 GMT
View Forum Message <> Reply to Message

On 08/08, Eric W. Biederman wrote:
>
> Oleg Nesterov <oleg@tv-sign.ru> writes:
>
> > On 08/08, Pavel Emelyanov wrote:
> >>
> >> +void switch_task_namespaces(struct task_struct *p, struct nsproxy *new)
> >> +{
> >> + struct nsproxy *ns;
> >> +
> >> + might_sleep();
> >> +
> >> + ns = p->nsproxy;
> >> + if (ns == new)
> >> + return;
> >> +
> >> + if (new)
> >> + get_nsproxy(new);
> >> + rcu_assign_pointer(p->nsproxy, new);
> >> +
> >> + if (ns && atomic_dec_and_test(&ns->count)) {
> >> + /*
> >> + * wait for others to get what they want from this
> >> + * nsproxy. cannot release this nsproxy via the
> >> + * call_rcu() since put_mnt_ns will want to sleep
> >> + */
> >> + synchronize_rcu();
> >> + free_nsproxy(ns);
> >> + }
> >> +}
> >
> > (I may be wrong, Paul cc'ed)
> >
> > This is correct with the current implementation of RCU, but strictly speaking,
> > we can't use synchronize_rcu() here, because write_lock_irq() doesn't imply
> > rcu_read_lock() in theory.
>
> But we should be able to do:
>
> write_lock_irq();
> rcu_read_lock();
> muck with other tasks nsproxy.
> rcu_read_unlock();
> write_unlock_irq();

>
> Which would make rcu fine.

Yes sure. I just meant that the patch looks incomplete. But we didn't
hear Paul yet, perhaps I'm just wrong.

> The real locking we have is that only a task is allowed to modify it's
> own nsproxy pointer. Other processes are not.
>
> The practical question is how do we enable other processes to read
> a particular tasks nsproxy or something pointed to by it?

task_lock(). The only problem we can't take it in do_notify_parent(),
but if we add read_lock(tasklist) to sys_unshare, we can safely access
->parent->nsproxy.

Oleg.