Subject: Re: [PATCH] kthread: saa7134-tvaudio.c
Posted by ebiederm on Wed, 30 Aug 2006 15:43:33 GMT
View Forum Message <> Reply to Message

Cedric Le Goater <clg@fr.ibm.com> writes:

>>> With the kthread api new kernel threads are started as children of keventd
>>> in well defined circumstances. If you don't do this kernel threads
>>> can wind up sharing weird parts of a parent process's resources and
>>> locking resources in the kernel long past the time when they are
>>> actually used by anything a user space process can kill.
>>>
>>> We have actually witnessed this problem with the kernels filesystem mount
>>> namespace. Mostly daemonize in the kernel unshares everything that
>>> could be a problem but the problem is sufficiently subtle it makes
>>> more sense to the change kernel threads. So these weird and subtle
>>> dependencies go away.
>>>
>>> So in essence the container work needs the new kthread api for the
>>> same reasons everyone else does it is just more pronounced in that
>>> case.
>>
>> That plus the obvious bit. For the pid namespace we have to declare
>> war on people storing a pid_t values. Either converting them to
>> struct pid * or removing them entirely. Doing the kernel_thread to
>> kthread conversion removes them entirely.
>
> we've started that war, won a few battles but some drivers need more work
> that a simple replace. If we could give some priorities, it would help to
> focus on the most important ones. check out the list bellow.

Sure, I think I can help.

There are a couple of test I can think of that should help.
1) Is the pid value stored. If not a pid namespace won't affect
   it's normal operation.

2) Is this thread started during kernel boot before this thread
   could have a user space parent. If it can't have a user space
   parent then it can't take a reference to user space resources.

3) Can the code be compiled modular and will it break when we stop
   exporting kernel_thread.

4) How frequently is this thing used. The more common code is probably
   in better shape and more likely to get a good maintainer response, and
   we care more :)

irqbalanced from arch/i386/kernel/io_apic.c should be safe to leave alone
because it doesn't store a pid_t, it is started during boot, and it can't
be compiled modular.

>From what I have seen you can shorten the list by several entries by removing
code like irqbalanced that can't possibly cause us any problems.
kvoyagerd from arch/i386/mach-voyager/voyager_thread.c is another one.

The first on my personal hit list is nfs.
> fs/lockd/clntlock.c
> fs/nfs/delegation.c
> net/sunrpc/svc.c

Because it does store pid_t values, it isn't started during kernel boot,
it can be compiled modular, and people use it all of the time.

I do agree from what I have seen, that changing idioms to the kthread way of
doing things isn't simply a matter of substitute and replace which is
unfortunate. Although the biggest hurdle seems to be to teach kernel threads
to communicate with something besides signals. Which is a general help anyway.

Unfortunately I'm distracted at the moment so I haven't gone through the entire
list but I hope this helps.

Eric

> arch/arm/kernel/ecard.c
> arch/i386/kernel/apm.c
> arch/i386/kernel/io_apic.c
> arch/i386/mach-voyager/voyager_thread.c
> arch/ia64/sn/kernel/xpc_main.c
> arch/mips/au1000/db1x00/mirage_ts.c
> arch/mips/kernel/apm.c
> arch/parisc/kernel/process.c
> arch/powerpc/platforms/pseries/eeh_event.c
> arch/powerpc/platforms/pseries/rtasd.c
> arch/s390/mm/cmm.c
> arch/sparc64/kernel/power.c
>
> drivers/base/firmware_class.c
> drivers/block/loop.c
> drivers/ieee1394/nodemgr.c
> drivers/macintosh/adb.c
> drivers/macintosh/mediabay.c
> drivers/macintosh/therm_pm72.c
> drivers/macintosh/therm_windtunnel.c
> drivers/media/dvb/dvb-core/dvb_ca_en50221.c
> drivers/media/dvb/dvb-core/dvb_frontend.c

> drivers/media/dvb/ttpci/av7110.c
> drivers/media/video/saa7134/saa7134-tvaudio.c
> drivers/media/video/tvaudio.c
> drivers/mmc/mmc_queue.c
> drivers/mtd/mtd_blkdevs.c
> drivers/net/wireless/airo.c
> drivers/pci/hotplug/cpci_hotplug_core.c
> drivers/pci/hotplug/cpqphp_ctrl.c
> drivers/pci/hotplug/ibmphp_hpc.c
> drivers/pci/hotplug/pciehp_ctrl.c
> drivers/pnp/pnpbios/core.c
> drivers/s390/net/lcs.c
> drivers/s390/net/qeth_main.c
> drivers/s390/scsi/zfcp_erp.c
> drivers/usb/atm/usbatm.c
> drivers/usb/storage/libusual.c
>
> fs/afs/cmservice.c
> fs/afs/kafsasyncd.c
> fs/afs/kafstimod.c
> fs/cifs/connect.c
> fs/jffs2/background.c
> fs/jffs/inode-v23.c
> fs/lockd/clntlock.c
> fs/nfs/delegation.c
>
> init/do_mounts_initrd.c
> kernel/kmod.c
> kernel/stop_machine.c
>
> net/bluetooth/bnep/core.c
> net/bluetooth/cmtp/core.c
> net/bluetooth/hidp/core.c
> net/bluetooth/rfcomm/core.c
> net/core/pktgen.c
> net/ipv4/ipvs/ip_vs_sync.c
> net/rxrpc/krxiod.c
> net/rxrpc/krxsecd.c
> net/rxrpc/krxtimod.c
> net/sunrpc/svc.c
> _____
> Containers mailing list
> Containers@lists.osdl.org
> https://lists.osdl.org/mailman/listinfo/containers

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers