Subject: Re: [PATCH] usbatm: Update to use the kthread api.
Posted by Alan Stern on Thu, 19 Apr 2007 14:56:51 GMT
View Forum Message <> Reply to Message

On Wed, 18 Apr 2007, Eric W. Biederman wrote:

> > 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.

Well, "kill <pid>" _is_ a real userspace interface.

Suppose you have a normal user application that runs a background process.
It's not a system service or anything like that, so it isn't controlled by
the runtime scripts in /etc/rc.d/init.d or wherever. How do you manage to
unmount the filesystems it uses at shutdown time? Simple -- you kill the
background process. Or rather, you rely on the shutdown script to kill
it for you.

My driver should work in the same way. Transparently. The user shouldn't
need to do anything special to shut down when the driver is loaded.

> 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.

What happens if there are user processes running in a PID namespace
different from the one used by the shutdown script? They won't get killed
at shutdown time, so the script won't be able to unmount the filesystems
they use.

> 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.

_My_ thread is in the kernel and it actually requires it. Unless you can
suggest a suitable alternative mechanism. For example, if you provided a
way for the thread to allow its PID always to show up in the PID namespace
used by the shutdown scripts, that would resolve the problem. Or if you
suggested a way for the thread to hold an open file reference that would
automatically be closed when the filesystem was unmounted, that would work
too.

The alternative I don't like at all is to tell people using the driver
that they need to add

 grep -q g_file_storage /proc/modules && rmmod g_file_storage

somewhere in their /etc/rc.d/init.d/halt script.

Alan Stern

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers