Subject: Re: [PATCH] usbatm: Update to use the kthread api.
Posted by ebiederm on Thu, 19 Apr 2007 02:13:56 GMT
View Forum Message <> Reply to Message

Alan Stern <stern@rowland.harvard.edu> writes:

> On Tue, 2 Jan 2007, Christoph Hellwig wrote:
>
>> > I have a driver that spawns a kernel thread (using kthread_create) which
>> > does I/O by calling vfs_write and vfs_read. It relies on signals to
>> > interrupt the I/O activity when necessary. Maybe this isn't a good way of
>> > doing things, but I couldn't think of anything better.
>>
>> Given that we have no other way to interrupt I/O then signals at those
>> lower level I don't see a way around the singals if you stick to that
>> higher level design.
>
> Okay.
>
>> > P.S.: What is the reason for saying "signals should be avoided in kernel
>> > threads at all cost"?
>>
>> The probem with signals is that they can come from various sources, most
>> notably from random kill commands issues from userland. This defeats
>> the notion of a fixed thread lifetime under control of the owning module.
>> Of course this issue doesn't exist for you above useage where you'd
>> hopefully avoid allowing signals that could terminate the thread.
>
> In my case the situation is exactly the reverse: I _want_ to allow signals
> to terminate the thread (as well as allowing signals to interrupt I/O).
>
> The reason is simple enough. At system shutdown, if the thread isn't
> terminated then it would continue to own an open file, preventing that
> file's filesystem from being unmounted cleanly. Since people should be
> able to unmount their disks during shutdown without having to unload
> drivers first, the simplest solution is to allow the thread to respond to
> the TERM signal normally sent by the shutdown scripts.

You need a real user space interface. Historically user space is
required to call unmount and the like, you should have a proper
shutdown interface and script as well.

> Since the thread is owned by the kernel, random kill commands won't have
> any bad effect. Only kill commands sent by the superuser can terminate
> the thread.

I'm putting the final touches on a patchset to finish converting
all kernel threads to the kthread API. So I have had a chance

to digest the arguments.

Upgrading kthreads to allow them to handle the when a thread exits
on it's own, and to set signal_pending so they terminate interruptible
sleeps was easy.

Handling signals from user space in kernel threads is a maintenance
disaster. It makes the kernel thread part of the ABI and makes it
so you can never change the implementation if user space comes to
rely on using them. If we don't handle signals kernel threads
remain an implementation detail leaving us free to change the
implementation later.

A pid namespace trivially changes the implementation making all kernel
threads invisible. Which means you can't send a signal to them.

So since sending signals to kernel threads is weird in terms of
semantics, it prevents from changing implementation details, and
because no kernel thread in the kernel appears to actually require it
at this time I refuse to support the concept.

Eric

_____